

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA



PROYECTO FIN DE CARRERA

**APLICACIÓN PARA EL CONTROL DE LA ESTIMULACIÓN
CARDÍACA, PROTOCOLOS DE PRUEBAS DIAGNÓSTICAS Y
SEGUIMIENTO DE PACIENTES DE CARDIOLOGÍA DEL
HOSPITAL UNIVERSITARIO DE GRAN CANARIA DR. NEGRÍN**

Autor: Yaiza Santana Santana

Tutores: Dr. Luis Miguel Hernández Acosta

Dr. José María Quintero González

Fecha: Junio 2017

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA



PROYECTO FIN DE CARRERA

APLICACIÓN PARA EL CONTROL DE LA ESTIMULACIÓN
CARDÍACA, PROTOCOLOS DE PRUEBAS DIAGNÓSTICAS Y
SEGUIMIENTO DE PACIENTES DE CARDIOLOGÍA DEL
HOSPITAL UNIVERSITARIO DE GRAN CANARIA DR. NEGRÍN

HOJA DE FIRMAS

Alumno/a

Fdo.: Yaiza Santana Santana

Tutor

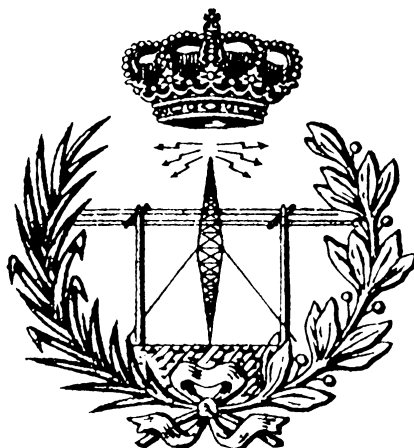
Fdo.: Dr. Luis Miguel
Hernández Acosta

Tutor

Fdo.: Dr. José María
Quinteiro González

Fecha: Junio 2017

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA



PROYECTO FIN DE CARRERA

APLICACIÓN PARA EL CONTROL DE LA ESTIMULACIÓN
CARDÍACA, PROTOCOLOS DE PRUEBAS DIAGNÓSTICAS Y
SEGUIMIENTO DE PACIENTES DE CARDIOLOGÍA DEL
HOSPITAL UNIVERSITARIO DE GRAN CANARIA DR. NEGRÍN

HOJA DE EVALUACIÓN

Calificación: _____

Presidente

Fdo.:

Vocal

Secretario/a

Fdo.:

Fdo.:

Fecha: Junio 2017

1. Introducción.

1.1 Descripción general del proyecto final de carrera.....	35
1.2 Contexto y ámbito del proyecto.....	35
1.3 Objetivos	39
1.4 Protocolos usados	40
1.5 Estructura de la memoria	41

2. Aplicación Web en su contexto.

2.1 Servicios webs	45
2.2 ¿Qué es una aplicación web?	45
2.3 Desarrollo ágil de aplicaciones web	48
2.4 Marcos de trabajo para aplicaciones web	51
2.5 Entornos de aplicación web: Desarrollo, test y producción.....	52

3. Planificación de Sistemas de Información PSI (Métrica 3).

3.1 Definición y objetivos del PSI	57
3.2 Participantes en las actividades del proceso PSI	58
3.3 Inicio del Plan de Sistemas de Información PSI	60
3.3.1 Tarea PSI 1.1: Análisis de la Necesidad del PSI	62
3.3.1.1 Solicitud formal de PSI	62
3.3.1.2 Aprobación de inicio del PSI	62
3.3.2 Tarea PSI 1.2: Identificación del alcance del PSI	63
3.3.2.1 Ámbito y objetivos del PSI	63
3.3.2.2 Objetivos estratégicos relacionados con el PSI	63
3.3.2.3 Factores críticos de éxito	64
3.3.3 Tarea PSI 1.3: Determinación de Responsables	64
3.4 Definición y organización del PSI	65
3.4.1 Tarea PSI 2.1: Especificación del Ámbito y Alcance	66
3.4.1.1 Descripción general de procesos de la organización afectados	67
3.4.1.2 Catálogo de objetivos de PSI	67
3.4.2 Tarea PSI 2.2: Organización del PSI.....	68
3.4.2.1 Estructura Organizativa (externo)	69
3.4.2.2 Catálogo de usuarios	70
3.4.2.3 Equipos de trabajo	70
3.4.3 Tarea PSI 2.3: Definición del Plan de Trabajo	71
3.4.3.1 Plan de trabajo	72
3.4.4 Tarea PSI 2.4: Comunicación del Plan de Trabajo	73

3.4.4.1 Aceptación del Plan de Trabajo	73
3.5 Estudio de información relevante y de los sistemas de información actuales	73
3.5.1 Tarea PSI 3.1: Selección y Análisis de Antecedentes	74
3.5.1.1 Información relevante	74
3.5.1.1.1 Información relevante sobre el entorno de trabajo de los Usuarios de Cardiología del HUGCDN	74
3.5.1.1.2 Información relevante sobre pruebas e intervenciones cardiológicas	75
3.5.1.1.3 Información relevante sobre dispositivos y sistemas implantados en el paciente	82
3.5.1.1.3.1 Generadores	82
3.5.1.1.3.2 Electrodos	84
3.5.1.1.4 Información relevante sobre seguimiento de pacientes	85
3.5.1.1.4.1 Evento médico	86
3.5.1.1.4.2 Programación bradicardia	87
3.5.1.1.4.3 Programación taquicardia	87
3.5.1.1.4.4 Nuevo problema	88
3.5.1.1.4.5 Ficha de paciente	88
3.5.1.1.4.6 Histórico de Visitas	90
3.5.1.1.4.7 Histórico de problemas	90
3.5.1.1.4.8 Histórico de Informes	91
3.5.1.1.4.9 Agenda del paciente	91
3.5.1.1.5 Información relevante sobre tipos de informes médicos que se generan con la aplicación	92
3.5.1.1.6 Bases de Datos Caché	92
3.5.1.1.6.1 Tablas de CardioWeb (c2hn.card.InfCardioFCK)	92
3.5.1.1.6.2 Tablas de <i>CardioEstim</i> (Package: c2hn.CardioInf)	95
3.5.2 Tarea PSI 3.2: Valoración de Antecedentes	97
3.5.2.1 Requisitos generales	97
3.5.2.2 Catálogo de normas del PSI	97
3.6 Identificación de requisitos	99
3.6.1 Tarea PSI 4.1: Estudio de los Procesos del PSI	99
3.6.1.1 Modelado de procesos de la organización	100
3.6.1.2 Unidades Organizativas	101
3.6.2 Tarea PSI 4.2: Análisis de las Necesidades de Información	106
3.6.2.1 Necesidades de información	107
3.6.2.2 Modelo Entidad / Relación extendido	107
3.6.2.2.1 Clase	107

3.6.2.2.2 Relaciones entre Clases	108
3.6.2.2.3 Diagrama de clases	110
3.6.3 Tarea PSI 4.3: Catalogación de Requisitos	111
3.6.3.1 Requisitos de los procesos afectados por el PSI	111
3.6.3.2 Tipos de requisitos	112
3.6.3.3 Catálogo de requisitos	112
3.6.3.3.1 Persistencia de los datos de paciente de cardiología	112
3.6.3.3.2 Cabecera de la página de información de paciente	112
3.6.3.3.3 Navegación de la aplicación	113
3.6.3.3.4 Sección de ficha de paciente	113
3.6.3.3.5 Sección de pruebas médicas que se realizan al paciente	115
3.6.3.3.6 Sección de sistemas	120
3.6.3.3.7 Sección de visitas	122
3.6.3.3.8 Acceso a la aplicación	125
3.6.3.3.9 Roles de usuario	125
3.6.3.3.10 Auditoría de los datos	125
3.7 Evaluación de riesgos	126
3.7.1 Identificación de los riesgos	126
3.7.2 Estimación del riesgo	127
3.7.3 Planificación de los riesgos	128
3.7.4 Supervisión de los riegos	129
3.8 Definición de la arquitectura tecnológica	130
4. Tecnologías empleadas.	
4.1 Bases de datos relacionales	133
4.2 Gestión de bases de datos con <i>PostgreSQL</i>	136
4.3 <i>Hibernate</i> : persistencia de los datos	138
4.3.1 Arquitectura de <i>Hibernate</i>	140
4.3.2 Ventajas de usar <i>Hibernate</i> como herramienta de persistencia	142
4.4 Eclipse como herramienta IDE	143
4.4.1 Arquitectura de <i>Eclipse</i>	144
4.4.1.1 <i>Workspace</i>	145
4.4.1.2 <i>Workbench</i>	145
4.4.2 Ventajas del uso de esta herramienta	146
4.4.2.1 Editor de texto para diferentes lenguajes	146
4.4.2.2 Barras de Menús y herramientas ajustadas a los lenguajes de programación utilizados	147

4.4.2.3 Explorador de proyectos	147
4.4.2.4 Servidores Web <i>Tomcat</i>	148
4.5 <i>Tomcat</i> : servidor para aplicaciones	149
4.5.1 Arquitectura del servidor <i>Tomcat</i>	149
4.6 <i>Java Databases Connectivity</i> (JDBC): API para acceder al sistema de gestión de BB.DD	153
4.6.1 Arquitectura JDBC	153
4.7 <i>Java Server Pages</i>	154
4.7.1 Arquitectura JSP	155
4.7.1.1 Filtros	156
4.7.1.2 <i>Servlets</i>	156
4.7.1.3 Páginas JSP	156
4.7.1.4 Servidor	157
4.7.2 Ventajas	157
4.7.2.1 Portabilidad	157
4.7.2.2 Soporte de herramientas de alta calidad	158
4.7.2.3 Separación de roles	158
4.7.2.4 Reutilización de componentes y bibliotecas de etiquetas	158
4.7.2.4.1 Variables y objetos implícitos	158
4.7.2.4.2 Directivas	159
4.7.2.4.3 Declaraciones	160
4.7.2.4.4 Scriptlets	160
4.7.2.4.5 Expresiones	161
4.7.2.4.6 Etiquetas JSP	162
4.7.2.4.7 Soporte para acciones, expresiones y secuencias de comandos	164
4.8 El concepto de componente de negocio	165
4.8.1 Desarrollo dirigido por el modelo ligero	165
4.8.2 Componente de negocio	167
4.9 <i>Openxava</i> : marco de trabajo	170
4.9.1 Elementos <i>Openxava</i>	171
4.9.1.1 Componentes de negocio	171
4.9.1.2 Módulos	173
4.9.1.3 Controladores	173
4.9.1.4 Editores	173
4.9.1.5 Validadores	174
4.9.1.6 Calculadores	174
4.9.2 Ventajas de usar el marco de trabajo <i>Openxava</i>	174

4.10 Java: clases, objetos, herencias	175
4.11 <i>Ajax: Asynchronous JavaScript And XML</i>	176
4.11.1 Funcionamiento del ciclo <i>Ajax</i>	177
4.11.2 Ventajas	178
4.12 Nivel de presentación de los datos	179
4.12.1 HTML5	179
4.12.2 CSS3	181
4.13 Uso de <i>Ireport</i> y librerías JasperReport para generar informes pdf	182
4.14 El estudio de la experiencia de usuario como valor añadido	182

5. Aplicación Cardiología.

5.1 Configuración de las tecnologías empleadas	187
5.1.1 Gestión de la Base de Datos con PostgreSQL	187
5.1.2 Configuración de <i>Hibernate</i>	188
5.1.3 Configuración de la herramienta IDE	189
5.1.4 Configuración del <i>Tomcat</i> , el servidor para aplicaciones	190
5.1.5 Configuración del servidor Tomcat para incluir el JDBC para postgresQL	192
5.1.6 Configuración para acceder al sistema de gestión de la BB.DD	193
5.1.7 Configuración para el marco de trabajo <i>Openxava</i>	194
5.1.7.1 Componentes de negocio	194
5.1.7.2 Módulos	194
5.1.7.3 Controladores	196
5.1.7.4 Editores	197
5.1.7.5 Validadores	200
5.1.7.6 Calculadores	202
5.1.8 Uso de las anotaciones JPA, <i>Openxava</i> y API JPA	203
5.1.8.1 Anotaciones JPA	203
5.1.8.1.1 Anotación para definir las clases del modelo de datos como Entidad	203
5.1.8.1.2 Propiedades de las clases de la aplicación web	204
5.1.8.1.3 Clave primaria de la clase	205
5.1.8.1.4 Especificación de la columna de la tabla	205
5.1.8.1.5 Referencias	206
5.1.8.1.6 Clases embebidas	208
5.1.8.1.7 Clases transitorias	210
5.1.8.2 API JPA	211
5.1.8.3 Anotaciones <i>Openxava</i> para las vistas en la interfaz de usuario	213
5.1.9 Configuración de AJAX	214

5.1.10 Modificación del nivel de presentación de los datos de <i>Openxava</i>	215
5.1.11 Configuración para el uso de la interfaz gráfica <i>Ireport</i>	216
5.1.11.1 Clase informe	217
5.1.11.2 Obtener datos de las vistas	217
5.1.11.3 Obtener datos de la BB.DD	220
5.1.11.3.1 Crear la conexión con la BB.DD a través del JDBC	220
5.1.11.3.2 Crear los campos	221
5.2 Persistencia de los datos de paciente de cardiología	223
5.3 Cabecera de la página de información de paciente	237
5.3.1 Visualización de los datos de paciente desde la BB.DD de Drago	243
5.4 Navegación de la aplicación	243
5.5 Sección de Ficha de paciente	249
5.5.1 Clase <i>Identificable.java</i>	249
5.5.2 Antecedentes	249
5.5.3 Alergias	253
5.5.4 Indicaciones	256
5.5.5 Procedimientos	259
5.5.6 Ingresos	263
5.5.7 Outcomes	266
5.5.8 Comentario	271
5.6 Generación automática de informes de historiales	273
5.7 Sección Pruebas médicas	275
5.7.1 Cateterismo	275
5.7.2 Ecocardiografías	280
5.7.2.1. Carga de plantillas para la prueba de ecocardiografía	285
5.7.3. Ecocardiografía Transtorácica	287
5.7.3.1 Estereotipos necesarios para la clase	293
5.7.4 Ergometrías	297
5.7.4.1. Estereotipos necesarios para la clase	303
5.7.5 Estudios electrofisiológicos	307
5.7.6 Hipertensión pulmonar	313
5.7.6.1. Estereotipos para la clase	320
5.7.6.2 Internacionalización para la clase	322
5.7.7 Holter´s	323
5.7.8. Resincronización cardíaca	329
5.8 Sección de sistemas	335
5.8.1 Generadores	335

5.8.2 Electrodos	353
5.8.2.1 Estereotipos para la clase	359
5.8.3 Problemas activos	362
5.9 Sección de visitas	367
5.10 Roles de usuario y acceso a la aplicación	399
5.11 Auditoría de los datos	403
5.12 Mayor usabilidad para la aplicación de cardiología del HUGCDN	405
5.12.1 No salir del objeto al grabar	405
5.12.2 Minimizar apartados	406
5.13 Búsqueda de pacientes y gráficas de información relevante	409
5.13.1 Botón Nuevo	409
5.13.2 Botón Generar PDF	409
5.13.3 Botón Generar Excel	410
5.13.4 Botón datos	410
5.13.5 Botón gráficas	411
5.13.5.1 Representación en barras	411
5.13.5.2 Representación lineal	412
5.13.5.3 Representación en porcentajes	412
5.13.6 Botón detalle	413
5.13.7 Botón lista	413
5.13.8 Botón ambos	414
5.13.9 Filtros	415
6. Test de la aplicación.	
6.1 Acceso a la aplicación	421
6.2 Búsqueda de paciente	421
6.3 Funcionalidad de la cabecera de la aplicación con información del paciente	421
6.4 Funcionalidad del módulo Ficha de paciente	422
6.4.1 Alergias del paciente	422
6.4.2 Antecedentes del paciente	423
6.4.3 Indicaciones del paciente	423
6.4.4 Procedimientos del paciente	424
6.4.5 Ingresos del paciente	425
6.4.6 <i>Outcomes</i> del paciente	426
6.4.7 Comentario del paciente	426
6.5 Funcionalidad del módulo Pruebas médicas	427
6.6 Funcionalidad del módulo Sistemas	428

6.7 Funcionalidad del módulo Visitas	429
6.8 Auditoría de los datos	430
7. Conclusiones.	
7.1 Introducción	433
7.2 Conclusiones	433
7.3 Líneas futuras	433
8. Presupuesto.	
8.1 Introducción	437
8.2 Coste de los recursos humanos	437
8.3 Coste de los recursos <i>Hardware</i>	438
8.4 Coste de los recursos <i>Software</i>	439
8.5 Coste de la redacción.....	440
8.6 Coste total	440
Bibliografía.	
Bibliografía	441

Índice de figuras.

Figura 1: Ilustración de las herramientas informáticas de trabajo de la especialidad de Cardiología	36
Figura 2: Representación del modelo arquitectónico cliente-servidor	46
Figura 3: Modelo 7 capas OSI, modelo 4 capas TCP/IP	47
Figura 4: Gráfica de la iteración en espiral	49
Figura 5: Gráfica de método en cascada	50
Figura 6: Ciclo del desarrollo ágil de aplicaciones	51
Figura 7: Ejemplo de Sistema de la información	57
Figura 8: Productos de entrada y salida de la Tarea PSI 1.1	62
Figura 9: Productos de entrada y salida de la Tarea PSI 1.2	63
Figura 10: Productos de entrada y salida de la Tarea PSI 1.3	65
Figura 11: Productos de entrada y salida de la Tarea PSI 2.1.	67
Figura 12: Gráfica de cumplimiento de objetivos	67
Figura 13: Productos de entrada y salida de la Tarea PSI 2.2	69
Figura 14: Estructura organizativa	69
Figura 15: Equipos de trabajo interactuando	71
Figura 16: Productos de entrada y salida de la Tarea PSI 2.3	71
Figura 17: Productos de entrada y salida de la Tarea PSI 2.4	73
Figura 18: Productos de entrada y salida de la Tarea PSI 3.1	74
Figura 19: Electrodo ventricular de fijación pasiva	85
Figura 20: Productos de entrada y salida de la Tarea PSI 3.2	97
Figura 21: Productos de entrada y salida de la Tarea PSI 4.1	100
Figura 22: Modelo de procesos de la organización	101
Figura 23: Ejemplo sobre Active Directory de Windows Server 2008	102
Figura 24: Acceso remoto, autenticación en el dominio Canaria Salud, y acceso a las aplicaciones de la Intranet <i>Cardioweb</i> y <i>Cardioestim</i>	104
Figura 25: Gráfico de agrupación de objetos que tienen mismos permisos, mismas políticas de seguridad, etc	106
Figura 26: Productos de entrada y salida de la Tarea PSI 4.2	107
Figura 27: Representación de una clase	107
Figura 28: Productos de entrada y salida de la Tarea PSI 4.3	111
Figura 29: Valoración de impacto	128
Figura 30: Flujo de datos en una aplicación web	133
Figura 31: Componentes más importantes en un sistema <i>PostgreSQL</i>	137
Figura 32: Algunas de las anotaciones que son utilizables desde <i>Openxava</i>	139

Figura 33: Validaciones predefinidas de Openxava	139
Figura 34: Objetos persistentes que sincronizan los datos entre la aplicación y la base de datos	141
Figura 35: Arquitectura de Hibernate en nuestra aplicación de cardiología	142
Figura 36: Representación del entorno de desarrollo	143
Figura 37: Representación de la estructura de Eclipse	144
Figura 38: Perspectiva para el desarrollo de la aplicación de cardiología	146
Figura 39: Representación del modelo a tres niveles del JDBC	154
Figura 40: Arquitectura del <i>Model 2 de JavaServer Pages</i>	155
Figura 41: Filtros que aparecen en el nivel servidor	156
Figura 42: Representación del modelo web cliente/servidor	157
Figura 43: Objetos y variables privilegiadas ya incluidas listas para usar	158
Figura 44: Directivas disponibles para JSP	159
Figura 45: Tipos de etiquetas JSP	163
Figura 46: Desarrollo dirigido por el modelo	166
Figura 47: Comparación <i>Openxava</i> y MDD	166
Figura 48: Esquema simple de cómo se desarrolla la aplicación de cardiología	167
Figura 49: Modelo Vista Controlador (MVC)	168
Figura 50: Componente de negocio	169
Figura 51: Representación de algunos elementos de la aplicación de cardiología	171
Figura 52: Comparativa de una aplicación web clásica y una aplicación web con tecnología <i>Ajax</i>	177
Figura 53: Representación del ciclo <i>Ajax</i> en la aplicación de cardiología	178
Figura 54: Especificaciones de los métodos <i>setProperty</i> , <i>getProperty</i> de la interfaz <i>EntityManager</i> [102]	212
Figura 55: Estructura de navegación de la aplicación web	245

Índice de capturas de pantalla.

Captura 1: Captura de pantalla de Sección “Sistemas” del Software <i>Cardioestim</i>	36
Captura 2: Captura de pantalla de Sección “Ficha del paciente” del Software <i>Cardioestim</i>	37
Captura 3: Captura de pantalla de Sección “Visita” del Software <i>Cardioestim</i>	37
Captura 4: Captura de pantalla de Sección “Consultar informe” del Software <i>Cardioestim</i>	38
Captura 5: Captura de pantalla del generador de informes de <i>Cardioweb</i>	38
Captura 6: Captura de pantalla de registro de parámetros de dispositivos de <i>Cardioweb</i>	39
Captura 7: Fragmento de la portada de página de AgileManifesto.org	50
Captura 8: Parámetros de entrada para la prueba cardiológica Hipertensión Pulmonar	76
Captura 9: Parámetros de entrada para la prueba cardiológica Resincronización Cardíaca	77
Captura 10: Parámetros de entrada para la prueba cardiológica Ecocardiografía Transtorácica	78
Captura 11: Parámetros de entrada para la prueba cardiológica Ergometría	79
Captura 12: Herramienta para elaborar el informe de la prueba cardiológica Holter	80
Captura 13: Herramienta para elaborar el informe de la prueba cardiológica Cateterismo	80
Captura 14: Herramienta para elaborar el informe de la prueba cardiológica Estudio Electrofisiológico	82
Captura 15: Captura de pantalla del listado de dispositivos implantados en el paciente de prueba	82
Captura 16: Captura de pantalla de Sección “Nuevo dispositivo” del Software <i>CardioEstim</i>	83
Captura 17: Captura de pantalla de parámetros del Generador registrado en el paciente del Software <i>CardioEstim</i>	83
Captura 18: Captura de pantalla de lista de electrodos en la Sección “Sistemas” del Software	85
Captura 19: Captura de pantalla de los parámetros de electrodo en Sección “Sistemas” del Software	85
Captura 20: Captura de pantalla de “Nueva visita” en la Sección “Visita” de la aplicación <i>CardioEstim</i>	86
Captura 21: Captura de pantalla de formulario de “Evento” en “Visita”	86

Captura 22: Captura de pantalla de “Programación Bradicardia” dentro del menú “Visita”	87
Captura 23: Captura de pantalla de “Programación Taquicardia” dentro del menú “Visita”	87
Captura 24: Captura de pantalla de “Nuevo problema” dentro del menú “Visita”	88
Captura 25: Captura de pantalla de pestaña “Ficha Paciente”	88
Captura 26: Captura de pantalla de la sección “Ficha de Paciente”	89
Captura 27: Captura de pantalla de la sección “Histórico de visitas”	90
Captura 28: Captura de pantalla de la sección “Histórico de problemas”	91
Captura 29: Captura de pantalla de la Sección “Consultar Informe”	91
Captura 30: Captura de pantalla del paciente	91
Captura 31: Captura de pantalla Acceso a la Intranet del Hospital	103
Captura 32: Captura de pantalla Acceso a los servicios de la Intranet del Hospital	103
Captura 33: Captura de los servicios activados para la alumna de la Intranet del Hospital	104
Captura 34: Captura de pantalla de la pasarela de autenticación del Gobierno de Canarias	105
Captura 35: Fragmento del diagrama UML donde se refleja que un paciente tiene 0, o muchas visitas médicas al servicio de Cardiología	108
Captura 36: Fragmento del diagrama UML donde se aprecia que la clase <i>Visita</i> tiene Diagnóstico	109
Captura 37: Fragmento del diagrama UML donde se representa que <i>tipoVisita</i> es enumerado	110
Captura 38: Ejemplo de los registros de la tabla <i>visita</i> de la BB.DD. de pruebas de cardiología	134
Captura 39: Datos de la tabla <i>diagnostico</i> de la BB.DD de prueba	135
Captura 40: Fragmento de diagrama UML donde se aprecia la relación 1 a 1 entre las clases <i>Visita</i> y <i>Diagnostico</i>	135
Captura 41: Captura de pantalla de las propiedades de la tabla <i>visita</i>	136
Captura 42: Fragmento de código desde la línea 61 a 77 de la clase <i>Paciente.java</i>	140
Captura 43: Captura de pantalla de la sección datos demográficos del paciente de la aplicación web	140
Captura 44: Página oficial de descarga para los diferentes IDE’s de escritorio Eclipse	144
Captura 45: Contenido de la distribución de <i>Openxava</i>	145

Captura 46: Sección de editor de texto de la perspectiva anterior representada	146
Captura 47: Barra de menú en <i>Eclipse</i>	147
Captura 48: Captura de pantalla del explorador de proyectos de <i>Eclipse</i>	148
Captura 49: Sección del servidor <i>Tomcat</i> incorporado para el despliegue del servidor web de desarrollo TOMCAT	149
Captura 50: Captura de pantalla de los principales directorios Tomcat	150
Captura 51: Captura de pantalla del contenido del directorio <i>bin</i>	151
Captura 52: Captura de pantalla del contenido del directorio <i>conf</i>	151
Captura 53: Captura de pantalla del contenido del directorio <i>lib</i>	152
Captura 54: Captura de pantalla del contenido del directorio <i>logs</i>	152
Captura 55: Captura de pantalla del contenido del directorio <i>webapps</i>	153
Captura 56: Captura de pantalla de código del fichero <i>index.jsp</i>	159
Captura 57: Captura de pantalla de la página que representa el código anterior	160
Captura 58: Captura de pantalla del fichero <i>index.jsp</i> con <i>scriptlets</i>	161
Captura 59: Expresiones JSP	162
Captura 60: Captura de pantalla de fragmento de código <i>index.jsp</i>	163
Captura 61: Captura de pantalla del directorio naviox donde figura el fichero que se está incluyendo	164
Captura 62: Inspeccionando elemento con el navegador web	164
Captura 63: Fragmento de código de la clase java Paciente	170
Captura 64: Barra de módulos que se generan automáticamente en <i>Openxava</i>	173
Captura 65: Fragmento de código del código fuente de la aplicación web	180
Captura 66: div que se muestra gracias a la línea 3 del código anterior	180
Captura 67: Captura de pantalla de la página web para <i>ecocardiograma</i> de la aplicación web	181
Captura 68: Captura de pantalla de fragmento de código de <i>cabecera.jsp</i>	182
Captura 69: Creación de la nueva base de datos <i>cardiología</i>	187
Captura 70: Captura de pantalla del contenido del fichero de configuración predefinido <i>hibernate.cfg.xml</i>	188
Captura 70: Capturas de pantalla del contenido del fichero de configuración predefinido <i>persistence.xml</i>	189
Captura 71: Ventana de inicio de Eclipse para seleccionar el <i>workspace</i> deseado	190
Captura 72: Características del servidor <i>Tomcat</i> añadido en <i>Eclipse</i> como servidor de ejecución	190
Captura 73: Captura de pantalla de las aplicaciones incluidas en el servidor	191
Captura 74: Captura de pantalla de la sección servidor del IDE Eclipse	191

Captura 75: Captura de pantalla de la aplicación que detecta los procesos para “matarlos”	192
Captura 76: Fragmento de código del fichero de configuración del servidor <i>context.xml</i>	192
Captura 77: Captura de pantalla de fragmento de código del archivo <i>web.xml</i>	193
Captura 78: Captura de pantalla de fragmento de código del fichero <i>mainNavigator.jsp</i>	195
Captura 79: Captura de pantalla de un fragmento de código del script <i>controladores.xml</i>	196
Captura 80: Captura de pantalla de la sección Pruebas médicas del paciente de la aplicación web	196
Captura 81: Captura de pantalla donde se muestra el fragmento de código del archivo <i>editores.xml</i>	197
Captura 82: Captura de pantalla del explorador de paquetes de Eclipse	198
Captura 83: Captura de pantalla del contenido de la clase <i>PesoFormateador.java</i>	198
Captura 84: Contenido del script <i>pesoEditor.jsp</i>	199
Captura 85: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	199
Captura 86: Captura de pantalla del estado nutricional de la ficha de paciente	200
Captura 87: Captura de pantalla de fragmento de código de la clase <i>Generador.java</i>	201
Captura 88: Captura de pantalla de la excepción lanzada al comprobar que los médicos son la misma persona	201
Captura 89: Captura de pantalla del código del archivo <i>MensajesCardiologia_es.propiedades</i>	202
Captura 90: Captura de pantalla de fragmento de código de la clase <i>Generador.java</i>	202
Captura 91: Captura de pantalla de un fragmento de la ficha <i>Generador</i> del paciente	202
Captura 92: Fragmento de código de la clase <i>Paciente.java</i>	203
Captura 93: Fragmento de código de la clase <i>Paciente.java</i>	203
Captura 94: Captura de pantalla de una propiedad requerida de la clase <i>Paciente.java</i>	204
Captura 95: Captura de pantalla de los métodos de propiedad <i>nombre</i> requerida de la clase <i>Paciente.java</i>	204
Captura 96: Captura de pantalla del código de la clase <i>Identificable.java</i>	205

Captura 97: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	206
Captura 98: Captura de pantalla del valor IMC	206
Captura 99: Captura de pantalla de fragmento de código de la clase <i>Generador.java</i>	207
Captura 100: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	207
Captura 101: Captura de pantalla de la aplicación web, de la sección Sistemas, ficha generadores del paciente	208
Captura 102: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	209
Captura 103: Captura de pantalla del código de la clase <i>Comentario.java</i>	209
Captura 104: Captura de pantalla de la aplicación web, de la sección Ficha de paciente, observaciones generales del paciente ..	210
Captura 105: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	210
Captura 106: Captura de pantalla de la aplicación web, de la sección Sistemas, ficha de problemas activos del paciente	211
Captura 107: Captura de pantalla de la llamada a la acción de lectura de los detalles del problema activo	211
Captura 108: Captura de pantalla de fragmento de código de los métodos de la clase <i>Paciente.java</i>	212
Captura 109: Captura de pantalla de la interfaz de usuario de la aplicación web	213
Captura 110: Captura de pantalla de un fragmento de código de la clase <i>Paciente.java</i>	214
Captura 111: Captura de pantalla de un fragmento de código de la clase <i>Paciente.java</i>	214
Captura 112: Captura de pantalla de la sección Pruebas médicas	214
Captura 113: Captura de pantalla de fragmento de código de <i>cabecera.jsp</i>	215
Captura 114: Captura de pantalla de la clase <i>InformePrueba.java</i> donde se incluyen las librerías <i>org.openxava.view.*</i>	218
Captura 115: Captura de pantalla de la clase <i>InformePrueba.java</i> donde se recogen los valores de las vistas	218
Captura 116: Captura de pantalla del atributo <i>view</i> de la clase <i>ViewBaseAction.java</i> ...	219
Captura 117: Captura de pantalla de código de clase <i>InformePrueba.java</i>	219
Captura 118: Captura de pantalla de código de clase <i>InformePrueba.java</i> donde se obtiene la plantilla generada con <i>Ireport</i>	220

Captura 119: Captura de pantalla de herramienta Ireport para crear conexión con la BB.DD	220
Captura 120: Código SQL incluido con <i>Ireport</i> para acceder a las columnas de la tabla deseada de la BB.DD	221
Captura 121: Captura de pantalla de la plantilla <i>ejemploIreport2.jrxml</i>	222
Captura 122: Captura de pantalla del informe médico visualizado a través del navegador web Internet Explorer	222
Captura 123: Captura de pantalla de la cabecera de la página de paciente	237
Captura 124: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	237
Capturas 125: Capturas de pantalla de fragmento de código de la clase <i>Paciente.java</i>	238
Capturas 126: Capturas de pantalla de fragmento de código de la clase <i>Paciente.java</i>	239
Captura 127: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	240
Captura 128: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	240
Captura 129: Captura de pantalla de fragmento de código del fichero <i>editores.xml</i>	241
Captura 130: Captura de pantalla del código de la clase <i>PesoFormateador.java</i>	241
Captura 131: Captura de pantalla del script <i>metrosEditor.jsp</i>	242
Captura 132: Captura de pantalla del script <i>pesoEditor.jsp</i>	242
Captura 133: Menú principal con acciones CRUD y paginado de pacientes	242
Captura 134: Captura de pantalla de la búsqueda de pacientes de la aplicación web de cardiología	244
Capturas 135: Capturas de pantalla de fragmento de código de la clase <i>Paciente.java</i>	246
Captura 136: Captura de pantalla de fragmento de código del fichero <i>EtiquetasCardiologias_es.properties</i>	247
Captura 137: Captura de pantalla del código de la clase <i>Identificable.java</i>	249
Captura 138: Captura de pantalla de la sección Ficha de paciente, apartado Antecedentes	250
Captura 139: Captura de pantalla del formulario crear antecedente	251
Captura 140: Captura de pantalla del formulario editar antecedente	251
Capturas 141: Capturas de pantalla de la clase <i>Antecedente.java</i>	253
Captura 142: Captura de pantalla de la clase <i>Paciente.java</i>	253

Captura 143: Captura de pantalla de la sección Ficha de paciente, apartado Alergias	254
Captura 144: Captura de pantalla del formulario crear alergia	254
Captura 145: Captura de pantalla del formulario editar alergia	255
Capturas 146: Capturas de pantalla de la clase <i>AlergiaNueva.java</i>	255
Captura 147: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	256
Captura 148: Captura de pantalla de la sección Ficha de paciente, apartado Indicaciones	256
Captura 149: Captura de pantalla del formulario crear indicación	257
Captura 150: Captura de pantalla del formulario editar indicación	257
Capturas 151: Capturas de pantalla del código de la clase <i>Indicacion.java</i>	258
Captura 152: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	258
Captura 153: Captura de pantalla de la sección Ficha de paciente, apartado Procedimientos	259
Captura 154: Captura de pantalla del formulario crear procedimiento	260
Captura 155: Captura de pantalla del formulario editar procedimiento	261
Capturas 156: Capturas de pantalla del código de la clase <i>Procedimiento.java</i>	262
Captura 157: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	263
Captura 158: Captura de pantalla de la sección Ficha de paciente, apartado Ingresos	263
Captura 159: Captura de pantalla del formulario crear ingreso	264
Captura 160: Captura de pantalla del formulario editar ingreso	264
Capturas 161: Capturas de pantalla del código de la clase <i>Ingresos.java</i>	266
Captura 162: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	266
Captura 163: Captura de pantalla de la sección Ficha de paciente, apartado <i>Outcomes</i>	267
Capturas 164: Capturas de pantalla de fragmentos de código de la clase <i>Paciente.java</i>	271
Captura 165: Captura de pantalla de sección Paciente, apartado Comentario	272
Captura 166: Captura de pantalla del código de la clase <i>Comentario.java</i>	272
Captura 167: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	273
Captura 168: Botones para exportación de historiales	273

Captura 169: Captura de pantalla del archivo <i>pdf</i> que contiene del historial de antecedentes exportado	274
Captura 170: Captura de pantalla del archivo <i>.csv</i> que contiene del historial de antecedentes exportado	274
Captura 171: Captura de pantalla de la sección Pruebas médicas, apartado Cateterismos	276
Captura 172: Captura de pantalla del formulario crear cateterismo	276
Captura 173: Captura de pantalla del formulario editar cateterismo	277
Captura 174: Capturas de pantalla del código de la clase <i>Cateterismo.java</i>	279
Captura 175: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	280
Captura 176: Captura de pantalla de la sección Pruebas médicas, apartado ecocardiografías	280
Captura 177: Captura de pantalla del formulario crear ecocardiografía	281
Captura 178: Captura de pantalla del formulario editar ecocardiografía	282
Captura 179: Captura de pantalla de la clase <i>Ecocardiografia.java</i>	284
Captura 180: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	284
Captura 181: Captura de pantalla del código de la clase <i>CargarPlantilla.java</i>	285
Captura 182: Captura de pantalla de fragmento de código de <i>EtiquetasCardiologia_es.properties</i>	285
Captura 183: Captura de pantalla de la sección Pruebas médicas, apartado Ecocardiografías Transtorácicas	287
Captura 184: Captura de pantalla del formulario crear Ecocardiografía transtorácica	288
Captura 185: Captura de pantalla del formulario editar Ecocardiografía transtorácica	289
Capturas 186: Capturas de pantalla del código de la clase <i>EcocardiografiaTranstoracica.java</i>	293
Captura 187: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	293
Captura 188: Captura de pantalla de fragmento de código del script <i>editores.xml</i>	293
Captura 189: Captura de pantalla de fragmento de la clase Formateadora <i>PesoFormateador</i>	294
Captura 190: Captura de pantalla del script <i>milímetrosEditor.jsp</i>	294
Captura 191: Captura de pantalla del script <i>porcentajeEditor.jsp</i>	294

Captura 192: Captura de pantalla del script <i>simpsonEditor.jsp</i>	295
Captura 193: Captura de pantalla de la sección Pruebas médicas, apartado Ergometrías	297
Captura 194: Captura de pantalla del formulario crear Ergometría	298
Captura 195: Captura de pantalla del formulario editar Ergometría	299
Captura 196: Capturas de pantalla del código de la clase <i>Ergometria.java</i>	303
Captura 197: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	303
Captura 198: Captura de pantalla de fragmento de código del script <i>editores.xml</i>	303
Captura 199: Captura de pantalla de la clase formateadora <i>PesoFormateador</i>	304
Captura 200: Captura de pantalla del script <i>lpmEditor.jsp</i>	304
Captura 201: Captura de pantalla del script <i>mmhgEditor.jsp</i>	305
Captura 202: Captura de pantalla del script <i>minutosEditor.jsp</i>	305
Captura 203: Captura de pantalla del script <i>segundosEditor.jsp</i>	305
Captura 204: Captura de pantalla de fragmento del script <i>EtiquetasCardiologias_es.properties</i>	306
Captura 205: Captura de pantalla de la sección Pruebas médicas, apartado Estudios electrofisiológicos	307
Captura 206: Captura de pantalla del formulario crear EEF	308
Captura 207: Captura de pantalla del formulario editar EEF	309
Capturas 208: Capturas de pantalla del código de la clase <i>EstudioElectrofisiologico.java</i>	310
Captura 209: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i> ...	311
Captura 210: Captura de pantalla de fragmento del script <i>EtiquetasCardiologias_es.properties</i>	311
Captura 211: Captura de pantalla de la sección Pruebas médicas, apartado Hipertensión pulmonar	313
Captura 212: Captura de pantalla del formulario crear HTP	314
Captura 213: Captura de pantalla del formulario editar HTP	315
Capturas 214: Capturas de pantalla del código de la clase <i>HipertensionPulmonar.java</i>	320
Captura 215: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	320
Captura 216: Captura de pantalla de fragmento de código del script <i>editores.xml</i>	321
Captura 217: Captura de pantalla de la clase formateadora <i>PesoFormateador</i>	321

Captura 218: Captura de pantalla del script <i>cm2Editor.jsp</i>	321
Captura 219: Captura de pantalla de fragmento del script <i>EtiquetasCardiologias_es.properties</i>	322
Captura 220: Captura de pantalla de la sección Pruebas médicas, apartado <i>Holter's</i>	323
Captura 221: Captura de pantalla del formulario crear <i>Holter</i>	324
Captura 222: Captura de pantalla del formulario editar <i>Holter</i>	325
Captura 223: Capturas de pantalla del código de la clase <i>Holter.java</i>	326
Captura 224: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i> ...	327
Captura 225: Captura de pantalla de la sección Pruebas médicas, apartado Resincronización cardíaca	329
Captura 226: Captura de pantalla del formulario crear Resincronización cardíaca	329
Captura 227: Captura de pantalla del formulario editar Resincronización cardíaca	330
Capturas 228: Capturas de pantalla del código de la clase <i>ResincronizacionCardiaca.java</i>	333
Captura 229: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	334
Captura 230: Captura de pantalla de la sección Sistemas, apartado Generadores	336
Captura 231: Captura de pantalla del formulario crear generador	337
Captura 232: Captura de pantalla del formulario editar generador	338
Capturas 233: Capturas de pantalla del código de la clase <i>Generador.java</i>	344
Capturas 234: Capturas de pantalla de la clase <i>TipoComplicacion.java</i>	347
Captura 235: Captura de pantalla del formulario añadir problema	347
Captura 236: Captura de pantalla del formulario editar problema	348
Capturas 237: Capturas de pantalla del código de la clase <i>Problema.java</i>	351
Captura 238: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	352
Capturas 239: Captura de pantalla de fragmento del script <i>EtiquetasCardiologias_es.properties</i>	353
Captura 240: Captura de pantalla de la sección Sistemas, apartado Electrodo	353
Captura 241: Captura de pantalla del formulario crear electrodo	353
Captura 242: Captura de pantalla del formulario editar electrodo	354
Capturas 243: Capturas de pantalla del código de la clase <i>Electrodo.java</i>	359
Captura 244: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	359

Captura 245: Captura de pantalla de fragmento de código del script <i>editores.xml</i>	360
Captura 246: Captura de pantalla de la clase formateadora <i>PesoFormateador</i>	360
Captura 247: Captura de pantalla del script <i>mvms.jsp</i>	361
Captura 248: Captura de pantalla del script <i>ohmios.jsp</i>	361
Captura 249: Captura de pantalla del script <i>mv.jsp</i>	361
Captura 250: Captura de pantalla de fragmento del script <i>EtiquetasCardiologias_es.properties</i>	362
Captura 251: Captura de pantalla de la Sección Sistemas, apartado Problemas activos	362
Captura 252: Captura de pantalla del formulario ver Problemas activos	363
Captura 253: Captura de pantalla de la acción <i>CalcularPacienteEnProblema</i>	364
Captura 254: Captura de pantalla de fragmento de código del script <i>controladores.xml</i>	364
Captura 255: Capturas de pantalla de fragmentos de código de la clase <i>Problema.java</i>	365
Captura 256: Fragmento de código de las vistas de la clase <i>Paciente.java</i>	365
Captura 257: Fragmento de código de la colección calculada de la clase <i>Paciente.java</i>	365
Captura 258: Captura de pantalla de la sección Visitas	367
Captura 259: Captura de pantalla del formulario crear visita	368
Captura 260: Captura de pantalla del formulario editar visita	370
Captura 261: Capturas de pantalla del código de la clase <i>Visita.java</i>	372
Captura 262: Captura de pantalla de fragmento de código de la clase <i>Paciente.java</i>	373
Capturas 263: Capturas de pantalla del código de la clase <i>Diagnostico.java</i>	378
Captura 264: Captura de pantalla del formulario Evento médico	379
Capturas 267: Capturas de pantalla del código de la clase <i>EventoMedico.java</i>	383
Captura 268: Captura de pantalla del formulario Programación Bradicardia	384
Capturas 269: Capturas de pantalla del código de la clase <i>ProgramacionBradicardia.java</i>	388
Captura 270: Captura de pantalla del formulario Programación Taquicardia	389
Capturas 271: Capturas de pantalla del código de la clase <i>ProgramacionTaquicardia.java</i>	396
Captura 272: Captura de pantalla de fragmento de código del script <i>editores.xml</i>	396
Captura 273: Captura de pantalla de la clase <i>PesoFormateador</i>	396
Captura 274: Captura de pantalla del script <i>voltiosEditor.jsp</i>	397

Captura 275: Captura de pantalla del script <i>vmsEditor.jsp</i>	397
Capturas 276: Capturas de pantalla de fragmento del script <i>EtiquetasCardiologias_es.properties</i>	398
Captura 277: Captura de pantalla del contenido del archivo <i>MensajesCardiologia_es.properties</i>	399
Captura 278: Captura de pantalla de la página de acceso a la aplicación de cardiología del HUGCDN	399
Captura 279: Captura de pantalla del contenido del archivo en <i>Eclipse</i>	400
Captura 280: Captura de pantalla con usuario y/o contraseña erróneo	400
Captura 281: Captura de pantalla del script <i>controladores.xml</i>	403
Captura 281_2: Captura de pantalla de fragmento de código de <i>Paciente.java</i>	403
Captura 282: Capturas de pantalla de la acción <i>GuardaAlergia.java</i>	404
Captura 283: de pantalla de registros de auditoría de los datos	405
Captura 284: Captura de pantalla de la clase <i>SaveAction.java</i>	406
Captura 285: Captura de pantalla de fragmento de código de la clase <i>Diagnostico.java</i>	407
Captura 286: Captura de pantalla de la página de inicio de la aplicación web de cardiología	409
Captura 287: Captura de pantalla de la página de inicio con filtros de búsqueda	410
Captura 288: Captura de pantalla del archivo PDF que contiene la lista de pacientes deseada	410
Captura 289: Captura de pantalla del archivo <i>excel</i> que contiene la lista de pacientes deseada	410
Captura 290: Captura de pantalla de la gráfica de barras de peso de paciente	411
Captura 291: Captura de pantalla de la gráfica lineal de peso de paciente	412
Captura 292: Captura de pantalla de la gráfica comparativa	412
Captura 293: Captura de pantalla de los valores del desplegable añadir parámetro	413
Captura 294: Captura de pantalla de la página de inicio de la aplicación web de cardiología	414
Captura 295: Captura de pantalla al pulsar botón Ambos	414
Captura 296: Desplegables para filtros de tipo numérico	415
Captura 297: Desplegables para filtros de tipo <i>string</i>	415
Captura 289: Captura de pantalla del filtro peso	416
Captura 290: Captura de pantalla del filtro peso con sus valores	416
Captura 291: Captura de pantalla del listado paciente filtrado	416
Captura 292: Valores del desplegable paginado	417

Índice de tablas.

Tabla 1: Inicio del Plan de Sistemas de Información	61
Tabla 2: Tabla de tareas para la definición y organización del PSI	66
Tabla 3: Tabla de objetivos	68
Tabla 4: Tabla del catálogo de usuarios	70
Tabla 5: Tabla del plan de trabajo	72
Tabla 6: Tareas del estudio de la información relevante y de los sistemas de información actuales	74
Tabla 7: Tareas para la identificación de requisitos	99
Tabla 8: Tabla de riesgos generales	127
Tabla 9: Tabla de riesgos específicos	127
Tabla 10: Tabla de riesgos, probabilidades aproximadas de que suceda, y tipo de riesgo	128
Tabla 11: Tabla de planificación de los riesgos	129
Tabla 12: Supervisión de los riesgos	129
Tabla 13: Tabla <i>alergianueva</i> e información relevante de sus atributos	223
Tabla 14: Tabla <i>antecedente</i> e información relevante de sus atributos	223
Tabla 15: Tabla <i>cateterismo</i> e información relevante de sus atributos	223
Tabla 16: Tabla <i>comentario</i> e información relevante de sus atributos	224
Tabla 17: Tabla <i>complicacionimplante</i> e información relevante de sus atributos	224
Tabla 18: Tabla <i>diagnostico</i> e información relevante de sus atributos	224
Tabla 19: Tabla <i>ecocardiografia</i> e información relevante de sus atributos	225
Tabla 20: Tabla <i>ecocardiografiatranstoracica</i> e información relevante de sus atributos	225
Tabla 21: Tabla <i>electrodo</i> e información relevante de sus atributos	226
Tabla 22: Tabla <i>ergometria</i> e información relevante de sus atributos	226
Tabla 23: Tabla <i>estudioelectrofisiologico</i> e información relevante de sus atributos	227
Tabla 24: Tabla <i>eventomedico</i> e información relevante de sus atributos	227
Tabla 25: Tabla <i>generador</i> e información relevante de sus atributos	228
Tabla 26: Tabla <i>hipertensionpulmonar</i> e información relevante de sus atributos	229
Tabla 27: Tabla <i>holter</i> e información relevante de sus atributos	230
Tabla 28: Tabla <i>indicacion</i> e información relevante de sus atributos	230
Tabla 29: Tabla <i>ingreso</i> e información relevante de sus atributos	230
Tabla 30: Tabla <i>lopd</i> e información relevante de sus atributos	230
Tabla 31: Tabla <i>paciente</i> e información relevante de sus atributos	231
Tabla 32: Tabla <i>problema</i> e información relevante de sus atributos	232

Tabla 33: Tabla <i>problemasactivos</i> e información relevante de sus atributos	232
Tabla 34: Tabla <i>procedimiento</i> e información relevante de sus atributos	233
Tabla 35: Tabla <i>programacionbradicardia</i> e información relevante de sus atributos	233
Tabla 36: Tabla <i>programaciontaquicardia</i> e información relevante de sus atributos	234
Tabla 37: Tabla <i>resincronizacioncardiaca</i> e información relevante de sus atributos	235
Tabla 38: Tabla <i>tipocomplicacion</i> e información relevante de sus atributos	235
Tabla 39: Tabla <i>visita</i> e información relevante de sus atributos	236
Tabla 40: Prueba de acceso a la aplicación	421
Tabla 41: Prueba de búsqueda de paciente	421
Tabla 42: Prueba de datos demográficos del paciente	421
Tabla 43: Estado del estado nutricional del paciente	422
Tabla 44: Prueba para consulta de alergias	422
Tabla 45: Prueba de crear alergias	422
Tabla 46: Prueba de modificación de alergia de paciente	422
Tabla 47: Prueba de consulta de antecedentes	423
Tabla 48: Prueba para crear antecedentes	423
Tabla 49: Prueba de modificación de antecedente	423
Tabla 50: Prueba de consulta de indicaciones	423
Tabla 51: Prueba para crear indicaciones	424
Tabla 52: Prueba para modificar indicaciones	424
Tabla 53: Prueba de consulta de procedimientos	424
Tabla 54: Prueba para crear procedimientos	424
Tabla 55: Prueba para modificar procedimientos	425
Tabla 56: Prueba de consulta de ingresos	425
Tabla 57: Prueba para crear ingresos	425
Tabla 58: Prueba para modificar ingresos	425
Tabla 59: Prueba de consulta de <i>outcomes</i>	426
Tabla 60: Prueba para grabar <i>outcomes</i>	426
Tabla 61: Prueba para modificar <i>outcomes</i>	426
Tabla 62: Prueba de consulta de comentario	426
Tabla 63: Prueba para grabar comentario	427
Tabla 64: Prueba para modificar comentario	427
Tabla 65: Prueba de consulta de pruebas médicas	427
Tabla 66: Prueba para crear pruebas médicas	427
Tabla 67: Prueba para modificar una prueba médica	428
Tabla 68: Prueba para generar informe de manera automática	428

Tabla 69: Prueba de consulta de los sistemas de paciente	428
Tabla 70: Prueba para crear un sistema implantado	429
Tabla 71: Prueba para modificar un sistema implantado	429
Tabla 72: Prueba para consultar problemas activos	429
Tabla 73: Prueba de consulta de las visitas del paciente	429
Tabla 74: Prueba para crear una visita del paciente	430
Tabla 75: Prueba para modificar una visita del paciente	430
Tabla 76: Prueba para auditar los datos de las BB.DD	430
Tabla 77: coeficiente de reducción establecido por el COIT	438
Tabla 78: Tabla de costes de los recursos <i>Hardware</i>	439
Tabla 79: Tabla de costes de los recursos <i>Software</i>	439
Tabla 80: Tabla de costes de redacción	440
Tabla 81: Tabla de costes totales	440

1. Introducción.

1. Introducción.

1.1 Descripción general del proyecto final de carrera.

Desde la aparición de la programación en la ingeniería se ha tratado de aplicarla en beneficio de la salud. Este contexto, unido al gran interés que existe sobre el conocimiento de las enfermedades, nos revela la importancia que tiene avanzar en la tecnología e informática para la sanidad. Muchas de las herramientas de trabajo de organismos públicos y privados de la salud de todo el mundo se apoyan en aplicaciones desarrolladas con tareas específicas que colaboran con su labor diaria.

El Hospital Universitario de Gran Canaria Dr. Negrín, referido de ahora en adelante HUGCDN, a través de una solicitud formal, demanda mejoras en las herramientas de Cardiología usadas actualmente. Para satisfacer concretamente una de estas demandas, se desarrolla una aplicación web para el control de la estimulación cardíaca, protocolos de pruebas diagnósticas y seguimiento de pacientes de Cardiología del Hospital Universitario de Gran Canaria Dr. Negrín.

1.2 Contexto y ámbito del proyecto.

Actualmente el servicio de Cardiología del Hospital Universitario de Gran Canaria Doctor Negrín dispone de dos aplicaciones web independientes para la captura de datos e informes en su trabajo diario. Estas dos aplicaciones están destinadas, por una parte, a la estimulación cardíaca y, por la otra, a la elaboración de informes sobre las pruebas diagnósticas realizadas además de la recogida de datos en los protocolos aplicados.

Además de recoger los datos y realizar los informes, también hace un seguimiento del paciente; gestionando, de ser necesario, la agenda de visitas que posteriormente debe realizar el mismo.

Funciones de las herramientas de trabajo



Figura 1: Ilustración de las herramientas informáticas de trabajo de la especialidad de Cardiología.

De estas dos herramientas web comentadas; la primera, *Cardioestim*, efectúa un control sobre los dispositivos y electrodos implantados en los pacientes para realizar la estimulación cardíaca, tales como desfibriladores, marcapasos y holter.

N.H.C.

999999 - PACIENTE PRUEBA RECETA
 DNI: 92920001R T Fecha de Nacimiento: 03/12/1940
 Telefonos: 655555555 / 958787847

Sistemas Visita Ficha Paciente Ficha Paciente Outcomes Historico Visitas Historico Problemas Consultar Informe Agenda

Sistemas

Sexo: **Mujer** Fecha de nacimiento: **03/12/1940** Edad actual: **74**
 Peso: **80.7** Kg. Estatura: **175.55** cm. Dirección: **PASEO CALLE DE PRUEBA , 5 5 3**
 B.S.A.: **1.983** I.M.C.: **26.18** Sobrepesol

Lista de Generadores

Fecha Implante	Tipo	Fabricante	Fecha Explante
02/11/2009	DEFIBRILADOR	MEDTRONIC	03/11/2009
03/11/2009	DEFIBRILADOR	MEDTRONIC	04/11/2009
04/11/2009	DEFIBRILADOR	GUIDANT	11/11/2009
11/11/2009	DEFIBRILADOR	MEDICO	12/11/2009
12/11/2009	DEFIBRILADOR	MEDTRONIC	13/11/2009
13/11/2009	DEFIBRILADOR	ST JUDE	17/11/2009
17/11/2009	MARCAPASOS	ST JUDE	17/11/2009
17/11/2009	DEFIBRILADOR	MEDTRONIC	19/11/2009
18/11/2009	DEFIBRILADOR	ST JUDE	20/11/2009
19/11/2009	DEFIBRILADOR	GUIDANT	19/11/2009
19/11/2009	DEFIBRILADOR	MEDTRONIC	20/11/2009
23/11/2009	DEFIBRILADOR	MEDTRONIC	23/11/2009
26/11/2009	DEFIBRILADOR	GUIDANT	30/11/2009
30/11/2009	DEFIBRILADOR	VITATRON	11/12/2009
11/12/2009	DEFIBRILADOR	MEDTRONIC	13/01/2010
13/01/2010	MARCAPASOS	MEDTRONIC	14/01/2010
19/01/2010	DEFIBRILADOR	MEDTRONIC	23/03/2010
23/03/2010	DEFIBRILADOR	MEDTRONIC	10/11/2015

Lista de Electrodos

Fecha Implante	Camara	Posicion	Amplitud	Impedancia	Umbral	Fecha Explante
28/10/2009	VENTRICULO DERECHO	APEX VD	34	34	34*4	20/11/2009
02/11/2009	VENTRICULO DERECHO	TRACTO DE SALIDA VD	8	8	8*8	20/11/2009
17/11/2009	AURICULA DERECHA	OREJUELA DERECHA	45	45	45*45	28/12/2009
19/11/2009	AURICULA DERECHA	OREJUELA DERECHA	3435	23433	34*34	29/12/2009
19/11/2009	VENTRICULO IZQUIERDO	VENA POSTERIOR	87	78	87*8	20/11/2009
20/11/2009	VENTRICULO IZQUIERDO	VENA POSTEROLATERAL	2	2	3*44	20/11/2009
20/11/2009	AURICULA DERECHA	OREJUELA DERECHA	3	3	2*4	21/11/2009
25/11/2009	VENTRICULO DERECHO	APEX VD	34	3	35*34	27/11/2009
11/12/2009	VENTRICULO IZQUIERDO	VENA LATERAL	78	78	87*78	05/01/2010
29/12/2009	VENTRICULO DERECHO	APEX VD	8778	7867	78*78	05/01/2010
29/12/2009	VENTRICULO IZQUIERDO	VENA ANTEROLATERAL	45	45	45*45	29/12/2009
29/12/2009	VENTRICULO DERECHO	TRACTO DE SALIDA VD	45	45	54*45	05/01/2010
05/01/2010	AURICULA DERECHA	OREJUELA DERECHA	99	99	9*9	18/01/2010
12/01/2010	VENTRICULO IZQUIERDO	VENA POSTEROLATERAL	123	13	12*123	13/01/2010
14/01/2010	VENTRICULO DERECHO	TRACTO DE SALIDA VD	78	78	7*788	---
19/01/2010	VENTRICULO DERECHO	VENA ANTEROLATERAL	55	55	55*55	---

Captura 1: Captura de pantalla de Sección "Sistemas" del Software *Cardioestim*.

Para imprimir el informe, pulse en el boton

 Servicio Canario de la Salud Hospital Universitario de Las Palmas de Gran Canaria		NHC: 999999 PACIENTE PRUEBA RECETA Fecha Nacimiento: 03/12/1940 DNI/Pasaporte: 9292001R T Nº afiliación: 53099999929 Teléfonos: 555555555 Domicilio: PASEO CALLE DE PRUEBA, 5 9 3 Población: AGUATONA, Provincia: Palmas, Las	
INFORME ELECTROFISIOLOGIA ESTIMULACION CARDIACA (929 449919 - 929 449548)			
Dispositivo		Fecha Implante: 23/03/2010	
Dispositivo: 35234		Fabricante: MEDTRONIC	
Tipo: DESFIBRILADOR			
Diagnosticos			
Bateria: Voltios	Situacion Bateria:	Tiempo Carga: Segundos	
Fibrilacion Auricular: No	Taquicardia Ventricular: No	Otras Arritmias: No	
Descargas: No	ATP: No	Impedancia Descarga: Ohmios	
Parametros			
ZONA TV-1: 54	ZONA TV: 53	ZONA FV: 3	
LSF: 789	LIF: 875	LMS: 78	
Modo: DDD	Sensor		
Tipo Estimulacion Ventricular: LV	Retraso Ventriculo:	78 mseg.	
Camara VENTRICULO DERECHO	Sensibilidad: 3	Salida V: 2	Salida MS: 2
Comentario Visita:		Tratamiento:	
		D00D4D44 D4D34DD D34D43DD D303DD D34DQ334D34 xxxd xdsdiddidid	
Debería acudir para revision el a las horas.			
FIRMA:			
Las Palmas de G.C. a 03 del 10 de 2012			

Captura 4: Captura de pantalla de Sección "Consultar informe" del Software *Cardioestim*.

Por otro lado; la segunda, *Cardioweb*, es la herramienta encargada de todo lo relacionado con las pruebas diagnósticas como ecocardiografías, ergometrías, estudios con *holter*, cateterismo, estudios electrofisiológicos, hipertensión pulmonar, resincronización cardíaca o ecocardiografías transtorácicas; usando, para ello, una serie de protocolos específicos como en el caso de la ecocardiografía.

N.H.C.

999999 - PACIENTE PRUEBA RECETA
 DNI: 92920001R T Fecha de Nacimiento: 03/12/1940
 Telefonos: 555555555 / 958787847 Seg.Soc.: 530999999929

ECOCARDIOGRAFIA	ERGOMETRIA	HOLTER	CATTERISMO	E.E.F.
R.T.P.	RESINCR0	E.T.T.	INFORME	CONSULTAR

Médico Asignado

INFORME DE ECOCARDIOGRAFÍA

Captura 5: Captura de pantalla del generador de informes de *Cardioweb*.

N.R.C.

999999 - PACIENTE PRUEBA RECETA
 DNI: 92920001R T Fecha de Nacimiento: 03/12/1940
 Telefonos: 55555555 / 958787847 Seg.Soc.: 530999999929

ECOCARDIOGRAFIA	ERGOMETRIA	HOLTER	CATETERISMO	E.E.F.
H.T.P.	RESINCRIO	E.T.T.	INFORME	CONSULTAR

H.T.P.

AI (mm)	<input type="text"/>	Raiz Ao (mm)	<input type="text"/>
DTDVI (mm)	<input type="text"/>	DTSVI (mm)	<input type="text"/>
TIV (mm)	<input type="text"/>	PP (mm)	<input type="text"/>
E-TIV (mm)	<input type="text"/>	DTDVD (mm)	<input type="text"/>
Teicholtz (%)	<input type="text"/>	Arteria Pulmonar (mm)	<input type="text"/>
Frecuencia Cardiaca (bpm)	<input type="text"/>	Tension arterial (mmHg)	<input type="text"/>
T1	<input type="text"/>	T2	<input type="text"/>
AreaAD (cm2)	<input type="text"/>	D1 (mm)	<input type="text"/>
AreaVD (cm2)	<input type="text"/>	D2 (mm)	<input type="text"/>
TAPSE (mm)	<input type="text"/>	Indice de excentr VI	<input type="text"/>
GP	<input type="text"/>	Vel E	<input type="text"/>
dP/dt	<input type="text"/>	Vel A	<input type="text"/>
G1	<input type="text"/>	GP	<input type="text"/>
G2	<input type="text"/>	GM	<input type="text"/>

Captura 6: Captura de pantalla de registro de parámetros de dispositivos de *Cardioweb*.

En el día a día, a veces, el acceso a la información necesaria resulta a veces un proceso lento debido a la dispersión de los sistemas de información utilizados; por ejemplo, requiriendo del uso de dos o más aplicaciones para atender cada visita médica.

Con el objetivo de mejorar el rendimiento en la especialidad de Cardiología de este Hospital e impulsar su desarrollo usando herramientas adecuadas de trabajo, es fundamental mejorar el acceso y procesamiento a/de los datos que actualmente existe, detectadas las deficiencias en cómo se trata la información requerida actualmente.

1.3 Objetivos.

El objetivo principal de este Proyecto Fin de Carrera (PFC) es desarrollar un paquete software para simplificar el trabajo a los especialistas en cardiología, creando una aplicación de fácil uso que unifique todos los servicios que ofrecen ahora las aplicaciones ya existentes y antes nombradas, *Cardiostim* y *Cardioweb*. Para ello se pretende añadir nuevas funcionalidades y usar tecnologías en su desarrollo, ayudando así a la toma de datos y almacenamiento de los parámetros de generadores y electrodos implantados, facilitando el seguimiento de las visitas de los pacientes, y los protocolos de las pruebas diagnósticas así como generando los informes correspondientes que sean necesarios.

Para lograr esto, en este PFC se desarrollará una aplicación web capaz de guardar la información de un generador o de un electrodo en el momento en el que se implanta, para así, poder revisar dicha información con posterioridad, si es el caso. Además, una vez almacenada la información del implante, la aplicación permitirá llevar un seguimiento del generador o del electrodo cada vez que el paciente asista a la consulta, recogiendo una serie de datos que sirven como histórico de la información relevante recopilada por los dispositivos que lleva implantado el paciente.

Así, tras cada visita al especialista, se podrá entregar un informe al paciente con los parámetros más relevantes de los dispositivos implantados y una serie de recomendaciones y anotaciones escritas por el cardiólogo que atiende al paciente en la visita. Para lograr esto, también se realizará control de acceso de los usuarios cuando la información de las consultas, pruebas e informes es registrada, visitada y actualizada.

Además, se requerirá una aplicación donde el médico pueda recoger los datos de cada uno de los ocho tipos de pruebas médicas citadas y que pueda guardar los valores de cada uno de estos protocolos. Poniendo un ejemplo, se debe guardar los datos de la prueba resincronización cardíaca [1], *resincro* en el formulario de *Cardioweb*, y; además, importar esos datos al informe de la correspondiente prueba, de tal forma que se simplifique la forma en la que el médico deba introducir los valores y, posibilitar, que estos sean accesibles rápidamente para, por ejemplo, crear el informe correspondiente pudiendo utilizar para ello plantillas de texto ya predefinidas.

1.4 Protocolos usados.

Para asegurar que en este PFC se cumplió con los objetivos en términos de calidad, coste y plazos, se usó la metodología MÉTRICA Versión 3 [2], la cual hace referencia al Modelo de Ciclo de Vida de Desarrollo propuesto en la norma ISO 12.207 "Information technology – Software life cycle processes" [3], y a las normas ISO/IEC TR 15.504/SPICE "Software Process Improvement and Assurance Standards Capability Determination" [4], UNE-EN-ISO 9001:2000 "Sistemas de Gestión de la Calidad. Requisitos" [5], UNE-EN-ISO 9000:2000 "Sistemas de Gestión de la Calidad. Fundamentos y Vocabulario" [6] y el estándar IEEE 610.12-1.990 "Standard Glossary of Software Engineering Terminology" [7]. Igualmente se han tenido en cuenta otras metodologías como "Structured systems analysis and design method" (SSADM) [8], "Merise" [9], "Information Engineering" [10], MAGERIT "Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información promovida por el Consejo Superior de Informática" [11] y "EUROMÉTODO" [12].

Asimismo, se siguió el conjunto de estándares Health Level Seven (HL7 Internacional) [13], el cual proporciona estándares de interoperabilidad que mejoran la atención en salud, optimizan el flujo de trabajo, reducen la ambigüedad y mejoran la transferencia de conocimientos entre todos los interesados, incluidos los prestadores de servicios de salud, organismos gubernamentales, la comunidad de proveedores y los pacientes. Además; se hizo seguimiento, a través del equipo profesional de informática, de tareas específicas para el cumplimiento de la serie de normas ISO/IEC 27000:2014 [14] e Information Technology Infrastructure Library (ITIL) [15], conjunto de procedimientos de gestión ideados aplicadas por el Hospital para lograr calidad y eficiencia en sus operaciones en el campo de las tecnologías de la información.

En el desarrollo de este PFC concretamente, la desarrolladora tuvo autorización para el tratamiento de datos de carácter personal dentro del Hospital, a través de un contrato de confidencialidad, de acuerdo a la LEY ORGÁNICA 15/1999 de 13 de Diciembre de protección de datos de Carácter Personal [16], pudiendo acceder exclusivamente a los datos necesarios para los fines indicados en el mismo, cumpliendo el deber de confidencialidad y accediendo solo al registro de los datos en las dependencias del Departamento de Informática, quedando expresamente prohibido el registro en soporte físico de los datos contenidos en la base de datos de carácter personal.

1.5 Estructura de la memoria.

Esta memoria de proyecto fin de carrera es un documento escrito donde se realiza un seguimiento del proyecto de forma detallada desde el nacimiento de la idea hasta la obtención del resultado final. Los capítulos donde se detalla esta información se presentan a continuación.

- **Capítulo 1.** Introducción. En este bloque, se describe el proyecto de manera general, los objetivos perseguidos en entre proyecto, y los protocolos usados para el desarrollo de este software médico.
- **Capítulo 2.** Aplicación Web en su contexto. En este capítulo, se pone en contexto al lector de qué es una aplicación web, la filosofía del desarrollo ágil de aplicaciones, qué es un marco de trabajo y los diferentes entornos de aplicación web necesarios.
- **Capítulo 3.** Planificación de Sistemas de Información PSI. En esta parte de la memoria se llevará a cabo la Planificación del Sistemas de Información, primer proceso de Métrica 3, para cubrir las necesidades detectadas y los objetivos

establecidos.

- **Capítulo 4.** Tecnologías empleadas. En este capítulo se explican las diferentes tecnologías empleadas para poder llevar a cabo este proyecto.
- **Capítulo 5.** Aplicación cardiología. En el quinto bloque se detalla la configuración necesaria del marco de trabajo usado para el desarrollo de la aplicación de Cardiología y toda la descripción de ésta; sus procesos y subprocesos, la integración de la base de datos en la aplicación web, auditoría de los datos, etc.
- **Capítulo 6.** Test de la aplicación. Se presenta una serie de pruebas realizadas para testear la aplicación y su correcto funcionamiento.
- **Capítulo 7.** Conclusiones. Se realiza la presentación de la conclusión final de este trabajo y análisis de los objetivos marcados. Finalmente se proponen las posibles mejoras y ampliaciones.
- **Capítulo 8.** Presupuesto. En este capítulo se realiza el análisis de coste de la realización del presente proyecto.
- **Capítulo 9.** Bibliografía. En la bibliografía se indica el conjunto de libros y páginas web, consultadas.

2. Aplicación Web en su contexto.

2. Aplicación web en su contexto.

2.1 Servicios web.

Un servicio web [17] es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

La principal razón para usar servicios Web es que se pueden utilizar con HTTP [18] sobre *Transmission Control Protocol* (TCP) [19] en el puerto de red 80. Dado que las organizaciones protegen sus redes mediante *firewalls* [20] (que filtran y bloquean gran parte del tráfico de Internet), cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores web. Los servicios Web utilizan este puerto, por la simple razón de que no resultan bloqueados. Es importante señalar que los servicios web se pueden utilizar sobre cualquier protocolo, sin embargo, TCP es el más común.

En la mayoría de los sistemas existentes antes de los servicios web, se trataba de una interfaz fija con poca flexibilidad y adaptabilidad a entornos o necesidades cambiantes. En cambio, los servicios web aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.

2.2 ¿Qué es una aplicación web?

Para entender que es una aplicación web debemos conocer primero el modelo arquitectónico cliente-servidor [21], ya que es el modelo básico que describe cómo los programas cooperativos funcionan en el entorno de red.

La arquitectura del cliente-servidor, consta de dos partes; el servidor, que escucha las peticiones y proporciona servicios de acuerdo con estas solicitudes, y el cliente, que establece una conexión con el servidor.

En el caso de nuestra aplicación web, el cliente (el navegador), hace una solicitud al

servidor (servidor web), a través de la red (internet). Una vez escuchada la petición, el servidor la responde suministrando datos u otros recursos de vuelta al cliente.

La red es la encargada de mediar la transferencia de datos mediante el Protocolo de Transferencia de Hipertexto, o HTTP antes mencionado, que especifica cómo se comunican el navegador y el servidor entre ellos.

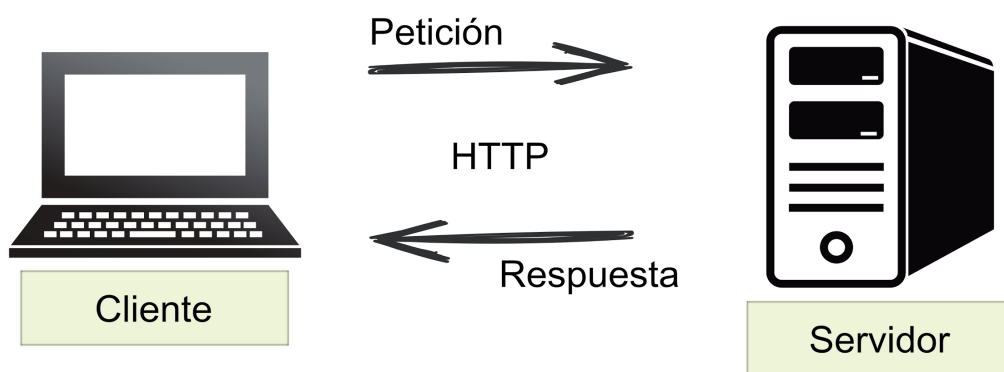


Figura 2: Representación del modelo arquitectónico cliente-servidor.

Este ciclo de petición-respuesta es la pieza fundamental de una aplicación web.

Tradicionalmente en las aplicaciones cliente-servidor, el cliente no hace mucho. Esto se conoce como cliente ligero [22], y en aplicaciones web los navegadores funcionan normalmente como clientes ligeros. Esto está cambiando hoy en día ya que los navegadores tienen que hacer cada vez más, y más trabajo en las aplicaciones web.

Una aplicación web se compone de una colección de programas del lado del cliente y del servidor, páginas HTML y otros recursos, los cuales se construyen sobre la *WorldWideWeb* [23], una de las estructuras más impresionantes que la especie humana ha construido jamás, y la *WorldWideWeb* sobre la capa de red, internet.

Estas capas mencionadas, *WorldWideWeb* e internet, forman parte del modelo de

interconexión de sistemas abiertos (OSI) [24], un modelo de referencia para los protocolos de la red de arquitectura en capas, creado en el año 1980 por la Organización Internacional de Normalización (**ISO**, *International Organization for Standardization*).



Figura 3: Modelo 7 capas OSI, modelo 4 capas TCP/IP.

Ejemplos de aplicaciones web son el correo web, tiendas virtuales web, bancos en línea, wikis, etc.

Las principales ventajas de las aplicaciones web son la ubicuidad y la conveniencia de utilizar un navegador como cliente que se ejecuta en casi cualquier dispositivo, como un PC, una tablet o un teléfono.

También tenemos la capacidad de actualizar y mantener aplicaciones web sin tocar el

software instalado en miles de equipos potenciales de cliente, siempre y cuando nos aseguremos de ésta funcione en los navegadores actuales.

Además, hay que considerar la increíble reducción en el coste de tecnologías de la información ya que es mucho más fácil mantener aplicaciones que están destinadas a ser desplegadas como aplicaciones web.

La principal desventaja del desarrollo de una aplicación web es que las aplicaciones web son difíciles de desarrollar y depurar. Hay una gran cantidad de lenguajes distintos de programación, y por ello hay que entenderlos con el fin de desarrollar aplicaciones web correctamente. Es por esto que en el capítulo 4 se dedicará a entender los lenguajes de programación que utilicemos además de las tecnologías empleadas.

2.3 Desarrollo ágil de aplicaciones.

Según el manifiesto por el desarrollo ágil de software [25], existen doce principios que lo caracterizan:

1. La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Se acepta que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Se entrega software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajan juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.

10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Desde una perspectiva de proceso de software, el desarrollo ágil es en general iterativo e incremental, es decir, se basa en el modelo en espiral, el cual comienza con el análisis de requerimientos, continúa con el diseño y la implementación de la aplicación web, y concluye el ciclo en la fase de pruebas de ésta. Una vez facilitado al HUGCDN un entregable tras pasar por un ciclo, continuamos iterando este modelo de nuevo las veces que sean necesarias, hasta que tengamos la aplicación definitiva para pasar al entorno de producción. Gracias a esto, se puede tener información del indicador más importante de éxito: saber si el software es funcional.

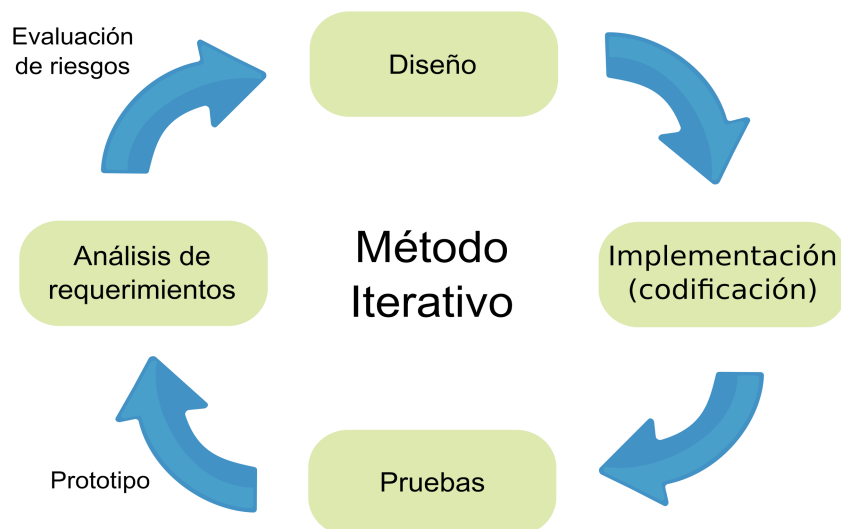


Figura 4: Gráfica de la iteración en espiral.

Con un método en cascada, es posible que no se descubra este problema hasta el final del proceso, de ahí la importancia de que el proceso sea iterativo.

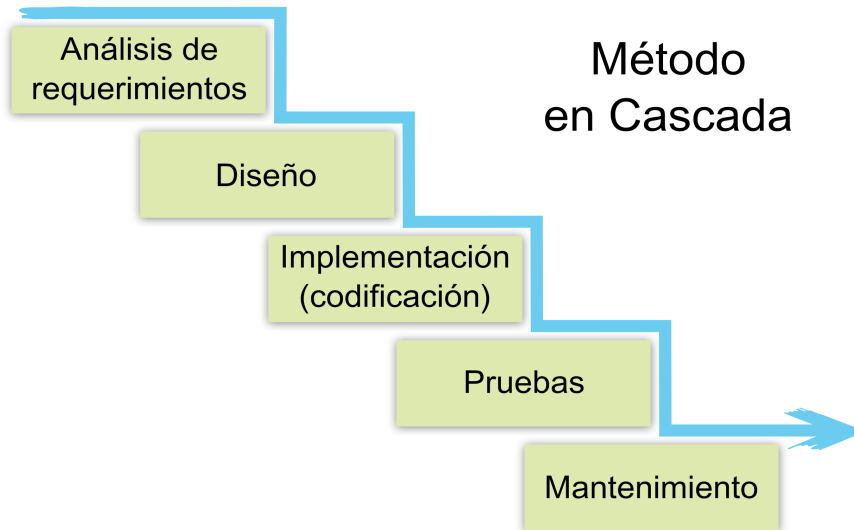
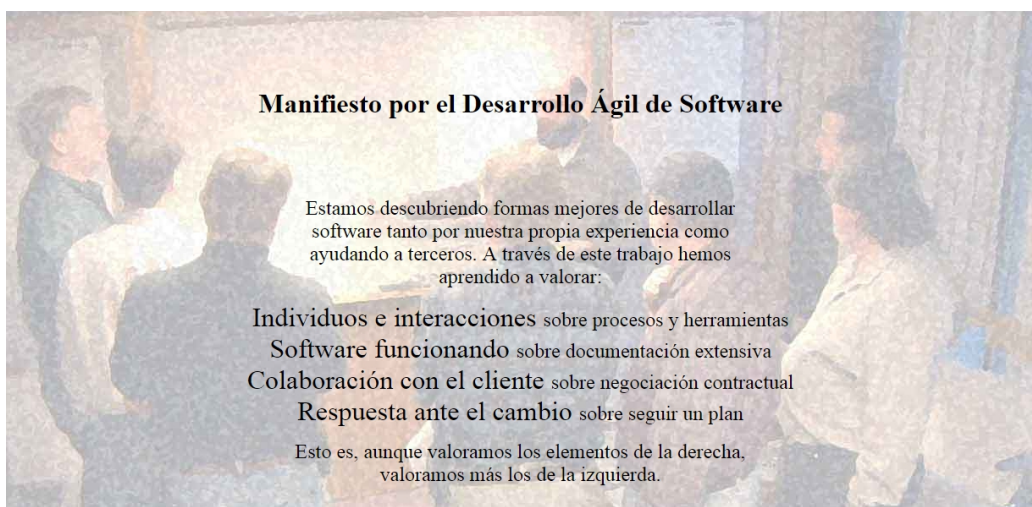


Figura 5: Gráfica de método en cascada.

En lugar de ir a través de las actividades de desarrollo del software en cascada, el método iterativo se desplaza por ellas muchas veces.

En este modelo destacan dos características, supone tener en cuenta los cambios en lugar de luchar contra ellos. y el programador juega un papel muy importante; se centra en las personas.

Por lo tanto la gestión del proyecto se basa más en las competencias del gestor del proyecto y la comunicación y coordinación, con menos énfasis en planificación, control y documentación.



Captura 7: Fragmento de la portada de página de AgileManifesto.org.



Figura 6: Ciclo del desarrollo ágil de aplicaciones [26].

El primer principio del que hablaremos para entender más en profundidad el desarrollo ágil de aplicaciones se llama convención sobre configuración. Esto significa que si se proporciona un marco de software por defecto, la tarea de desarrollo se simplifica enormemente. ya que no debemos pasar mucho tiempo configurando los componentes rutinarios. Por ejemplo, hay muchos comportamientos rutinarios asociados con las aplicaciones web, por ejemplo, cumplir con el protocolo HTTP. El marco de trabajo facilita la configuración este tipo de cuestiones, para que el programador tan solo deba especificar los aspectos no convencionales de la aplicación web; aquellos aspectos de la aplicación que son diferentes de otras aplicaciones construidas utilizando el mismo marco.

Si el programador se encuentra así mismo configurando los aspectos rutinarios de la aplicación, no está utilizando el marco de forma adecuada, está luchando contra él.

En definitiva, el desarrollador debe centrarse en crear nuevas funcionalidades para la aplicación web.

2.4 Marcos de trabajo para aplicaciones web.

Un **framework**, o **marco de trabajo** es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como

referencia, para enfrentar y resolver nuevos problemas de índole similar.

En él, desarrollamos soluciones genéricas, que son las partes que no cambian en ninguna instancia de este marco y, luego, el programador extiende el comportamiento de la estructura, para que se adapte a las necesidades de la aplicación en particular, o la funcionalidad que se pretende poner en práctica para hacer frente a un problema más específico, a través de este dominio general.

Dentro del marco de trabajo podemos encontrar soportes como:

- Conectores para comunicar la aplicación web con un motor de bases de datos.
- Apoyo para la actualización del esquema asociado a esta base de datos para cuando este modelo cambie.
- Plantillas para la generación de contenido dinámico.
- Lectura de datos en la base de datos en el momento en que se está construyendo la página, por lo que el navegador se mantiene coherente con los mismos.
- Registro de los diferentes usuarios que acceden e interactúan en la aplicación; mediante asignación de identificador único de sesión, el navegador de cada usuario transmite este identificador de sesión de vuelta al servidor web, con cada petición que se hace desde éste.
- Software's que asisten a una aplicación para interactuar o comunicarse con otras aplicaciones; *middleWare* [27].
- Internacionalización de la web, para determinar el idioma que se está usando y entregar contenido en consecuencia.
- Test para hacer pruebas automáticas, como por ejemplo *JUnit*.

Una vez que se escoge el marco adecuado y se aprende a usarlo, los proyectos futuros se pueden desarrollar mucho más rápido, y son mucho más fáciles de completar, por lo que la productividad se incrementa.

2.5 Entornos de aplicación web.

Es habitual, a la hora de afrontar un proyecto de desarrollo web, que el equipo encargado de éste soporten varios entornos de aplicación: de desarrollo, de pruebas y test, y de producción.

El entorno de desarrollo [28] es el conjunto de hardware, software, procedimientos, etc que facilitan o automatizan las actividades de desarrollo web. En este entorno, destacamos las herramientas IDE (*Integrated Development Environment*) [29], las cuales están destinadas a dar soporte informático al desarrollo integrando editores de código, herramientas de construcción automáticas, depuradores, servidores de bases de datos de pruebas, servidores web para desarrollo, etc en un único programa.

En aras de cumplir los plazos de desarrollo establecidos, se persigue que los desarrolladores puedan hacer su trabajo lo más eficiente posible, y que prioricen la generación del código que define la aplicación. Es por ello que el entorno de desarrollo además debe estar configurado de tal modo que permita el testeado de la aplicación en todas sus fases de desarrollo, y que además utilice datos de prueba para no afectar a la integridad de la información.

El entorno de pruebas y test simula y refleja dónde se desplegará la aplicación web, el entorno de producción, para utilizarlo como prueba final, y no tener que introducir cambios en éste último al editar código.

Por último, el entorno de producción es donde se desplegará la aplicación para que los usuarios puedan acceder a ella y hacer uso de sus funcionalidades.

3. Planificación de Sistemas de Información PSI (Métrica 3).

3. Planificación de Sistemas de Información (Métrica 3).

3.1 Definición y objetivos del PSI.

El sistema de información [30] es el conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, generados para cubrir las necesidades o los objetivos del proyecto. Estos elementos son las personas, datos, actividades, técnicas de trabajo, y recursos materiales que se requieren para el desarrollo de las tareas que se llevarán a cabo. Todos estos elementos interactúan para procesar los datos (incluidos los procesos manuales y automáticos) y dan lugar a información más elaborada, que se distribuye de la manera más adecuada posible para los profesionales del Hospital Negrín, en función de sus objetivos.

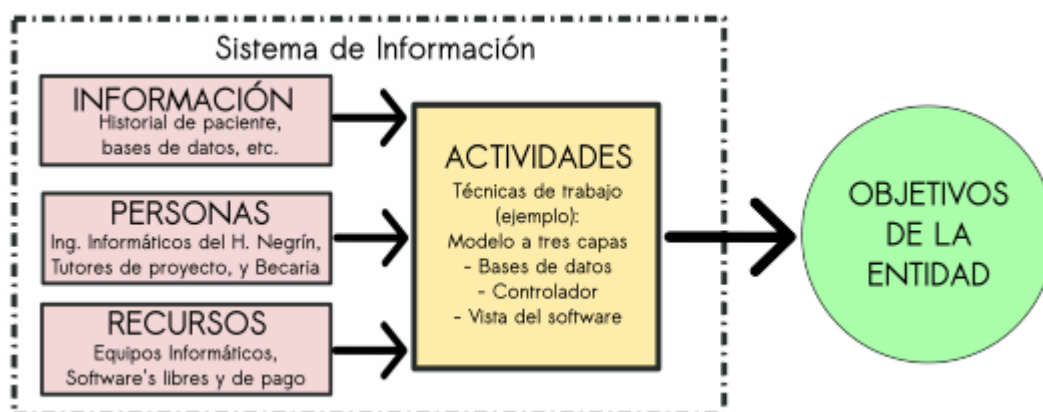


Figura 7: Ejemplo de Sistema de la información.

El Plan de Sistemas de Información para este proyecto tiene como objetivo la obtención de un marco de referencia para el desarrollo de sistemas de información que responda a los objetivos estratégicos de la organización. Así mismo, explicar el plan a las personas de la organización y a las unidades organizativas afectadas sobre las que recae el Plan, el apoyo de los altos directivos y la cualificación de los recursos de las distintas unidades implicadas, son factores críticos de éxito del Plan de Sistemas de Información.

Para la elaboración del Plan de Sistemas de Información se estudian las necesidades de información de los procesos de la organización afectados por el Plan, con el fin de definir los requisitos generales y obtener modelos conceptuales de información. Por otra parte se evalúan las opciones tecnológicas y se propone un entorno. Tras analizar las prioridades relacionadas con las distintas variables que afectan a los sistemas de

información, se elabora un calendario de proyectos con una planificación lo más detallada posible de los más inmediatos. Además, se propone una sistemática para mantener actualizado el Plan de Sistemas de Información para incluir en él todos los cambios necesarios, garantizando el cumplimiento adecuado del mismo.

3.2 Participantes en las actividades del proceso PSI.

En este PFC podemos encontrar participantes que se pueden clasificar de la siguiente manera:

1. **Perfil Directivo.** Personas con un cargo alto en la dirección de la organización, conocimiento de los diferentes objetivos que se persiguen y autoridad para validar y aprobar cada uno de los procesos realizados durante el desarrollo del Sistema de Información. Además deben tener un conocimiento del entorno y de la organización suficiente para proporcionar, a lo largo de todo el proyecto, unos requisitos del Sistema adecuados, completos y suficientemente importantes como para considerarse en el catálogo definitivo de requisitos.

Asimismo, provee los recursos necesarios para el cumplimiento de los objetivos propuestos, revisa y aprueba formalmente cada uno de los procesos, aporta información sobre las necesidades planteadas y valida los resultados con el fin de garantizar la identificación, comprensión e incorporación de todos los requisitos con las prioridades adecuadas.

2. **Perfil Jefe de Proyecto.** Personas que realizan la estimación del esfuerzo necesario para llevar a cabo el proyecto, selecciona la estrategia de desarrollo, determina la estructura del mismo seleccionando los procesos principales de MÉTRICA Versión 3 que lo integran, fija el calendario de hitos y entregas y establece la planificación del proyecto. Es el encargado de dirigir el proyecto, realizando las labores de seguimiento y control del mismo, revisión y evaluación de resultados y coordinación del equipo de proyecto. Se ocupa también de la gestión y resolución de incidencias que puedan surgir durante el desarrollo del proyecto así como de la actualización de la planificación inicial.

Además, aportan información relativa a las normas y procedimientos habituales en la organización, y ofrecen asesoramiento sobre todos los aspectos de seguridad y calidad relativos tanto al producto como al proceso seguido para su obtención, analizando los riesgos y determinando las medidas de control oportunas.

3. **Perfil Consultor.** Personas que asesoran en los aspectos relativos al negocio y en los aspectos relacionados con la informática, su aplicación e integración en la organización. Aportan su conocimiento y experiencia práctica a la hora de valorar alternativas tecnológicas para el sistema de información, especialmente durante su implantación y puesta en producción.

4. **Perfil Analista.** Personas que elaboran un catálogo detallado de requisitos que permita describir con precisión el sistema de información, para lo cual mantendrán entrevistas y sesiones de trabajo con los responsables de la organización y usuarios, actuando de interlocutor entre éstos y el equipo de proyecto en lo que a requerimientos se refiere. Estos requisitos les permiten elaborar los distintos modelos que sirven de base para el diseño; modelos de clases e interacción de objetos en análisis orientado a objeto. Asimismo realizan la especificación de las interfaces entre el sistema y el usuario.

Participan en la obtención del diseño de los datos a partir del modelo de clases, teniendo presentes las características específicas del sistema de gestión de la BB.DD concreto a utilizar, los requisitos establecidos, y las particularidades del entorno tecnológico, para conseguir una mayor eficiencia en el tratamiento de los datos.

Revisan los productos resultantes para determinar si son conformes o no a los procedimientos, normas o criterios especificados, comprobando que se han llevado a cabo las medidas preventivas o correctoras necesarias.

5. **Perfil programador.** Personas que desarrollan el código que dará lugar al producto requerido en base al diseño técnico realizado por los analistas, generando también el código asociado a los procedimientos de migración y carga inicial de datos en caso necesario. Igualmente, se encarga de la realización de las pruebas unitarias y participa en las pruebas de conjunto de la aplicación.

6. **Perfil Colaborador con el HUGCDN.** Personas que aportan otra visión y contribuyen con determinadas actividades que se lleven a cabo, conllevando a veces la toma de decisiones y seguimiento del proyecto.

7. **Perfil Usuario.** Personas que usan la aplicación diariamente como herramienta fundamental de trabajo.

3.3 Inicio del Plan de Sistemas de Información (PSI).

El objetivo de esta actividad es determinar la necesidad del Plan de Sistemas de Información y llevar a cabo el arranque formal del mismo, con el apoyo del nivel más alto de la organización. Como resultado, se obtiene una descripción general del Plan de Sistemas de Información que proporciona una definición inicial del mismo, identificando los objetivos estratégicos a los que apoya, así como el ámbito general de la organización al que afecta, lo que permite implicar a las direcciones de las áreas afectadas por el Plan de Sistemas de Información.

Además, se identifican los factores críticos de éxito y los participantes en el Plan de Sistemas de Información, nombrando a los máximos responsables.

A continuación se incluye una tabla resumen con las tareas de esta actividad:

TAREA	PRODUCTOS	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
PSI 1.1 Análisis de la necesidad del PSI.	Descripción general del PSI: <ul style="list-style-type: none"> Aprobación de inicio del PSI. 	<u>Sesiones de trabajo:</u> <u>entrevistas con el sistema experto;</u> ingenieros informáticos y los interlocutores de los usuarios de ambas aplicaciones; Jefe de Servicios y responsables de determinadas áreas de Informática del Hospital.	Perfil Directivo. Perfil Colaborador.
PSI 1.2 Identificación del alcance del PSI.	Descripción general del PSI: <ul style="list-style-type: none"> Ámbito y objetivos del PSI. Objetivos estratégicos relacionados con el PSI. Factores críticos de éxito. 	<u>Factores críticos de éxito:</u> extraídos de las sesiones de trabajo. <u>Sesiones de trabajo:</u> entrevistas con el sistema experto; ingenieros en informática, los interlocutores de los usuarios de ambas aplicaciones; Jefe de Servicios y responsables de determinadas áreas del departamento de Informática del Hospital.	Perfil Directivo. Perfil Colaborador.
PSI 1.3 Determinación de responsables.	Descripción general del PSI: <ul style="list-style-type: none"> Responsables del PSI. 	<u>Sesiones de trabajo:</u> entrevistas con el sistema experto; ingenieros en informática, los interlocutores de los usuarios de ambas aplicaciones; Jefe de Servicios y responsables de determinadas áreas del departamento de Informática del Hospital.	Perfil Directivo. Perfil Colaborador.

Tabla 1: Inicio del Plan de Sistemas de Información.

3.3.1 Tarea PSI 1.1: Análisis de la Necesidad del PSI.

Tal como ya se indicó, el Jefe de Servicios y los ingenieros informáticos que conforman el departamento de Informática del Hospital Universitario de Gran Canaria Dr. Negrín (HUGCDN), mediante una solicitud formal, requieren mejoras para las herramientas software de trabajo utilizadas normalmente en la especialidad de Cardiología; *Cardioestim* y *Cardioweb*.

Tras varias reuniones de coordinación, todos ellos resaltan la necesidad de una nueva aplicación que incorpore mejoras tecnológicas, seguridad y protocolos usados por el HUGCDN, y una mayor eficiencia de la herramienta de trabajo de los cardiólogos para un mejor seguimiento del paciente de Cardiología.

Productos de la Tarea PSI 1.1.

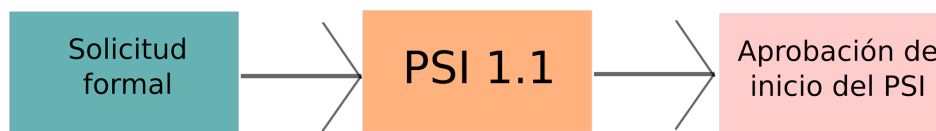


Figura 8: Productos de entrada y salida de la Tarea PSI 1.1.

3.3.1.1 Solicitud formal de PSI.

Que formalmente corresponde con la información obtenida mediante una batería de preguntas sobre la funcionalidad de las herramientas web actualmente usadas por la especialidad de cardiología *Cardioestim* y *Cardioweb*. Esto permite conocer, de manera general, los productos existentes así como los productos resultantes al final del proceso de desarrollo que cumplan las expectativas y necesidades del sistema de información.

3.3.1.2 Aprobación de inicio del PSI.

En la reunión mantenida el día 20 de Noviembre de 2015 con los diferentes perfiles definidos anteriormente se aprobó el inicio del PSI.

3.3.2 Tarea PSI 1.2: Identificación del alcance del PSI.

Se define el ámbito del Plan de Sistemas de Información en términos de procesos de la organización afectados y, como consecuencia, las direcciones de las áreas implicadas. Se determinan los objetivos estratégicos de la organización que deben ser considerados en el Plan de Sistemas de Información, así como aquellos aspectos que la dirección considera factores críticos de éxito para el mismo.

Productos de la tarea PSI 1.2.



Figura 9: Productos de entrada y salida de la Tarea PSI 1.2.

3.3.2.1 Ámbito y objetivos del PSI.

El acceso a la información necesaria para atender a un paciente resulta a veces un proceso lento debido a la dispersión de los sistemas de información utilizados; por ejemplo, necesitando del uso de dos o más aplicaciones para atender cada visita médica.

Siguiendo el desarrollo de este PFC, nuestro objetivo general será mejorar el rendimiento en las aplicaciones de la especialidad de Cardiología de este Hospital e impulsar su desarrollo usando herramientas adecuadas de trabajo. Para ello, en base a las deficiencias detectadas en cómo se trata la información requerida, es fundamental mejorar el acceso y procesamiento a/de los datos que actualmente se está empleando.

Asimismo, es necesario adaptarse a la evolución tecnológica, seguridad y protocolos aplicados actualmente por el HUGCDN ya que dichas aplicaciones fueron desarrolladas en el año 2006. También se pretende simplificar el trabajo informático de los facultativos y secretaría, reduciendo los tiempos dedicados a la interfaz para mejorar la atención prestada.

3.3.2.2 Objetivos estratégicos relacionados con el PSI.

Los objetivos estratégicos de este proyecto se centran en la corrección de fallos de

programación existentes en las dos aplicaciones *Cardioestim* y *Cardioweb* para mejorar el seguimiento del paciente, e introducir mejoras en el registro de sus pruebas médicas e intervenciones cardiológicas así como en la generación de diferentes tipos de informes médicos. Además, para mayor comodidad de los médicos especialistas, se unificarán las funcionalidades de estas dos aplicaciones en una sola y, finalmente, se añadirán otras nuevas.

Para llevar a cabo esta nueva aplicación se elegirán las tecnologías más adecuadas para el cumplimiento de los objetivos generales. Además, nos centraremos en la amigabilidad de la interfaz para que, cualquier usuario, aprenda rápidamente su manejo.

3.3.2.3 Factores críticos de éxito.

En esta sección describimos qué debe suceder para que se cumplan los objetivos preestablecidos, los factores críticos de éxito (CSFs) [31] y sus respectivos Indicadores Críticos de Rendimiento (KPIs) que permitan evaluar el cumplimiento o no de los objetivos estratégicos.

A través de la percepción de los facultativos mediante encuestas sobre los siguientes indicadores, algunos de ellos no cuantificables, nos harán saber si los objetivos son alcanzados o no. Estos indicadores a tener en cuenta son:

- Tiempo medio de uso de la aplicación web.
- Amigabilidad de la interfaz.
- Mejora del seguimiento del paciente a través de su nueva interfaz.

3.3.3 Tarea PSI 1.3: Determinación de Responsables.

Delimitado el ámbito del Plan de Sistemas de Información, se implica a las unidades organizativas afectadas, informándoles de la decisión y solicitando su participación en el estudio que se va a iniciar. Las personas seleccionadas serán los participantes en la Dirección del Plan de Sistemas de Información.

Productos de la tarea PSI 1.3.

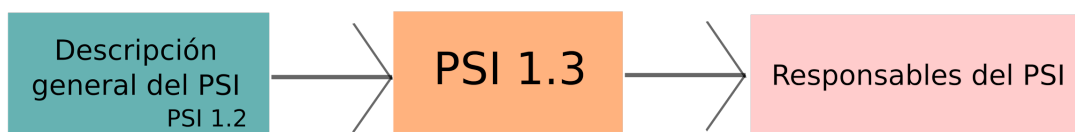


Figura 10: Productos de entrada y salida de la Tarea PSI 1.3.

Responsables del PSI.

Lo forman los distintos perfiles descritos en el apartado de Participantes de las actividades del proceso PSI anterior.

3.4 Definición y organización del PSI.

En esta actividad se detalla el alcance del plan, se organiza el equipo de personas que lo va a llevar a cabo y se elabora un calendario de ejecución. Todos los resultados o productos de esta actividad constituirán el marco de actuación del proyecto más detallado que en PSI 1 en cuanto a objetivos, procesos afectados, participantes, resultados y fechas de entrega.

TAREA	PRODUCTOS	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
PSI 2.1 Especificación del Ámbito y Alcance.	Descripción general de procesos de la organización afectados. Catálogo de objetivos de PSI: <ul style="list-style-type: none"> Objetivos generales Objetivos específicos de cada proceso (si los hubiera) 	Catalogación.	Perfil Directivo. Perfil Jefe de Proyectos. Perfil Colaborador.
PSI 2.2 Organización del PSI.	Catálogo de usuarios. Equipos de trabajo.	<u>Sesiones de trabajo</u> : se catalogaron los distintos perfiles; de usuario y de equipos de trabajos.	Perfil Directivo. Perfil Jefe de Proyectos. Perfil Colaborador.
PSI 2.3 Definición del Plan de Trabajo.	Plan de trabajo.	Mediante la técnica de Planificación podemos programar las tareas y gracias a la estimación, aproximar fechas de finalización para la obtención del software final.	Perfil Directivo. Perfil Jefe de Proyectos. Perfil Colaborador.
PSI 2.4 Comunicación del Plan de Trabajo.	Plan de trabajo: <ul style="list-style-type: none"> Aceptación del Plan de Trabajo por parte de los implicados 		Perfil Directivo. Perfil Jefe de Proyectos. Perfil Colaborador.

Tabla 2: Tabla de tareas para la definición y organización del PSI.

3.4.1 Tarea PSI 2.1: Especificación del Ámbito y Alcance.

En esta tarea se describe el ámbito de los procesos de la organización a considerar y los distintos objetivos específicos del Plan de Sistemas de Información.

Los responsables de los distintos procesos de la organización afectados por el Plan de Sistemas de Información participaron de forma activa en la definición de los objetivos, sin perder de vista los resultados de la actividad anterior (PSI 1.3).

Productos de la Tarea PSI 2.1.

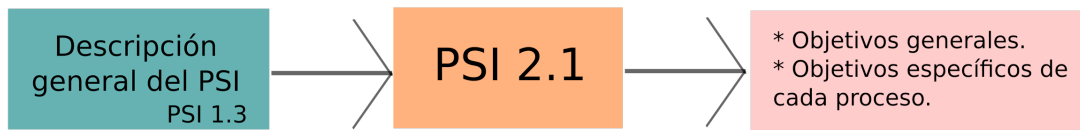


Figura 11: Productos de entrada y salida de la Tarea PSI 2.1.

3.4.1.1 Descripción general de procesos de la organización afectados.

Los principales procesos afectados en este desarrollo son las citas de pruebas cardiológicas y visitas médicas programadas por cardiología: el profesional médico asiste al paciente en consulta por padecer alguna enfermedad relacionada con el corazón, tras haberse realizado una prueba cardiológica, intervención quirúrgica o instalación de algún equipo médico como marcapasos o electrodos.

3.4.1.2 Catálogo de objetivos de PSI.



Figura 12: Gráfica de cumplimiento de objetivos [32].

Objetivos Generales	Objetivos Estratégicos	Objetivos Específicos
<ul style="list-style-type: none"> • Adaptación a la evolución tecnológica, seguridad y protocolos usados por el HUGCDN. • Mejorar la atención al paciente, agilizando la interfaz del usuario. • Integración con las diferentes plataformas del HUGCDN, tanto a nivel de Sistemas de información como de equipamiento electromédico. 	<ul style="list-style-type: none"> • Elección de las tecnologías más adecuadas para el cumplimiento de los objetivos generales. • Unificación de funcionalidades. • Nuevas funcionalidades para la generación de informes. • Nuevas funcionalidades para mejorar el seguimiento de paciente. • Amigabilidad de la interfaz. 	<ul style="list-style-type: none"> • Control de accesos, consultas y actualizaciones de los informes. • Ficha de paciente con datos relevantes para el especialista. • Almacenado de la información de un generador o de un electrodo en el momento en el que se implanta. • Generación de informes con los parámetros más relevantes de los dispositivos implantados y una serie de recomendaciones y anotaciones escritas por el cardiólogo utilizando plantillas de texto ya predefinidas. • Seguimiento del generador o del electrodo cada vez que el paciente asista a la consulta. • Recogida de una serie de datos que nos servirán como histórico de la información relevante recopilada por los dispositivos que lleva implantado el paciente. • Mejor acceso a los datos. • Mejor procesamiento de los datos.

Tabla 3: Tabla de objetivos.

3.4.2 Tarea PSI 2.2: Organización del PSI.

En esta tarea seleccionamos a los participantes, valorando el número y perfil de los profesionales de Sistemas y Tecnologías de la Información y Comunicaciones (STIC) necesarios en función de los objetivos perseguidos en este proyecto.

Adicionalmente, se concretan también aspectos logísticos relacionados con el material, salas de reuniones, estándares de documentación, etc.

Productos de la tarea PSI 2.2.

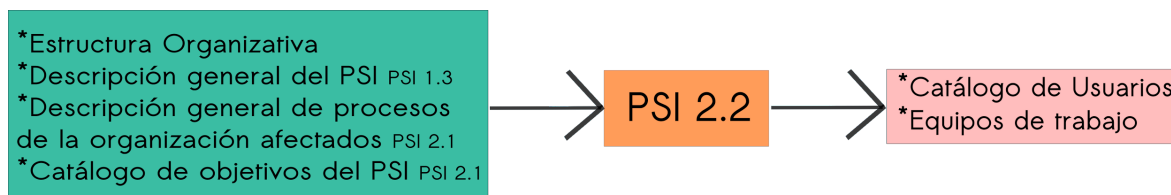


Figura 13: Productos de entrada y salida de la Tarea PSI 2.2.

3.4.2.1 Estructura Organizativa (externo).



Figura 14: Estructura organizativa.

- **Departamento de Sistemas:** Este departamento, formado por los Ingenieros, se ocupan de todo lo relacionado con equipos informáticos, redes, etc.
- **Departamento de Microinformática:** Está formado por Ingenieros de una empresa externa, ubicados en las dependencias de Informática del Hospital. Su misión es resolver todas las incidencias informáticas.
- **Departamento de Desarrollo:** Este departamento, formado por seis Ingenieros se encargan de todo lo referente a software, sistemas operativos, etc.

3.4.2.2 Catálogo de usuarios.

Tipos de Perfiles Participantes	Personal	Nº de Participantes
1. Perfil Directivo	Jefe de Servicios e Ingenieros Informáticos del Servicio de Informática del HUGCDN: Francisco Sosa, Juan Sosa y Monica Lubillo	3
2. Perfil Jefe de Proyectos	Ingeniero Informático del área de Desarrollo del Servicio de Informática del HUGCDN: Juan Sosa	1
3. Perfil Consultor	Ingenieros Informáticos del área de Desarrollo del Servicio de Informática del HUGCDN, y facultativo especialista en Cardiología	3
4. Perfil Analista	Ingeniero Informático del área de Desarrollo del Servicio de Informática del HUGCDN y Proyectante: Monica Lubillo y Yaiza Santana	2
5. Perfil Programador	Proyectante PFC: Yaiza Santana	1
6. Perfil Colaborador	Tutores de la ULPGC del Proyecto y Proyectante PFC: Jose M ^a Quintero, Luis Hernández y Yaiza Santana.	2

Tabla 4: Tabla del catálogo de usuarios.

3.4.2.3 Equipos de trabajo.

En el siguiente esquema podemos ver cómo los distintos equipos de trabajo interactúan entre ellos, y cómo también existe una colaboración externa de la Universidad de Las Palmas de Gran Canaria.

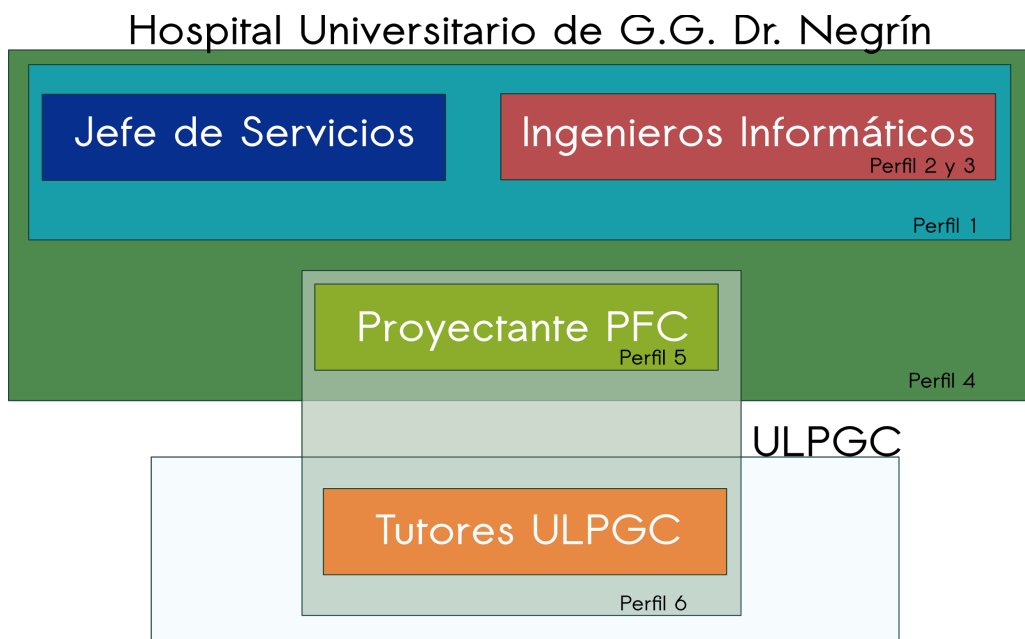


Figura 15: Equipos de trabajo interactuando.

3.4.3 Tarea PSI 2.3: Definición del Plan de Trabajo.

El objetivo de esta tarea es determinar todos los productos finales del Plan de Sistemas de Información, así como la fecha prevista de obtención y entrega de los mismos. En este caso, fue necesario planificar las distintas actividades y estimar los tiempos requeridos para llevarlas a cabo, teniendo en cuenta la disponibilidad de los usuarios del Plan de Sistemas de Información. Aquí se consideraron también los factores críticos de éxito, identificados en la actividad anterior y recogidos en la descripción general de procesos de la organización afectados, ya que podrían condicionar la elaboración del plan de trabajo.

Productos de la Tarea PSI 2.3.

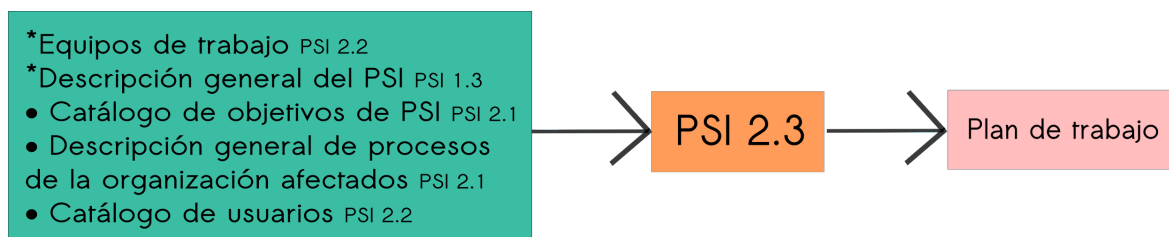


Figura 16: Productos de entrada y salida de la Tarea PSI 2.3.

3.4.3.1 Plan de trabajo.

Nombre de la tarea	Inicio	Finalización	Duración
Bloque 1. Planificación de Sistemas de Información.	16/11/15	27/11/15	9,625d
Tarea 1.1. Actividad PSI 1	16/11/15	16/11/15	4h
Tarea 1.2. Actividad PSI 2	16/11/15	18/11/15	16h
Tarea 1.3. Actividad PSI 3	18/11/15	18/11/15	4h
Tarea 1.4. Actividad PSI 4	18/11/15	20/11/15	16h
Tarea 1.5. Actividad PSI 5	18/11/15	19/11/15	10h
Entregable 1.1. Actividad PSI 5	19/11/15	19/11/15	0
Tarea 1.6. Actividad PSI 6	20/11/15	23/11/15	6h
Tarea 1.7. Actividad PSI 7	23/11/15	25/11/15	20h
Tarea 1.8. Actividad PSI 8	26/11/15	27/11/15	10h
Tarea 1.9. Actividad PSI 9	27/11/15	27/11/15	3h
Entregable 1.2. Actividad PSI 9	27/11/15	27/11/15	0
Hito 1.1	27/11/15	27/11/15	0
Bloque 2. Desarrollo de Sistemas de Información.	27/11/15	12/04/16	97,375d
Tarea 2.1. Actividad EVS 1	27/11/15	27/11/15	3h
Tarea 2.2. Actividad EVS 2	30/11/15	30/11/15	6h
Tarea 2.3. Actividad EVS 3	30/11/15	02/12/15	18h
Tarea 2.4. Actividad EVS 4	02/12/15	03/12/15	10h
Entregable 2.1. Actividad EVS 4	03/12/15	03/12/15	0
Tarea 2.5. Actividad EVS 5	03/12/15	03/12/15	3h
Hito 2.1. Actividad EVS 6	03/12/15	03/12/15	0
Tarea 2.6. Actividad ASI 1	03/12/15	07/12/15	16h
Tarea 2.7. Actividad ASI 2	07/12/15	10/12/15	24h
Entregable 2.2. Actividad ASI 2	10/12/15	10/12/15	0
Tarea 2.8. Actividad ASI 3	08/12/15	09/12/15	10h
Tarea 2.9. Actividad ASI 4	08/12/15	09/12/15	12h
Tarea 2.10. Actividad ASI 5	08/12/15	09/12/15	10h
Tarea 2.11. Actividad ASI 6	08/12/15	09/12/15	10h
Tarea 2.12. Actividad ASI 7	08/12/15	09/12/15	12h
Tarea 2.13. Actividad ASI 8	08/12/15	11/12/15	24h
Tarea 2.14. Actividad ASI 9	11/12/15	15/12/15	18h
Entregable 2.3. Actividad ASI 9	15/12/15	15/12/15	0
Tarea 2.15. Actividad ASI 10	15/12/15	17/12/15	18h
Hito 2.2. Actividad ASI 11	17/12/15	17/12/15	0
Tarea 2.16. Actividad DSI 1	18/12/15	25/12/15	6d
Tarea 2.17. Actividad DSI 2	22/12/15	29/12/15	6d
Tarea 2.18. Actividad DSI 3	22/12/15	29/12/15	6d
Tarea 2.19. Actividad DSI 4	22/12/15	29/12/15	6d
Tarea 2.20. Actividad DSI 5	22/12/15	29/12/15	6d
Tarea 2.21. Actividad DSI 6	22/12/15	30/12/15	7d
Tarea 2.22. Actividad DSI 7	31/12/15	05/01/16	4d
Entregable 2.4. Actividad DSI 7	05/01/16	05/01/16	0
Tarea 2.23. Actividad DSI 8	06/01/16	12/01/16	5d
Tarea 2.24. Actividad DSI 9	06/01/16	12/01/16	5d
Tarea 2.25. Actividad DSI 10	06/01/16	12/01/16	5d
Tarea 2.26. Actividad DSI 11	06/01/16	12/01/16	5d
Entregable 2.5. Actividad DSI 11	12/01/16	12/01/16	0
Hito 2.3. Actividad DSI 12	12/01/16	12/01/16	0
Tarea 2.27. Actividad CSI 1	13/01/16	13/01/16	8h
Tarea 2.28. Actividad CSI 2	14/01/16	23/03/16	50d
Entregable 2.6. Actividad CSI 2	23/03/16	23/03/16	0
Tarea 2.29. Actividad CSI 3	22/03/16	24/03/16	3d
Tarea 2.30. Actividad CSI 4	23/03/16	25/03/16	3d
Tarea 2.31. Actividad CSI 5	28/03/16	31/03/16	4d
Tarea 2.32. Actividad CSI 6	25/03/16	01/04/16	6d
Entregable 2.7. Actividad CSI 6	01/04/16	01/04/16	0
Tarea 2.33. Actividad CSI 7	25/03/16	28/03/16	2d
Tarea 2.34. Actividad CSI 8	25/03/16	30/03/16	4d
Hito 2.4. Actividad CSI 9	01/04/16	01/04/16	0

Tarea 2.35. Actividad IAS 1	04/04/16	04/04/16	1d
Tarea 2.36. Actividad IAS 2	05/04/16	05/04/16	1d
Tarea 2.37. Actividad IAS 3	06/04/16	06/04/16	1d
Tarea 2.38. Actividad IAS 4	06/04/16	06/04/16	1d
Tarea 2.39. Actividad IAS 5	07/04/16	07/04/16	1d
Tarea 2.40. Actividad IAS 6	08/04/16	08/04/16	1d
Tarea 2.41. Actividad IAS 7	06/04/16	07/04/16	1d
Entregable 2.8. Actividad IAS 7	07/04/16	07/04/16	0
Tarea 2.42. Actividad IAS 8	05/04/16	06/04/16	1d
Tarea 2.43. Actividad IAS 9	11/04/16	11/04/16	1d
Entregable 2.9. Actividad IAS 9	11/04/16	11/04/16	0
Tarea 2.44. Actividad IAS 10	12/04/16	12/04/16	1d
Hito 2.5	12/04/16	12/04/16	0
Bloque 3. Memoria y presentación del proyecto final de carrera.	16/11/15	09/05/16	125,5d
Tarea 3.1	16/11/15	29/04/16	119,5d
Tarea 3.2	02/05/16	09/05/16	6d
Entregable 3.1	29/04/16	29/04/16	0
Entregable 3.2	09/05/16	09/05/16	0
Hito 3.1	09/05/16	09/05/16	0

Tabla 5: Tabla del plan de trabajo.

3.4.4 Tarea PSI 2.4: Comunicación del Plan de Trabajo.

Se comunica a los usuarios del Plan de Sistemas de Información el Plan de Trabajo a llevar a cabo con el fin de que sea aceptado.

Productos de la Tarea PSI 2.4.



Figura 17: Productos de entrada y salida de la Tarea PSI 2.4.

3.4.4.1 Aceptación del Plan de Trabajo.

En el día 4 de Diciembre se acepta el plan de trabajo por parte de los participantes.

3.5 Estudio de la información relevante y de los sistemas de información actuales.

En esta actividad recopilamos y analizamos todos los antecedentes generales que puedan afectar a los procesos y a las unidades organizativas implicadas en el Plan de

Sistemas de Información, así como a los resultados del mismo.

TAREA	PRODUCTOS	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
PSI 3.1 Selección y Análisis de Antecedentes.	Valoración de antecedentes.	Sesiones de trabajo: entrevistas con el sistema experto e interlocutores de los usuarios.	Perfil Consultor. Perfil Directivo. Perfil Colaborador.
PSI 3.2 Valoración de antecedentes.	Catálogo de requisitos: <ul style="list-style-type: none"> Requisitos generales. Catálogo de normas del PSI.	Catalogación de los requisitos generales del software.	Perfil Consultor. Perfil Directivo. Perfil Colaborador.

Tabla 6: Tareas del estudio de la información relevante y de los sistemas de información actuales.

3.5.1 Tarea PSI 3.1: Selección y Análisis de Antecedentes.

En esta tarea se revisa toda la información y documentación relevante a considerar, teniendo en cuenta todos aquellos antecedentes de interés.

Para la obtención de esta información, se entrevistó a las personas de la organización que pudiesen aportar información adicional sobre antecedentes que deban ser considerados en el Plan de Sistemas de Información, al margen de la documentación disponible.

Productos de la Tarea PSI 3.1.

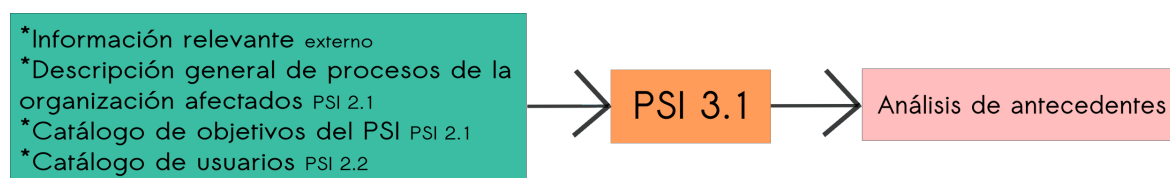


Figura 18: Productos de entrada y salida de la Tarea PSI 3.1.

3.5.1.1 Información relevante.

3.5.1.1.1 Información relevante sobre el entorno de trabajo de los Usuarios de Cardiología del HUGCDN.

Actualmente el entorno de trabajo de los Usuarios de Cardiología del HUGCDN lo forman los siguientes elementos:

MATERIAL OFICINA:

Mesa.

Silla.

Otros.

HARDWARE:

Pantalla de ordenador.

Teclado.

Ratón.

Terminal IGEL; accede a un servidor de terminales.

Servidor de terminales.

SOFTWARE:

Windows Server 2008.

Sistema Operativo Windows 7.

Navegador web Internet Explorer 8.

Intranet del Hospital HUGCDN.

Software *CardioEstim*.

Software *CardioWeb*.

Bases de datos Caché.

Otras bases de datos.

3.5.1.1.2 Información relevante sobre pruebas e intervenciones cardiológicas.

Hipertensión Pulmonar (H.T.P.) [33]: la ecocardiografía es el método no invasivo de elección para el estudio del paciente con sospecha de hipertensión pulmonar. La ecocardiografía permite estimar la presión arterial pulmonar sistólica y, además, puede proporcionar información adicional acerca de la causa y las consecuencias de la enfermedad. Se miden determinadas variables, parámetros de interés, como por ejemplo la excursión sistólica del plano del anillo tricuspídeo (TAPSE) o el índice de Tei, que seguidamente son evaluados. La ecocardiografía puede proporcionar datos que sugieran una mala evolución de la enfermedad a corto y medio plazo por lo que es imprescindible su correcta determinación y caracterización.

H.T.P.

AI (mm)	<input type="text"/>	Raiz Ao (mm)	<input type="text"/>
DTDVI (mm)	<input type="text"/>	DTSVI (mm)	<input type="text"/>
TIV (mm)	<input type="text"/>	PP (mm)	<input type="text"/>
E-TIV (mm)	<input type="text"/>	DTDVD (mm)	<input type="text"/>
Teicholtz (%)	<input type="text"/>	Arteria Pulmonar (mm)	<input type="text"/>
Frecuencia Cardiaca (lpm)	<input type="text"/>	Tension arterial (mmHg)	<input type="text"/>
T1	<input type="text"/>	T2	<input type="text"/>
AreaAD (cm2)	<input type="text"/>	D1 (mm)	<input type="text"/>
AreaVD (cm2)	<input type="text"/>	D2 (mm)	<input type="text"/>
TAPSE (mm)	<input type="text"/>	Indice de excentr VI	<input type="text"/>
GP	<input type="text"/>	Vel E	<input type="text"/>
dP/dt	<input type="text"/>	Vel A	<input type="text"/>
G1	<input type="text"/>	GP	<input type="text"/>
G2	<input type="text"/>	GM	<input type="text"/>
<input type="button" value="Guardar Valores Registrados"/>			

Captura 8: Parámetros de entrada para la prueba cardiológica Hipertensión Pulmonar.

Resincronización Cardíaca (Resincro) [34]: el tratamiento de resincronización cardíaca (TRC), a veces denominado estimulación eléctrica biventricular, es útil si los impulsos eléctricos que controlan la contracción y relajación del músculo cardíaco no se desplazan por el corazón de una manera rápida y uniforme. El ventrículo izquierdo puede contraerse una fracción de segundo después del ventrículo derecho en vez de simultáneamente. Con frecuencia, partes del ventrículo izquierdo muestran un retraso de la contracción y el dispositivo puede conseguir una contracción más breve y más uniforme. Esta falta de coordinación afecta a la capacidad del corazón de bombear con eficacia. El médico buscará signos de mala coordinación del músculo cardíaco en el trazado electrocardiograma o la ecografía (ultrasonidos).

El TRC no sólo funciona como un marcapasos sino que resincroniza el latido de los dos ventrículos estimulándolos simultáneamente y, en concreto, mejora la contracción del izquierdo. Esto mejorará la eficiencia global del corazón. El TRC, por tanto, difiere de los marcapasos típicos, que sólo estimulan el ventrículo derecho y controlan la frecuencia cardíaca. Los dispositivos de TRC no sólo mejoran los síntomas de la insuficiencia cardíaca sino que se ha demostrado que prolongan la supervivencia a largo plazo. Los médicos suelen combinar un TRC con un DAI en el mismo dispositivo. En este caso, el

dispositivo se denomina CRT-D (dispositivo de resincronización asociado a un desfibrilador).

RESINCRO

Retraso interventricular (Modo M)	<input type="text"/>	Tiempo preeyectivo aórtico	<input type="text"/>
Retraso interventricular (Tipo preeyectivo pulmonar)	<input type="text"/>	Retraso entre el SIV y la PL	<input type="text"/>
Fin del acortamiento tras cierre valvular aórtico	<input type="text"/>	Movimiento en M del SIV por DTI color	<input type="text"/>
Apreciación subjetiva del mov. asincrónico del SIV (4C)	<input type="text"/>	Tiempo de llenado diastólico con respecto al RR(4C)	<input type="text"/>
Insuficiencia mitral presistólica	<input type="text"/>	dP/dt(4C) < 800 mmHg/s	<input type="text"/>
Insuficiencia mitral > grado II	<input type="text"/>		
<input type="button" value="Guardar Valores Registrados"/>			

Captura 9: Parámetros de entrada para la prueba cardiológica Resincronización Cardíaca.

Ecocardiografía Transtorácica (E.T.T.) [35]: el ecocardiograma estándar, se denomina ecocardiograma transtorácico y se realiza colocando el transductor sobre la pared torácica obteniendo las imágenes a través de ella. Variando la posición y orientación del transductor podemos obtener imágenes de distintos planos del corazón. A saber:

- Posición paraesternal: colocando el transductor en la región paraesternal, se puede dirigir el haz de ultrasonidos en distintos ejes para observar las estructuras cardiacas. El eje más importante es el eje largo de cavidades izquierdas. Otros posibles ejes son: eje largo de entrada al ventrículo derecho (VD), eje largo de salida del VD y ejes cortos paraesternales (para la observación de las válvulas).
- Posición apical: el transductor se coloca en el 5º espacio intercostal izquierdo. Nos permite ver el corazón en cortes coronales.
- Posición subcostal: colocamos el sensor por debajo del reborde costal inferior. Se usa sobre todo en pacientes con enfermedad pulmonar obstructiva crónica (EPOC).
- Posición supraesternal: se coloca el transductor en el hueco supraesternal. Se usa para la visualización de la arteria aorta sobre todo.

E.T.T.

DTDVI(mm)	<input type="text"/>	DTSVI(mm)	<input type="text"/>
Teicholtz (%)	<input type="text"/>	TIV(mm)	<input type="text"/>
PP (mm)	<input type="text"/>	DTDVD (mm)	<input type="text"/>
AI(mm)	<input type="text"/>	Raiz Ao (mm)	<input type="text"/>
GM Mitral	<input type="text"/>	GP aórtico	<input type="text"/>
GM aórtico	<input type="text"/>	PAPs	<input type="text"/>
E-TIV (mm)	<input type="text"/>	Onda E	<input type="text"/>
Onda A	<input type="text"/>	FE (Simpson)	<input type="text"/>
<input type="button" value="Guardar Valores Registrados"/>			

Captura 10: Parámetros de entrada para la prueba cardiológica Ecocardiografía Transtorácica.

Ergometría [36]: prueba diagnóstica que consiste en realizar un registro del electrocardiograma durante un esfuerzo controlado. Se realiza de forma ambulatoria. Requiere que el paciente no fume por lo menos 8 horas antes del estudio, vaya equipado con ropa holgada y cómoda, y calzado apropiado (zapatillas de deporte o similar) para andar o correr. A su llegada se le explica la forma de realizarla.

Se le colocan unos electrodos adhesivos en el tórax, se le conecta al equipo y, siguiendo las instrucciones, debe andar o correr sobre un tapiz rodante o bicicleta estática. El paciente debe indicar al personal médico, presente en la prueba, cualquier incidencia que se presente (fatiga, cansancio, dolor en el pecho, palpitaciones, disnea, etc). En todo momento se tiene control de la TA y del electrocardiograma. Si, durante la realización del ejercicio, el paciente presenta angina, se dice que la prueba ha sido clínicamente positiva (en caso contrario, la prueba se considera clínicamente negativa).

La prueba se considera concluyente cuando el paciente ha alcanzado el 85% de la frecuencia cardíaca máxima para su edad (que se calcula con la fórmula $FCMP = 208.75 - (0.73 \times \text{edad})$).

ERGOMETRIA

PROTOCOLO	Bruce <input type="checkbox"/>	Modificado <input type="checkbox"/>	Manual <input type="checkbox"/>
Ritmo Basal	Sinusal <input type="checkbox"/>	FA <input type="checkbox"/>	Beta-bloqueante <input type="checkbox"/>
FC Basal	<input type="text"/> lpm	FC Máxima	<input type="text"/> lpm <input type="text"/> %
TA Basal	<input type="text"/> mmHg	TA Máxima	<input type="text"/> mmHg
Tiempo Ejercicio	<input type="text"/> min <input type="text"/> seg		
METS	<input type="text"/>		
Resultado	Positiva <input type="checkbox"/>	Negativa <input type="checkbox"/>	No concluyente <input type="checkbox"/>
Notas			
<div style="border: 1px solid gray; padding: 5px;"> <!-- Empty text area for notes --> </div>			
<div style="border: 1px solid orange; padding: 2px 10px; display: inline-block; color: orange;"> Guardar Valores Registrados </div>			

Captura 11: Parámetros de entrada para la prueba cardiológica Ergometría.

Holter [37]: Un monitor Holter es una máquina que registra los ritmos cardíacos en forma continua. Se lleva puesto por 24 a 48 horas durante la actividad normal. Los electrodos (pequeños parches conductores) se pegan en el pecho y se conectan por alambres a un pequeño monitor de registro. El paciente carga el monitor Holter en un bolsillo o en una bolsa que se lleva puesta alrededor del cuello o la cintura. Mientras lleva puesto el monitor, éste registra la actividad eléctrica del corazón. El paciente debe:

- Llevar un registro diario de las actividades que realiza mientras está usando el monitor y cómo se siente.
- Después de 24 a 48 horas, debe devolver el monitor al consultorio del médico.
- El médico observará los registros y mirará si ha habido algún ritmo cardíaco anormal reflejando todo ello en un informe.

Médico
Asignado



Captura 12: Herramienta para elaborar el informe de la prueba cardiológica Holter.

Cateterismo [38]: El cateterismo cardíaco es un procedimiento complejo e invasivo que consiste en la introducción de unos catéteres que se llevan hasta el corazón para valorar la anatomía del mismo y de las arterias coronarias, así como para ver la función del corazón (cuánta sangre bombea), medir presiones de las cavidades cardiacas e, incluso, saber si hay alguna válvula alterada. Además, permite ver si existen defectos congénitos (de nacimiento), como comunicaciones (orificios) en el tabique auricular o ventricular, medir concentraciones de oxígeno en diferentes partes del corazón y obtener muestras de tejido cardiaco (biopsia) para el diagnóstico de ciertas enfermedades.

Médico
Asignado



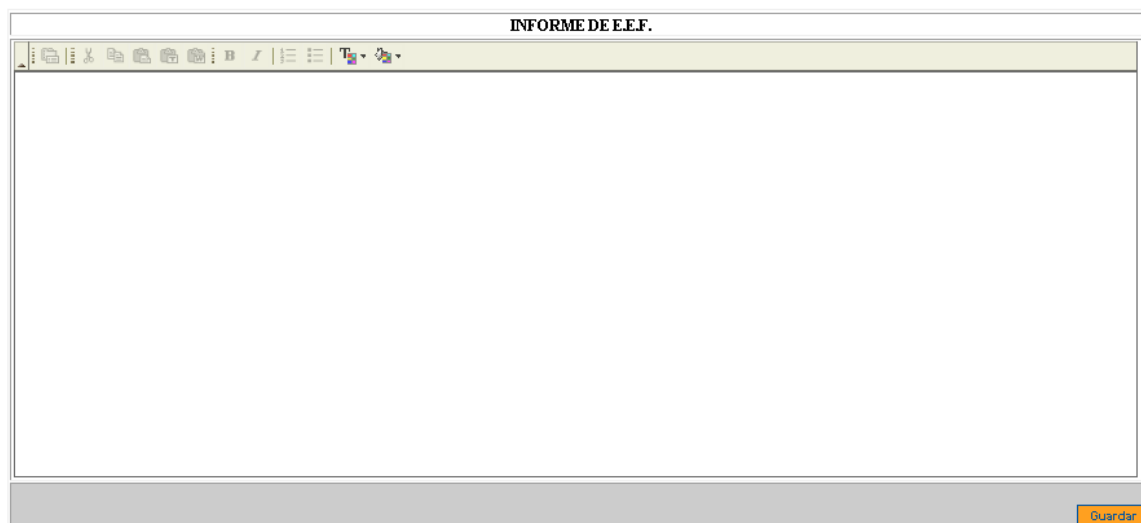
Captura 13: Herramienta para elaborar el informe de la prueba cardiológica Cateterismo.

Estudio Electrofisiológico (E.E.F) [39]: El estudio electrofisiológico es una prueba que sirve para el diagnóstico de pacientes que tienen o pueden tener alteraciones del ritmo cardíaco (arritmias). Permite conocer el tipo y gravedad de las arritmias, el lugar del corazón donde se originan y los trastornos que produce. Sirve además para enfocar mejor el tratamiento que debe aplicarse a dichas arritmias en caso de existir.

Para la realización del estudio electrofisiológico es necesario que el paciente esté en ayunas.

- El sujeto debe desvestirse y tumbarse en una camilla especial en el laboratorio de Electrofisiología. Generalmente se suele realizar la técnica tras administración de sedación al paciente.
- Se le aplica anestesia local en la zona de punción (venas y/o arterias de ingle, brazo o cuello) para que la punción no resulte dolorosa, y se introducen catéteres (cables finos, largos y flexibles) que se dirigen al corazón por medio de rayos X (radioscopia) o de otro tipo de sistemas de localización que no requieren radiación.
- Los catéteres sirven para registrar de forma permanente la actividad eléctrica del corazón desde su interior, definir el tipo de arritmia y dónde se localiza. La actividad eléctrica se muestra en unos monitores. Los catéteres también pueden servir como marcapasos si se conectan a un aparato estimulador externo.
- A veces es necesario suministrarles algún fármaco durante la prueba para precisar el diagnóstico de la arritmia. En otras ocasiones se requerirá aplicar una descarga eléctrica. Para ello se anestesia al paciente.
- El procedimiento tiene una duración variable, pudiendo durar entre 30 minutos y varias horas.
- Cuando termina, el paciente tiene que permanecer en reposo varias horas más, para evitar complicaciones en la zona de punción.

Médico
Asignado



Captura 14: Herramienta para elaborar el informe de la prueba cardiológica Estudio Electrofisiológico.

3.5.1.1.3 Información relevante sobre dispositivos y sistemas implantados en el paciente.

3.5.1.1.3.1 Generadores.

CardioEstim permite crear un listado de cualquier dispositivo implantado en el paciente. Cada dispositivo dispone de un formulario donde se registran sus características, complicaciones en el momento del implante, y datos y comentarios de interés para el seguimiento posterior del paciente.

Sistemas					
Sexo	Mujer	Peso	80.7 kg.	B.S.A.	1.983
Fecha de nacimiento	03/12/1941	Estatura	175.55 cm.	I.M.C.	26.18 Sobrepeso
Edad actual	75	Dirección	PASEO CALLE DE PRUEBA , 5 3		
Lista de Generadores					
Fecha Implante	Tipo	Fabricante	Fecha Explante		
02/11/2009	DEFIBRILADOR	MEDTRONIC	03/11/2009		
03/11/2009	DEFIBRILADOR	MEDTRONIC	04/11/2009		
04/11/2009	DEFIBRILADOR	GUIDANT	11/11/2009		
11/11/2009	DEFIBRILADOR	MEDICO	12/11/2009		
12/11/2009	DEFIBRILADOR	MEDTRONIC	13/11/2009		
13/11/2009	DEFIBRILADOR	ST JUDE	17/11/2009		
17/11/2009	MARCAPASOS	ST JUDE	17/11/2009		
17/11/2009	DEFIBRILADOR	MEDTRONIC	19/11/2009		
18/11/2009	DEFIBRILADOR	ST JUDE	26/11/2009		
19/11/2009	DEFIBRILADOR	GUIDANT	19/11/2009		
19/11/2009	DEFIBRILADOR	MEDTRONIC	23/11/2009		
23/11/2009	DEFIBRILADOR	MEDTRONIC	23/11/2009		
26/11/2009	DEFIBRILADOR	GUIDANT	30/11/2009		
30/11/2009	DEFIBRILADOR	VITATRON	11/12/2009		
11/12/2009	DEFIBRILADOR	MEDTRONIC	13/01/2010		
13/01/2010	MARCAPASOS	MEDTRONIC	14/01/2010		
19/01/2010	DEFIBRILADOR	MEDTRONIC	23/03/2010		
23/03/2010	DEFIBRILADOR	MEDTRONIC	10/11/2015		

Captura 15: Captura de pantalla del listado de dispositivos implantados en el paciente de prueba.

N.H.C. 999999 - **PACIENTE PRUEBA RECETA**
 DNI: **92920001R T** Fecha de Nacimiento: **03/12/1940**
 Telefonos: **555555555 / 958787847**

Sistemas Visita Ficha Paciente Ficha Paciente OutComes Historico Visitas Historico Problemas Consultar Informe Agenda

Generador

Modelo	<input type="text"/>	Numero de Serie	<input type="text"/>	Tipo	<input type="text"/>
Fabricante	<input type="text"/>	Fecha de Implante	<input type="text"/>		
Resincronizacion Cardiaca	<input type="text"/>	Lugar Implante	<input type="text"/>		
Medico	<input type="text"/>	Medico ayudante	<input type="text"/>	Enfermero <input type="text"/>	
Fecha explante	<input type="text"/>	Causa Explante	<input type="text"/>		
Region Implante	<input type="text"/>	Acceso	<input type="text"/>		

Complicaciones del implante

Neumotorax <input type="checkbox"/>	Hemotorax <input type="checkbox"/>	Taponamiento <input type="checkbox"/>	Derrame Pericardico <input type="checkbox"/>	Diseccion Coronaria <input type="checkbox"/>	Perforacion Coronaria <input type="checkbox"/>	Perforacion Seno Coronario <input type="checkbox"/>	Hematoma <input type="checkbox"/>
Infeccion <input type="checkbox"/>	Sepsis <input type="checkbox"/>	Trombosis Subclavia <input type="checkbox"/>	Insuficiencia Renal <input type="checkbox"/>	Insuficiencia Cardiaca <input type="checkbox"/>	Shock <input type="checkbox"/>	Muerte <input type="checkbox"/>	Estimulacion Frenica <input type="checkbox"/>
Ninguno <input type="checkbox"/>							

Comentario dispositivo

Grabar

Captura 16: Captura de pantalla de Sección “Nuevo dispositivo” del Software *CardioEstim*.

Generador

Modelo	898989	Numero de Serie	898989	Tipo	DESFIBRILADOR
Fabricante	MEDTRONIC	Fecha de Implante	02/11/2009		
Resincronizacion Cardiaca	SI	Lugar Implante	QUIROFANO CCV		
MedicoP	NOELIA CASTRO BUENO	MedicoA	OSCAR MAURICIO MORERA PI	Enfermero <input type="text"/>	
Fecha explante	03/11/2009	Causa Explante	ENDOCARDITIS		
Umbral de desfibrilacion	NO REALIZADO	Valor Umbral de desfibrilacion	89 Julios.	Impedancia de descarga 89 Ohmios.	
Region Implante	Subpectoral	Acceso	Subclavia derecha		

Complicaciones del implante

Neumotorax <input type="checkbox"/>	Hemotorax <input type="checkbox"/>	Taponamiento <input type="checkbox"/>	Derrame Pericardico <input checked="" type="checkbox"/>	Diseccion Coronaria <input type="checkbox"/>	Perforacion Coronaria <input type="checkbox"/>	Perforacion Seno Coronario <input type="checkbox"/>	Hematoma <input type="checkbox"/>
Infeccion <input type="checkbox"/>	Sepsis <input type="checkbox"/>	Trombosis Subclavia <input type="checkbox"/>	Insuficiencia Renal <input type="checkbox"/>	Insuficiencia Cardiaca <input type="checkbox"/>	Shock <input type="checkbox"/>	Muerte <input type="checkbox"/>	Estimulacion Frenica <input type="checkbox"/>
Ninguno <input type="checkbox"/>							

Comentario dispositivo

y89vdvf3

Grabar

Captura 17: Captura de pantalla de parámetros del Generador registrado en el paciente del Software *CardioEstim*.

Desfibrilador [40]. Restablece el ritmo cardíaco normal mediante la aplicación de una descarga eléctrica. Los hay externos (sobre la piel) e implantables (definitivos), este último solo está indicado para algunas situaciones específicas.

La desfibrilación se basa en la aplicación brusca y breve de una corriente eléctrica de alto voltaje para detener y revertir las arritmias cardíacas rápidas (taquicardia ventricular sostenida, fibrilación ventricular); situaciones en las que el número de latidos cardiacos

aumentan en exceso o se produce una actividad eléctrica desorganizada, debido a que alguna zona o foco del corazón 'dispara' impulsos de forma descontrolada, que no son efectivos o producen una inestabilidad hemodinámica (deterioro de los signos vitales) que pueden llevar a una persona a una parada cardíaca. El choque eléctrico detiene la arritmia, lo que permite al médico, identificar y solucionar las causas que la produjeron.

Marcapasos [41]. Este dispositivo electrónico envía impulsos al corazón para que mantenga el ritmo normal. Su implantación se realiza mediante una pequeña incisión debajo de la clavícula.

Un marcapasos artificial es un dispositivo electrónico diseñado para producir impulsos eléctricos con el objeto de estimular el corazón cuando falla la estimulación fisiológica o normal. Estos impulsos, una vez generados, necesitan de un cable conductor (o electrocatéter) que se interponga entre ellos para alcanzar su objetivo. De esta forma, un sistema de estimulación cardíaca consta de un generador de impulsos eléctricos (o marcapasos propiamente dicho) y de un cable.

Holter implantable [42]. Se trata de un dispositivo (del tamaño de un encendedor pequeño) que se implanta bajo la piel de la zona localizada bajo la clavícula izquierda y sirve para registrar el ritmo cardíaco de manera continua, durante el tiempo que permanezca implantado (generalmente meses).

Este instrumento es útil para vigilar el ritmo cardíaco y detectar irregularidades del mismo (arritmias), es decir, ritmos excesivamente rápidos (taquicardias) o por el contrario lentos (bradicardias). Esta prueba se utiliza en pacientes que consultan por síntomas (como mareos, desmayos, palpitaciones) que pueden deberse a una arritmia, pero que son poco frecuentes y por lo tanto no han podido detectarse con un electrocardiograma normal (ECG) o con un Holter convencional (el cual graba los latidos cardíacos sólo durante 24-48 horas).

3.5.1.1.3.2 Electroodos.

CardioEstim permite crear un listado de cada electrodo implantado en el paciente. En este formulario se registran sus características y datos de interés para el seguimiento posterior del paciente.

Lista de Electrodos						
Fecha Implante	Camara	Posicion	Amplitud	Impedancia	Umbral	Fecha Explante
28/10/2009	VENTRICULO DERECHO	APEX VD	34	34	34*4	20/11/2009
05/11/2009	VENTRICULO DERECHO	TRACTO DE SALIDA VD	8	8	8*8	20/11/2009
17/11/2009	AURICULA DERECHA	OREJUELA DERECHA	46	46	46*46	28/12/2009
19/11/2009	AURICULA DERECHA	OREJUELA DERECHA	3425	23423	34*34	29/12/2009
19/11/2009	VENTRICULO IZQUIERDO	VENA POSTERIOR	87	78	87*8	20/11/2009
20/11/2009	VENTRICULO IZQUIERDO	VENA POSTEROLATERAL	2	2	3*44	20/11/2009
20/11/2009	AURICULA DERECHA	OREJUELA DERECHA	3	3	2*4	21/11/2009
26/11/2009	VENTRICULO DERECHO	APEX VD	34	3	35*34	27/11/2009
11/12/2009	VENTRICULO IZQUIERDO	VENA LATERAL	76	76	67*76	05/01/2010
29/12/2009	VENTRICULO DERECHO	APEX VD	6776	7667	76*76	05/01/2010
29/12/2009	VENTRICULO IZQUIERDO	VENA ANTEROLATERAL	46	46	46*46	29/12/2009
29/12/2009	VENTRICULO DERECHO	TRACTO DE SALIDA VD	46	46	54*46	05/01/2010
05/01/2010	AURICULA DERECHA	OREJUELA DERECHA	99	99	9*9	18/01/2010
12/01/2010	VENTRICULO IZQUIERDO	VENA POSTEROLATERAL	123	13	12*123	13/01/2010
14/01/2010	VENTRICULO DERECHO	TRACTO DE SALIDA VD	76	76	7*766	...
19/01/2010	VENTRICULO DERECHO	VENA ANTEROLATERAL	68	68	68*68	...

Nuevo electrodo

Captura 18: Captura de pantalla de lista de electrodos en la Sección “Sistemas” del Software.

Existen determinados electrodos que son implantados en el paciente en determinadas intervenciones quirúrgicas. Por citar un ejemplo [43], para la resincronización cardíaca se requiere el implante de electrodos epicárdicos en el ventrículo izquierdo.

Electrodos

Modelo	<input type="text"/>	Numero de Serie	<input type="text"/>
Fabricante	<input type="text"/>	Camara	<input type="text"/>
Fecha de Implante	<input type="text"/>	Fecha explante	<input type="text"/>
Deflexion	<input type="text"/>	Amplitud-mV	<input type="text"/>
Intrinsecoide	<input type="text"/> mV/ms	Causa Explante	<input type="text"/>
Umbral	<input type="text"/> V * <input type="text"/> ms	Impedancia-Ohmios	<input type="text"/>

Grabar

Captura 19: Captura de pantalla de los parámetros de electrodo en Sección “Sistemas” del Software.

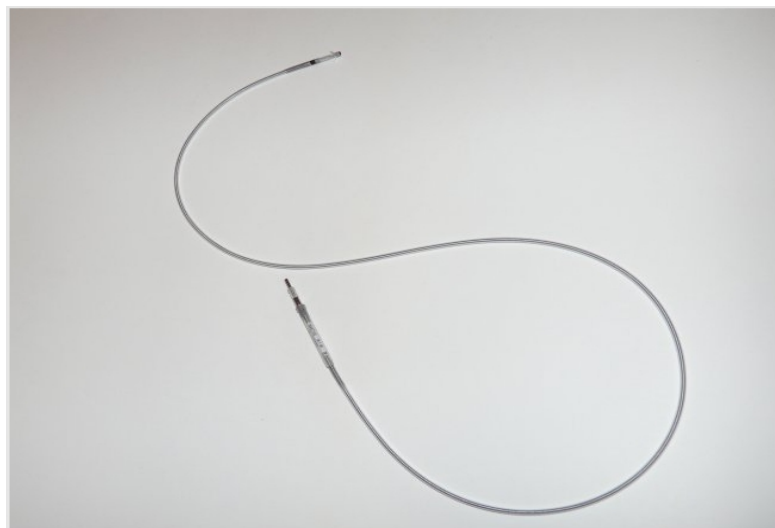


Figura 19: Electrodo ventricular de fijación pasiva [44].

3.5.1.1.4 Información relevante sobre seguimiento de pacientes.

Cuanta más información dispongamos del paciente, de sus patologías, intervenciones, pruebas etc, mejor será para realizar un buen seguimiento de paciente.

En cada visita médica, el cardiólogo realiza un diagnóstico del paciente, anotando distintos datos de dispositivos, comentarios, tratamientos y recomendaciones.

Captura 20: Captura de pantalla de “Nueva visita” en la Sección “Visita” de la aplicación *CardioEstim*.

A su vez, el especialista puede añadirle a esa visita un evento médico, una programación bradicardia, y programación taquicardia, y generar tantos problemas como desee (se añaden a la colección de problemas del paciente).

3.5.1.1.4.1 Evento médico.

Captura 21: Captura de pantalla de formulario de “Evento” en “Visita”.

3.5.1.1.4.2 Programación Bradicardia.

Programacion Bradicardia

Modo Estimulacion: DDD

LIF: 875 lpm.

Tipo Estimulacion Ventricular: LV

Intervalo AV Estimulacion: 78 ms.

Ciclos búsqueda AV: 0

Camara VD: VENTRICULO DERECHO

Sensibilidad VD: 3

Salida VD: 2 V * 2 ms

Config. Estimulacion/Deteccion VD: 4

Grabar

Captura 22: Captura de pantalla de “Programación Bradicardia” dentro del menú “Visita”.

3.5.1.1.4.3 Programación taquicardia.

N.H.C.

999999 - PACIENTE PRUEBA RECETA

DNI: 92920001R T Fecha de Nacimiento: 03/12/1940

Telefonos: 555555555 / 958787847

Sistemas Visita Ficha Paciente Ficha Paciente OutComes Historico Visitas Historico Problemas Consultar Informe Agenda

Programacion Taquicardia

Zona TV-1: 54 lpm.

Zona TV: 53 lpm.

Zona FV: 3 lpm.

ATP TV-1: NO

ATP TV: NO

Num rafagas ATP1 TV-1:

Num rafagas ATP1 TV:

ATP2 TV-1:

ATP2 TV:

Num rafagas ATP2 TV-1:

Num rafagas ATP2 TV:

Descarga TV-1: 4386 J.

Descarga TV: 34 J.

Descarga FV: 34 J.

Descarga 2 TV-1:

Descarga 2 TV:

Descarga 2 FV: 0 J.

Max Energia TV-1:

Max Energia TV:

Max Energia FV: 0 J.

Prorroga TV-1:

Prorroga TV:

Prorroga FV:

Modo Taqui:

Comentario Programacion Taquicardia

Comentario...

Grabar

Captura 23: Captura de pantalla de “Programación Taquicardia” dentro del menú “Visita”.

3.5.1.1.4.4 Nuevo problema.

N.H.C.
 Buscar Limpjar

999999 - **PACIENTE PRUEBA RECETA**
 DNI: 92920001R T Fecha de Nacimiento: 03/12/1940
 Telefonos: 555555555 / 958787847

Sistemas Visita **Ficha Paciente** Ficha Paciente OutComes Historico Visitas Historico Problemas Consultar Informe Agenda

Problemas

Fecha de Comienzo Fecha de Resolucion
 Tipo Problema Grado

Problemas

Grabar

Captura 24: Captura de pantalla de "Nuevo problema" dentro del menú "Visita".

3.5.1.1.4.5 Ficha de paciente.

En esta sección se visualiza los datos del paciente; sus indicaciones, procedimientos, antecedentes e ingresos de éste.

N.H.C.
 Buscar Limpjar

999999 - **PACIENTE PRUEBA RECETA**
 DNI: 92920001R T Fecha de Nacimiento: 03/12/1940
 Telefonos: 555555555 / 958787847

Sistemas Visita **Ficha Paciente** Ficha Paciente OutComes Historico Visitas Historico Problemas Consultar Informe Agenda

Ficha del paciente

INDICACIONES		Nueva indicacion
Fecha Indicacion	Indicacion	
14/01/2010	CARDIOPATIA ISQUEMICA	
30/11/2009	CARDIOPATIA ISQUEMICA	
28/11/2009	MIOCARDIOPATIA DILATADA	
27/11/2009	SI	
27/11/2009	TVS DOCUMENTADA	
27/11/2009	OTRO	
27/11/2009	btbtyt	

PROCEDIMIENTOS			Nuevo Procedimiento
Fecha Procedimiento	Tipo	Diagnostico	
14/01/2010	ECOCARDIOGRAMA	jlo	
11/12/2009	CATERERISMO	fwsfaefwaecccccccccccccccccccccaufcauuuuuuuuuuufwefccccccccccccccccccweifa	
10/12/2009	ECOCARDIOGRAMA	KYGKFK	
10/12/2009	CATERERISMO	UYJ	
26/11/2009	prueba otro	diagnostico...	
26/11/2009	ECOCARDIOGRAMA	bbdg	
25/11/2009	CATERERISMO	HYTDHTH	
19/11/2009	CATERERISMO		
13/11/2009	OTRO NUEVO		
12/11/2009	SPECT		

ANTECEDENTES			Nuevo Antecedente
Fecha Antecedentes	Tipo	Comentario	
14/01/2010	DM1	iluhl	
27/11/2009	HTA	Nada que comentar...	
26/11/2009	DM2	Nada que comentar...	
25/11/2009	DM2	Nada que comentar...	
12/11/2009	FIBRILACION AURICULAR	Comentario... ESO	
12/11/2009	HIPERTIROIDISMO	Comentario...	
11/11/2009	PROBANDO OTROS	Comentario...PROBANDO r2323	
11/11/2009	DM2	Comentario...11	
10/11/2009	DM1	Comentario...dm1	
10/11/2009	HTA	Comentario...gtrhrd	
10/11/2009	TRASPLANTE RENAL	Comentario...ccccccccex	
10/11/2009	FIBRILACION AURICULAR	Comentario...dxwic	
10/11/2009		Comentario... VNN V	
02/11/2009	FUMADOR	Comentario...fuma	

INGRESOS			Nuevo Ingreso
Fecha Ingreso	Servicio	Comentario	
14/01/2010	ACV	olloi	
11/12/2009	CG1		
27/11/2009	CG1	Nada que comentar...	
26/11/2009	CG2	Nada que comentar...	
25/11/2009	CCV	Nada que comentar...	
13/11/2009	CCV	Comentario...11	
11/11/2009		Comentario...PRUEBA INGRESO	
11/11/2009		Comentario...nuevo	

Captura 25: Captura de pantalla de pestaña "Ficha Paciente".

En algunos dispositivos implantados existe la posibilidad que éste envíe información sobre su estado, o algunos valores de interés vía (sms) a un servidor de Londres, los cuales finalmente se registran en la aplicación de seguimiento de paciente.

Actualmente este servicio está en desuso ya que no se implantan estos dispositivos en los pacientes debido a su elevado coste. Sin embargo, se implementa esta función en el nuevo Software de Cardiología con vistas a futuro y por ser de gran interés.

Ficha del paciente

Peso	80.7	kg.	B.S.A.	1.983	
Estatura	175.55	cm.	I.M.C.	26.18	Sobrepeso

Electrocardiograma Basal

Ritmo	vitr	QRS	67	ms.
Cicatriz	SI	Localizacion	LATERAL	

Dispositivos Previos

MCPPrevio	SI	MCPDependiente	SI
DAIPrevio	SI	Terapias Previas	SI

Outcomes

Fecha de Exitus	11/11/2009	Causa Exitus CV	SI	Causa Exitus	vitr
Respondedor	SI	Remodelador	SI		
VV-ms	7				

Alergias

aerferse

Comentario

ireffra

Captura 26: Captura de pantalla de la sección "Ficha de Paciente".

3.5.1.1.4.6 Histórico de Visitas.

N.H.C.

Buscar **Limpiar**

999999 - **PACIENTE PRUEBA RECETA**
 DNI: **92920001R T** Fecha de Nacimiento: **03/12/1940**
 Telefonos: **555555555 / 958787847**

Sistemas	Visita	Ficha Paciente	Ficha Paciente OutComes	Historico Visitas	Historico Problemas	Consultar Informe	Agenda
----------	--------	----------------	-------------------------	-------------------	---------------------	-------------------	--------

Historico Visitas

Historico de Visitas				
Fecha Grabacion	Tipo Visita	Dispositivo	Situacion Bateria	Evento
03/10/2012		35234		
12/07/2010		35234		SI
28/01/2010		231		SI
26/01/2010	PRE ALTA	231	MOL1	
25/01/2010	REGULAR	231	MOL2	NO
19/01/2010	EXTRAORDINARIA	231	BOL	SI
18/01/2010				
14/01/2010	EXTRAORDINARIA	453513	BOL	SI
13/01/2010	REGULAR	453513	ERI	SI
11/01/2010	REGULAR	34354533	BOL	SI
05/01/2010	REGULAR	34354533	MOL1	SI
29/12/2009	REGULAR	34354533	BOL	SI
28/12/2009	REGULAR	34354533	BOL	SI
11/12/2009	EXTRAORDINARIA	34354533	MOL2	SI
09/12/2009	REGULAR	787878	BOL	SI
30/11/2009	EXTRAORDINARIA	787878	MOL1	NO
27/11/2009		555555		NO
26/11/2009	REGULAR	45454	MOL2	SI
23/11/2009	REGULAR	134341	BOL	
20/11/2009	REGULAR	454545	BOL	
19/11/2009	REGULAR	3122243	BOL	
18/11/2009	REGULAR	11111111	MOL1	
13/11/2009	EXTRAORDINARIA	11111111	BOL	
11/11/2009	REGULAR	2121214	BOL	
10/11/2009	EXTRAORDINARIA	555555555	EOL	
06/11/2009	REGULAR	555555555	MOL1	
05/11/2009	EXTRAORDINARIA	555555555	MOL1	
04/11/2009	REGULAR	555555555	BOL	
03/11/2009	EXTRAORDINARIA	121212	MOL2	
02/11/2009	REGULAR	898989	BOL	
30/10/2009		2444442	MOL1	
30/10/2009	REGULAR	2444442	EOL	
29/10/2009	EXTRAORDINARIA	2444442	MOL1	
28/10/2009	REGULAR	11111111		

Captura 27: Captura de pantalla de la sección “Histórico de visitas”.

3.5.1.1.4.7 Histórico de problemas.

Se visualizan todos los problemas cardiológicos registrados y asociados al paciente, tanto los resueltos como los activos.

Historico Problemas

Historico de Problemas			
Fecha Comienzo	Problemas Activos	Grado	Fecha Resolucion
14/01/2010	INFECCION BOLSAELECTRODOS	LEVE	---
14/01/2010	DETERIORO CLINICO	MODERADO	---
14/01/2010	DETERIORO CLINICO	MODERADO	---
06/01/2010	DETERIORO CLINICO	MODERADO	14/01/2010
27/11/2009	ARRITMIAS FRECUENTES	LEVE	14/01/2010
27/11/2009	RECALL	MODERADO	27/11/2009
27/11/2009	INFECCION BOLSAELECTRODOS	MODERADO	27/11/2009
27/11/2009	INFECCION BOLSAELECTRODOS	LEVE	27/11/2009
27/11/2009	INFECCION BOLSAELECTRODOS	MODERADO	27/11/2009
27/11/2009	ener	GRAVE	27/11/2009
27/11/2009	ERI	GRAVE	27/11/2009
27/11/2009	ARRITMIAS FRECUENTES	LEVE	27/11/2009
27/11/2009	ARRITMIAS FRECUENTES		---
26/11/2009	ARRITMIAS FRECUENTES	MODERADO	27/11/2009
19/11/2009	ERI	MODERADO	23/11/2009
12/11/2009	HOLA	LEVE	27/11/2009
11/11/2009	Prueba otro	GRAVE	14/01/2010
09/11/2009	DETERIORO CLINICO	LEVE	11/11/2009
05/11/2009	ERI	GRAVE	---
12/11/2009	DETERIORO CLINICO	LEVE	---
04/11/2009	INFECCION BOLSAELECTRODOS	MODERADO	---
04/11/2009		LEVE	04/11/2009
04/11/2009		GRAVE	04/11/2009
21/10/2009		LEVE	22/10/2009
21/10/2009		GRAVE	28/10/2009
21/10/2009		GRAVE	04/11/2009
21/10/2009		LEVE	04/11/2009

Captura 28: Captura de pantalla de la sección “Histórico de problemas”.

3.5.1.1.4.8 Histórico de Informes.

Se visualizan todos los informes cardiológicos registrados y asociados al paciente.

Informe			
Historico de Informes			
Fecha Grabacion	Tipo Visita	Dispositivo	Situacion Bateria
03/10/2012		35234	
12/07/2010		35234	
28/01/2010		231	
28/01/2010	PRE ALTA	231	MOL1
25/01/2010	REGULAR	231	MOL2
19/01/2010	EXTRAORDINARIA	231	BOL
18/01/2010			
14/01/2010	EXTRAORDINARIA	463513	BOL
13/01/2010	REGULAR	463513	ERI
11/01/2010	REGULAR	34354633	BOL
05/01/2010	REGULAR	34354633	MOL1
29/12/2009	REGULAR	34054633	BOL
28/12/2009	REGULAR	34354633	BOL
11/12/2009	EXTRAORDINARIA	34354633	MOL2
09/12/2009	REGULAR	787878	BOL
30/11/2009	EXTRAORDINARIA	787878	MOL1
27/11/2009		666666	
26/11/2009	REGULAR	46464	MOL2
23/11/2009	REGULAR	134341	BOL
20/11/2009	REGULAR	464646	BOL
19/11/2009	REGULAR	3122243	BOL
18/11/2009	REGULAR	11111111	MOL1
13/11/2009	EXTRAORDINARIA	11111111	BOL
11/11/2009	REGULAR	2121214	BOL
10/11/2009	EXTRAORDINARIA	6666666666	EOL
06/11/2009	REGULAR	6666666666	MOL1
05/11/2009	EXTRAORDINARIA	6666666666	MOL1
04/11/2009	REGULAR	6666666666	BOL

Captura 29: Captura de pantalla de la Sección “Consultar Informe”.

3.5.1.1.4.9 Agenda del paciente.

Agenda	
Fecha de Visita	<input type="text"/>
Tipo Agenda	<input type="text"/>
Ver	

Captura 30: Captura de pantalla del paciente.

3.5.1.1.5 Información relevante sobre tipos de informes médicos que se generan con la aplicación.

Gracias a la aplicación *Cardioweb* se emiten informes de pruebas cardiológicas.

3.5.1.1.6 Bases de Datos Caché.

A continuación se facilita parte de la estructura de las tablas de las Bases de Datos de *CardioEstim* y *CardioWeb* anteriores al nuevo desarrollo software:

3.5.1.1.6.1 Tablas de CardioWeb (c2hn.card.InfCardioFCK).

Properties

- property **CodPers** as %String(MAXLEN=6,TRUNCATE=1);
25.11.2008 Se añade propiedad CodPers
- property **MedicoAsig** as %String(MAXLEN=40,TRUNCATE=1);
- property **NumeroHC** as %Integer;
- property **TextoInforme** as %String(MAXLEN=10000,TRUNCATE=1);
- property **TiposInforme** as %String(MAXLEN=20,TRUNCATE=1);
- property **episodio** as %String(TRUNCATE=1);
- property **fechaBajaReg** as %Date(FORMAT=4);
- property **fechaInforme** as %Date(FORMAT=4);
- property **horaInforme** as %String(MAXLEN=5,TRUNCATE=1);
- property **userid** as %String(MAXLEN=10,TRUNCATE=1);

Índices.

- index (**idxNHC** on **NumeroHC**);

c2hn.card.Ergometria

- property **Episodio** as %Integer;
- property **FCMax** as %String(MAXLEN=4,TRUNCATE=1);
- property **FCbasal** as %String(MAXLEN=4,TRUNCATE=1);
- property **FechaBajaReg** as %Date(FORMAT=4);
- property **FechaInforme** as %Date(FORMAT=4);

- property **HoraGrabacion** as %String(MAXLEN=5,TRUNCATE=1);
- property **METS** as %String(MAXLEN=15,TRUNCATE=1);
- property **MedicoAsig** as %String(MAXLEN=50,TRUNCATE=1);
- property **NumeroHC** as %Integer;
- property **Porcentaje** as %String(MAXLEN=2,TRUNCATE=1);
- property **Protocolo** as %String(MAXLEN=12,TRUNCATE=1);
- property **Resultado** as %String(MAXLEN=15,TRUNCATE=1);
- property **RitmoBasal** as %String(MAXLEN=16,TRUNCATE=1);
- property **TAMax** as %String(MAXLEN=4,TRUNCATE=1);
- property **TAbasal** as %String(MAXLEN=4,TRUNCATE=1);
- property **TextoLibre** as %String(MAXLEN=1000,TRUNCATE=1);
- property **TiempoEjMin** as %String(MAXLEN=4,TRUNCATE=1);
- property **TiempoEjSeg** as %String(MAXLEN=4,TRUNCATE=1);
- property **Userid** as %String(MAXLEN=10,TRUNCATE=1);

Indices

- index (**idxNHC** on **NumeroHC**);

c2hn.card.InfoCardio

- property **AI** as %String(MAXLEN=10,TRUNCATE=1);

propiedades para PROTOCOLO DE HIPERTENSION PULMONAR

- property **AId** as %String(MAXLEN=10,TRUNCATE=1);
- property **AoGM** as %String(MAXLEN=10,TRUNCATE=1);
- property **AoGP** as %String(MAXLEN=10,TRUNCATE=1);
- property **AoTHP** as %String(MAXLEN=10,TRUNCATE=1);
- property **AreaAD** as %String(MAXLEN=10,TRUNCATE=1);
- property **AreaVD** as %String(MAXLEN=10,TRUNCATE=1);
- property **Arteria** as %String(MAXLEN=10,TRUNCATE=1);
- property **D1** as %String(MAXLEN=10,TRUNCATE=1);
- property **D2** as %String(MAXLEN=10,TRUNCATE=1);

- property **DTDVD** as %String(MAXLEN=10,TRUNCATE=1);
- property **DTDVI** as %String(MAXLEN=10,TRUNCATE=1);
- property **DTSVI** as %String(MAXLEN=10,TRUNCATE=1);
- property **DVIId** as %String(MAXLEN=10,TRUNCATE=1);
- property **DVIs** as %String(MAXLEN=10,TRUNCATE=1);
- property **ETIV** as %String(MAXLEN=10,TRUNCATE=1);
- property **FE** as %String(MAXLEN=10,TRUNCATE=1);
- property **FrecCardia** as %String(MAXLEN=10,TRUNCATE=1);
- property **G1** as %String(MAXLEN=10,TRUNCATE=1);
- property **G2** as %String(MAXLEN=10,TRUNCATE=1);
- property **G2P** as %String(MAXLEN=10,TRUNCATE=1);
- property **GM** as %String(MAXLEN=10,TRUNCATE=1);
- property **GP** as %String(MAXLEN=10,TRUNCATE=1);
- property **MedicoAsig** as %String(MAXLEN=50,TRUNCATE=1);
- property **MitGM** as %String(MAXLEN=10,TRUNCATE=1);
- property **NúmeroHC** as %Integer;
- property **OndaA** as %String(MAXLEN=10,TRUNCATE=1);
- property **OndaE** as %String(MAXLEN=10,TRUNCATE=1);
- property **PP** as %String(MAXLEN=10,TRUNCATE=1);
- property **Paps** as %String(MAXLEN=10,TRUNCATE=1);

- property **Raiz** as %String(MAXLEN=10,TRUNCATE=1);
- property **SIV** as %String(MAXLEN=10,TRUNCATE=1);
- property **T1** as %String(MAXLEN=10,TRUNCATE=1);
- property **T2** as %String(MAXLEN=10,TRUNCATE=1);
- property **TACP** as %String(MAXLEN=10,TRUNCATE=1);
- property **TAPSE** as %String(MAXLEN=10,TRUNCATE=1);
- property **TIV** as %String(MAXLEN=10,TRUNCATE=1);
- property **Teicholtz** as %String(MAXLEN=10,TRUNCATE=1);
- property **Tension** as %String(MAXLEN=10,TRUNCATE=1);
- property **VDd** as %String(MAXLEN=10,TRUNCATE=1);

propiedades para PROTOCOLO E.T.T.

- property **VelA** as %String(MAXLEN=10,TRUNCATE=1);
- property **VelE** as %String(MAXLEN=10,TRUNCATE=1);
- property **aprecMovAsSiv** as %String(MAXLEN=10,TRUNCATE=1);

- property **dP** as %String(MAXLEN=10,TRUNCATE=1);
- property **dPdt** as %String(MAXLEN=10,TRUNCATE=1);
- property **episodio** as %Integer;
- property **excentrVI** as %String(MAXLEN=10,TRUNCATE=1);
- property **fechaBajaReg** as %Date(FORMAT=4);
- property **fechaInforme** as %Date(FORMAT=4);
- property **finAcort** as %String(MAXLEN=10,TRUNCATE=1);
- property **horalInforme** as %String(MAXLEN=5,TRUNCATE=1);
- property **insufMitral** as %String(MAXLEN=10,TRUNCATE=1);
- property **insufMitralPresist** as %String(MAXLEN=10,TRUNCATE=1);
- property **movSivDti** as %String(MAXLEN=10,TRUNCATE=1);
- property **retrIntervPlm** as %String(MAXLEN=10,TRUNCATE=1);
- property **retrInteventr** as %String(MAXLEN=10,TRUNCATE=1);

Propiedades para CARDIAC RESYNCHRONIZATION ECHOCARDIOGRAPY SCORE (CRES)

- property **retrSivPI** as %String(MAXLEN=10,TRUNCATE=1);
- property **tiempLlenDiast** as %String(MAXLEN=10,TRUNCATE=1);
- property **tiempPreeyec** as %String(MAXLEN=10,TRUNCATE=1);
- property **userid** as %String(MAXLEN=10,TRUNCATE=1);

3.5.1.1.6.2 Tablas de *CardioEstim* (Package: c2hn.CardioInf).

c2hn.CardioInf.AntecedentesP

Properties

- property **Comentario** as %String(MAXLEN=500);
- property **Comentariol** as %String(MAXLEN=500);
- property **Episodio** as %Integer;
- property **Fecha** as %Date(FORMAT=4);
- property **FechaI** as %Date(FORMAT=4);
- property **FechaIn** as %Date(FORMAT=4);
- property **Indicacion** as %String;
- property **NumeroHC** as %Integer;

- property **Serviciol** as %String(MAXLEN=50);
- property **Tipo** as %String(MAXLEN=50);
- property **fechaBajaReg** as %Date(FORMAT=4);
- property **uid** as %String(MAXLEN=9);

c2hn.CardioInf.ElectroVisita

- property **AmplitudMV** as %String(MAXLEN=50);
- property **Camara** as %String(MAXLEN=50);
- property **Comentario** as %String(MAXLEN=250);
- property **Episodio** as %Integer;
- property **FechaGrabacion** as %Date(FORMAT=4);
- property **ImpedanciaOhmios** as %String(MAXLEN=50);
- property **NumeroHC** as %Integer;
- property **NumeroSerie** as %String(MAXLEN=50);
- property **UmbralMS** as %String(MAXLEN=50);
- property **UmbralV** as %String(MAXLEN=50);
- property **fechaBajaReg** as %Date(FORMAT=4);
- property **uid** as %String(MAXLEN=9);

c2hn.CardioInf.Problemas

- property **Episodio** as %Integer;
- property **FechaC** as %Date(FORMAT=4);
- property **FechaR** as %Date(FORMAT=4);
- property **Grado** as %String(MAXLEN=15);
- property **NumeroHC** as %Integer;
- property **Problemas** as %String(MAXLEN=250);
- property **TipoProblema** as %String(MAXLEN=50);
- property **fechaBajaReg** as %Date(FORMAT=4);
- property **uid** as %String(MAXLEN=9);

3.5.2 Tarea PSI 3.2: Valoración de Antecedentes.

En esta tarea se realiza la valoración de los antecedentes analizados en la tarea anterior y las conclusiones se recogen en el catálogo de requisitos. La realización de esta valoración ayuda a establecer términos de referencia en cuanto a estándares, procedimientos, mejoras del actual software, normativas, etc.

Productos de la Tarea PSI 3.2.

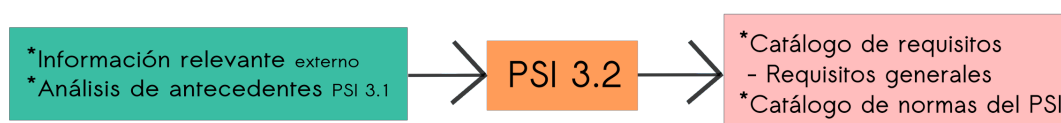


Figura 20: Productos de entrada y salida de la Tarea PSI 3.2.

3.5.2.1 Requisitos generales.

Aquí identificamos a grandes rasgos los requisitos generales, para, en el siguiente apartado, poder realizar una descripción completa del comportamiento del software desarrollado:

- Cumplir las funcionalidades principales de pruebas médicas y seguimiento de paciente.
- Añadir nuevas funcionalidades que ayuden al especialista con el seguimiento de paciente de manera gráfica.
- Mejorar el modelo entidad/relación de los datos proponiendo una más intuitivo.
- Mayor amigabilidad de la interfaz.
- Integración con las diferentes plataformas del HUGCDN, tanto de los sistemas de información como de los equipos electromédicos usados por los cardiólogos.

3.5.2.2 Catálogo de normas del PSI.

Actualmente el Hospital HUGCDN pretende certificar sus procesos mediante la norma ITIL, y es por ello que se persigue para esta nueva aplicación que se cumplan sus requerimientos.

Además, se propone seguir cumpliendo con el conjunto de estándares usados por el

Hospital; HL7 e ISO 27000.

3.6 Identificación de requisitos.

El objetivo final de esta actividad va a ser la especificación de los requisitos del software médico, así como obtener un modelo de información que los complemente.

Para conseguir este objetivo, se estudia el proceso o procesos de la organización incluídos en el ámbito del Plan de Sistemas de Información. Para ello es necesario llevar a cabo sesiones de trabajo con los usuarios, analizando cada proceso tal y como debería ser, y no según su situación actual, ya que ésta puede estar condicionada por los sistemas de información existentes.

Del mismo modo, se identifican los requisitos de información, y se elabora un modelo de información que represente las distintas entidades implicadas en el proceso, así como las relaciones entre ellas. Por último, se clasifican los requisitos identificados según su prioridad, con el objetivo de incorporarlos al catálogo de requisitos del Plan de Sistemas de Información.

TAREA	PRODUCTOS	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
PSI 4.1 Estudio de los Procesos del PSI.	Modelo de procesos de la organización.	Modelado de Procesos de la Organización. Sesiones de trabajo.	Perfil Consultor. Perfil Directivo. Perfil Colaborador.
PSI 4.2 Análisis de las Necesidades de la Información.	Necesidades de información. Modelo de información.	Modelo Entidad/Relación extendido. Diagrama de clases. Sesiones de trabajo.	Perfil Consultor. Perfil Directivo. Perfil Colaborador.
PSI 4.3 Catalogación de Requisitos.	Catálogo de requisitos: <ul style="list-style-type: none">Requisitos de los procesos afectados por el PSI	Catalogación.	Perfil Consultor. Perfil Directivo. Perfil Colaborador.

Tabla 7: Tareas para la identificación de requisitos.

3.6.1 Tarea PSI 4.1: Estudio de los Procesos del PSI.

En esta tarea se estudia cada proceso de la organización incluido en el ámbito del Plan de Sistemas de Información. Para cada uno de ellos, es necesario identificar las actividades o funciones, la información comprometida en ellas y las unidades organizativas; usuarios y recursos, que participan en el desarrollo de cada actividad. Para obtener esta información se llevan a cabo sesiones de trabajo con los usuarios

implicados en cada uno de los procesos a analizar; cardiólogos e ingenieros de informática. Una vez contrastadas las conclusiones, se elabora el modelo correspondiente a cada proceso. Si existe relación entre los distintos modelos, se unifican en la medida de lo posible, con el fin de proporcionar una visión global en el contexto de la organización y facilitar una identificación de requisitos más objetiva.

Productos de la Tarea PSI 4.1.

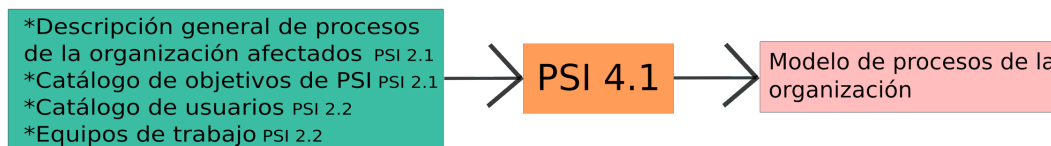


Figura 21: Productos de entrada y salida de la Tarea PSI 4.1.

3.6.1.1 Modelado de procesos de la organización [45].

Los procesos de la organización afectados son:

Proceso principal. Constituido por las consultas de pruebas y visitas médicas programadas de cardiología. En ellas se ve implicada parte de la información de la historia clínica del paciente alojada por el Servicio Canario de Salud (Drago Web) [46], fundamental para la adecuada asistencia al paciente. También se accede a la información cardiológica de cada paciente, que se encuentra en la base de datos dedicada a esta especialidad. La unidad organizativa implicada en este proceso lo forman los especialistas, los equipos locales o remotos de trabajo, y según el caso, los equipos médicos utilizados en la consulta.

Proceso de soporte: resolución de incidencias (ver evento 1). Este proceso está dedicado a la resolución de incidencias del software para cardiólogos. En este proceso se ve implicada la misma información anterior junto al parte de incidencia a resolver. Los usuarios implicados en este proceso son los facultativos médicos que, o bien detectan un problema, o simplemente lanzan una petición y; también, los ingenieros encargados de resolver este tipo de cuestiones.

Proceso de soporte: aplicación de protocolos (ver evento 2). Aplicación de los protocolos de seguridad y calidad usados actualmente por el Hospital. Aquí se engloban

la información y la unidad organizativa de los anteriores procesos citados.

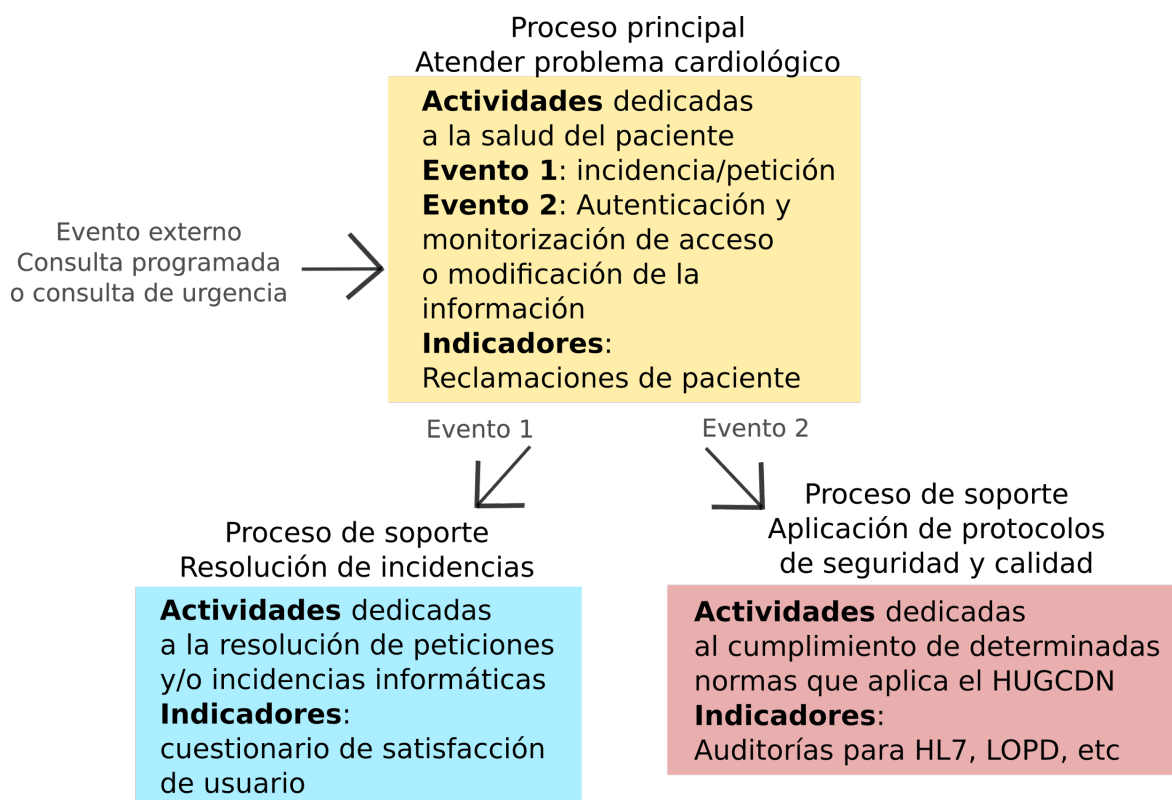


Figura 22: Modelo de procesos de la organización.

3.6.1.2 Unidades Organizativas.

Debido al gran número de equipos a utilizar por el Hospital, además de tener equipos físicos de acceso local, también se dispone de una infraestructura virtualizada por medio de varios servidores con el sistema operativo para servidores Windows Server 2008. El acceso de cada usuario de cardiología a su escritorio de trabajo se realiza mediante una sesión remota.

En cuanto a la implementación del servicio de directorio para la red distribuida de ordenadores del hospital nos referimos al término *Active Directory* [47]. Este protocolo viene implementado de forma similar a una base de datos, la cual almacena en forma centralizada toda la información relativa al dominio de autenticación; *CanariaSalud*. Una de sus ventajas es la sincronización presente entre los distintos servidores de autenticación de todo el dominio. En estos servidores se crean objetos tales como usuarios, equipos o grupos, con el objetivo de administrar los inicios de sesión en los equipos conectados a la red, así como también la administración de políticas de ésta. Así facilitamos el control, la administración y la consulta de todos los elementos lógicos de

una red (como pueden ser usuarios, equipos y recursos).

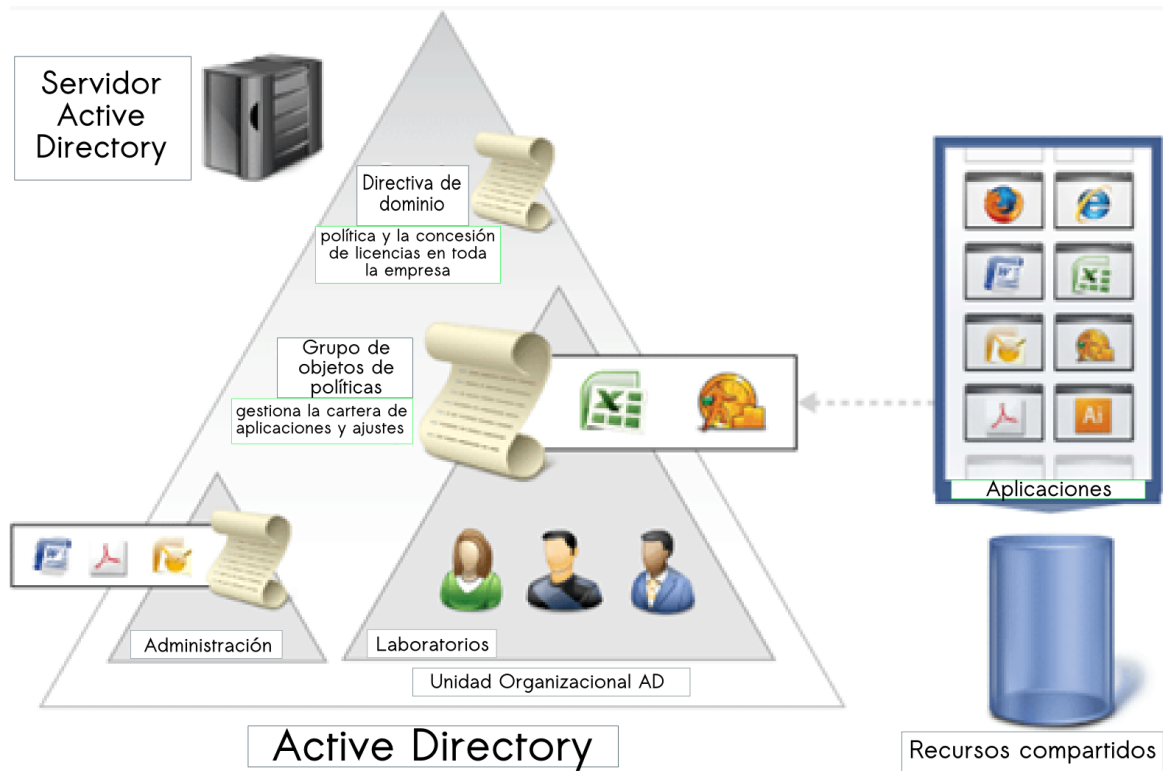


Figura 23: Ejemplo sobre Active Directory de Windows Server 2008.

Tras autenticarse en el dominio *CanariaSalud*, los usuarios trabajan con un escritorio Windows 7, y acceden a través del Navegador web Internet Explorer 8 tras identificarse en la Intranet del Hospital, a las principales herramientas de trabajo; las aplicaciones independientes web *CardioEstim* y *CardioWeb*. Estas aplicaciones acceden a dos bases de Datos *Caché*.



Captura 31: Captura de pantalla Acceso a la Intranet del Hospital.



Captura 32: Captura de pantalla Acceso a los servicios de la Intranet del Hospital.



Captura 33: Captura de los servicios activados para la alumna de la Intranet del Hospital.

A continuación se muestra un gráfico de este entorno de trabajo:

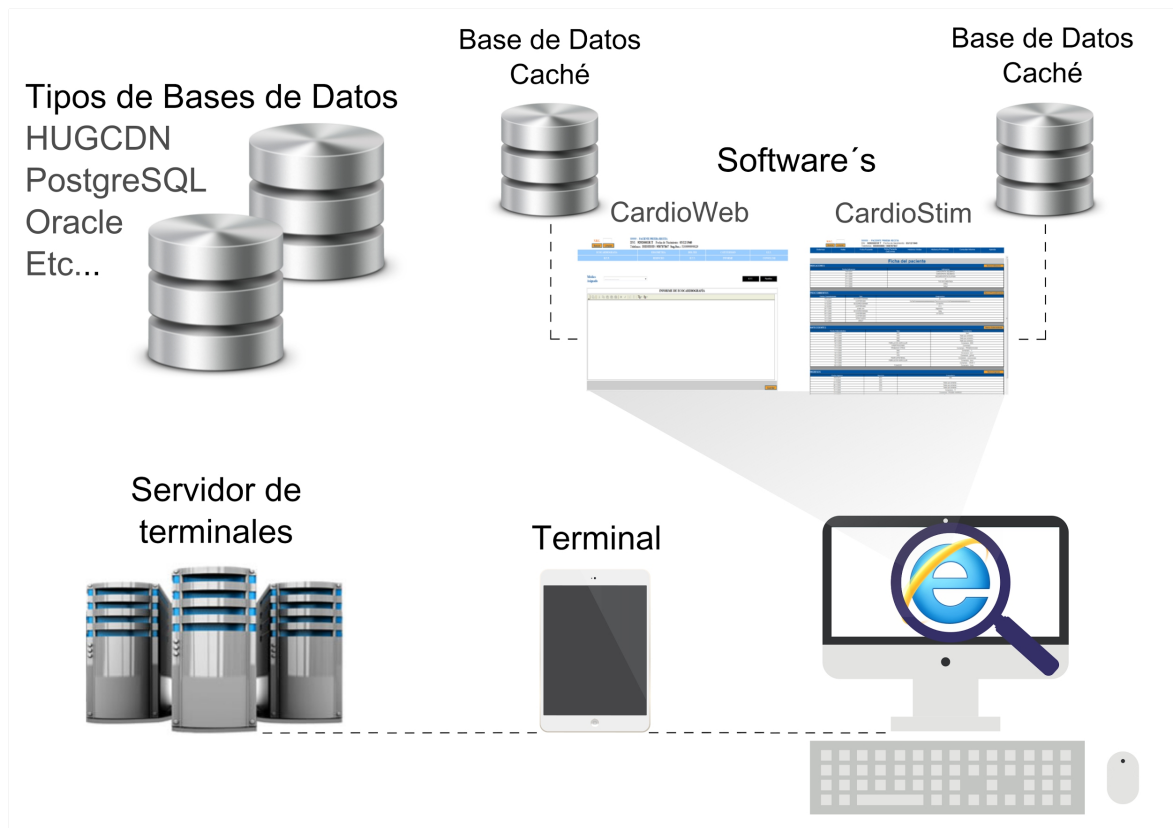


Figura 24: Acceso remoto, autenticación en el dominio Canaria Salud, y acceso a las aplicaciones de la Intranet *Cardioweb* y *Cardioestim*.

Si el usuario navega fuera de la Intranet, el Gobierno de Canarias pone a su disposición un portal de acceso a internet que le pedirá usuario y contraseña para poder acceder a URL's externas, para la autenticación y monitorización de esta navegación por políticas de seguridad.

Principal G.D.I. Contacto

User

Password

Login

Está usted accediendo al servicio de acceso a Internet del Gobierno de Canarias. Para poder acceder a dicho servicio es preciso que se valide con sus credenciales actuales de [Gestión de Identidades](#). Si tiene alguna incidencia o consulta de carácter técnico relacionada con la prestación de este servicio, por favor, contacte con CiberCentro enviando un correo a cibercentro@gobiernodecanarias.org, o llamando al teléfono 912 (dentro de la Red Ibercom).

© Gobierno de Canarias Supervisión y Encuestas Aviso Legal

Captura 34: Captura de pantalla de la pasarela de autenticación del Gobierno de Canarias.

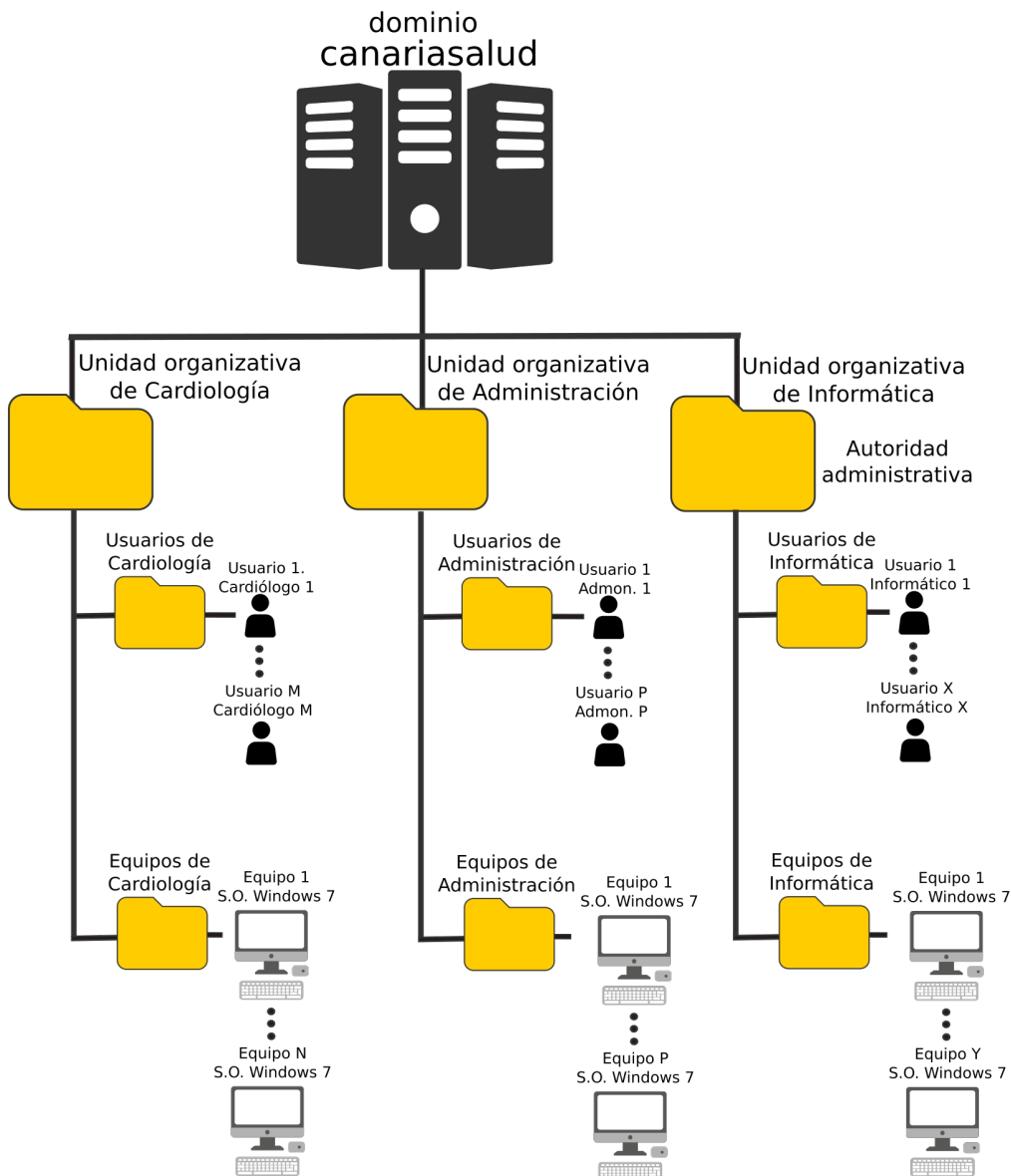


Figura 25: Gráfico de agrupación de objetos que tienen mismos permisos, mismas políticas de seguridad, etc.

3.6.2 Tarea PSI 4.2: Análisis de las Necesidades de Información.

En esta tarea, mediante sesiones de trabajo, se identificó las necesidades de información de cada uno de los procesos analizados en la actividad anterior. Además, se realizó un modelo de información que refleja las principales entidades y relaciones existentes entre ellas.

Los resultados del análisis realizado en esta tarea son la base para la identificación de requisitos.

Productos de la Tarea PSI 4.2.



Figura 26: Productos de entrada y salida de la Tarea PSI 4.2.

3.6.2.1 Necesidades de información.

La información que se necesitaron para el desarrollo de esta aplicación, la enumeramos a continuación:

- Historia clínica del paciente.
- Información cardiológica del paciente.
- Información de la unidad organizativa del HUGCDN.
- Información de los usuarios de la aplicación.
- Información para el servicio de informática: para la resolución de incidencias, estadísticas, etc.

3.6.2.2 Modelo Entidad / Relación extendido.

Gracias a este modelo podremos visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenimiento.

Un diagrama de clases está compuesto por los siguientes elementos:

3.6.2.2.1 Clase.

Es la unidad básica que encapsula toda la información de un objeto (instancia de una clase). Ésta es representada por un rectángulo que posee tres divisiones:

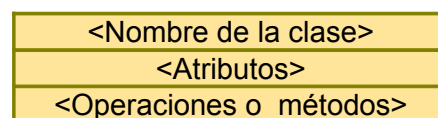


Figura 27: Representación de una clase.

En donde:

Atributos: son características de la clase (variables de instancia) que la caracterizan.

Métodos: son las operaciones de una clase y definen como ésta interactúa con su entorno.

Visibilidad: según el tipo de los atributos y métodos, public, private o protected se definen el grado de comunicación y visibilidad de ellos con el entorno:

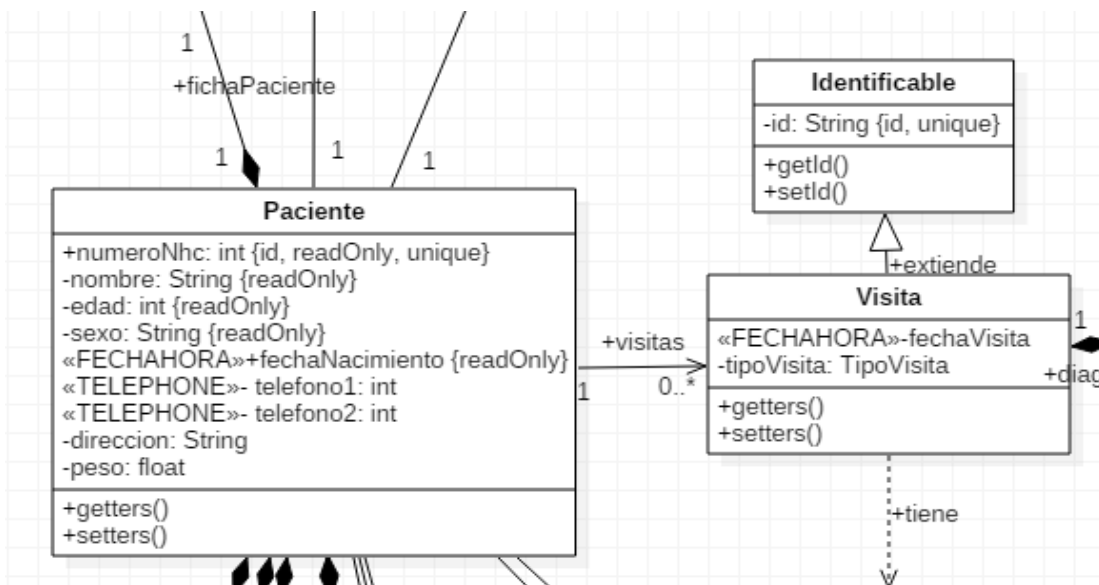
1. **public (+):** Indica que será visible tanto dentro como fuera de la clase, es decir, es accesible desde cualquier lugar.
2. **private (-):** Indica que sólo será accesible desde dentro de la clase (sólo otros métodos de la clase pueden acceder a él).
3. **protected (#):** Indica que no será accesible desde fuera de la clase, pero sí podrá ser accesado por métodos de la clase además de métodos de las subclases que se deriven.

3.6.2.2 Relaciones entre Clases.

Explica cómo se pueden relacionar dos o más clases.

Cardinalidad de las relaciones: indica el grado y nivel de dependencia, se anotan en cada extremo de la relación y éstas pueden ser:

1. **uno a muchos:** 1..* (1..n)
2. **0 a muchos:** 0..* (0..n)
3. **número fijo:** m (m denota el número)



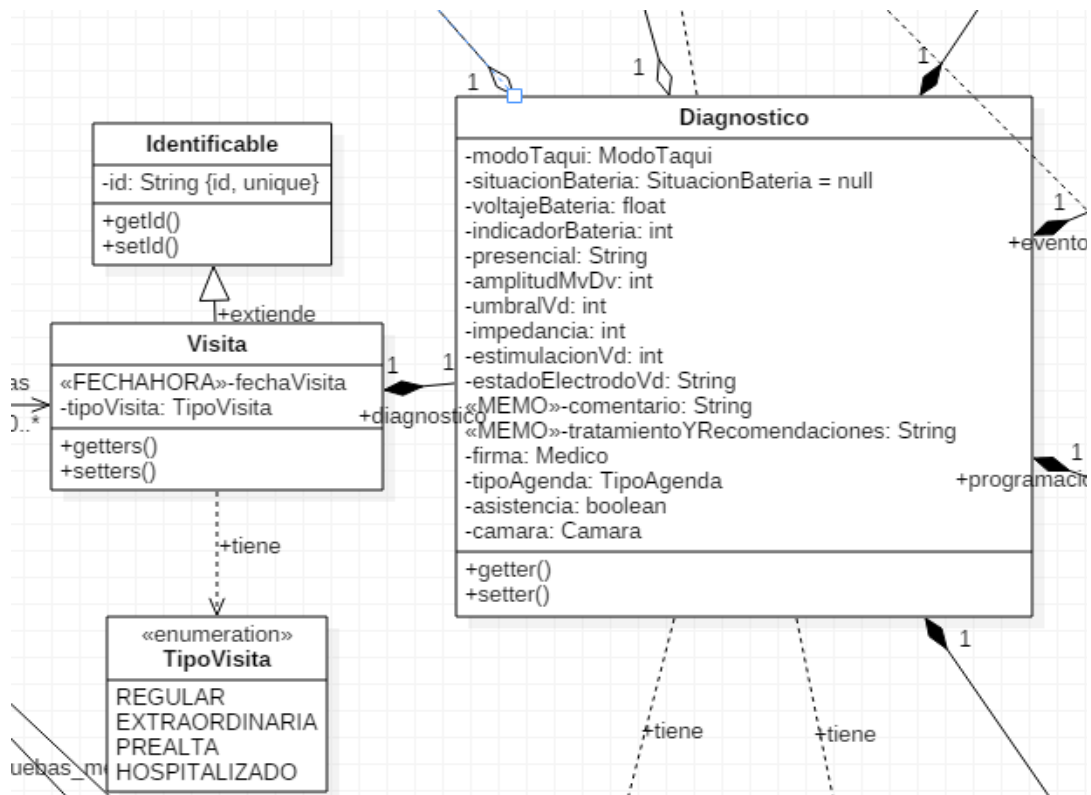
Captura 35: Fragmento del diagrama UML donde se refleja que un paciente tiene 0, o muchas visitas médicas al servicio de Cardiología.

Herencia (Especialización/Generalización). →

Indica que una subclase (Visita) hereda los métodos y atributos especificados por una Super Clase (Identificable), por ende la Subclase además de poseer sus propios atributos (fechaVisita y tipoVisita) y métodos, poseerá las características y atributos visibles de la Super Clase (public y protected).

Agregación. ◊ →

1. **Composición:** Se representa como un rombo relleno. Es una relación estática, en donde el tiempo de vida del objeto incluido está condicionado por el tiempo de vida del que lo incluye.
2. **Agregación:** Se representa como un rombo en blanco. Es una relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Los rombos que representan a ambas relaciones van en el objeto que posee las referencias:



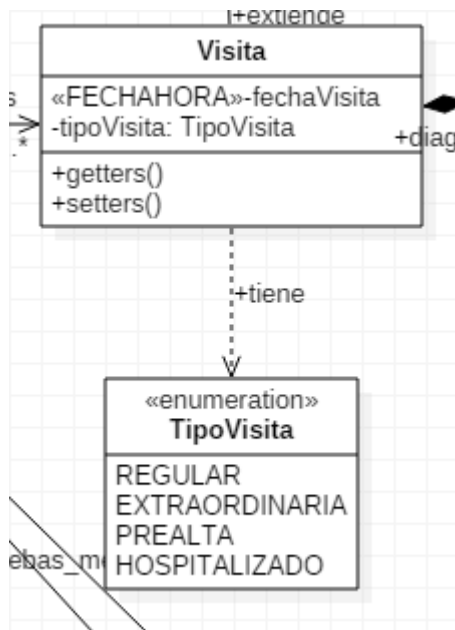
Captura 36: Fragmento del diagrama UML donde se aprecia que la clase Visita tiene Diagnóstico. Si se elimina Visita, se elimina Diagnóstico.

Asociación. →

Permite asociar objetos que colaboran entre sí. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.

Dependencia o Instanciación (uso). -.->

Denota la dependencia que tiene una clase de otra.



Captura 37: Fragmento del diagrama UML donde se representa que *tipoVisita* es enumerado.

3.6.2.2.3 Diagrama de clases.

Para realizar el diagrama de clases hemos utilizado el programa StarUML [48]. En nuestro caso sólo representamos las clases y sus relaciones sin entrar en detalle de los atributos y métodos para una mejor visualización del diagrama. Más adelante se facilitarán las tablas de todas las clases descritas.

El diagrama de clases se muestra a continuación:

3.6.3 Tarea PSI 4.3: Catalogación de Requisitos.

En esta tarea se analiza la información recogida en las tareas anteriores de esta actividad. Se definen los requisitos detallados que se incorporan al catálogo de requisitos generales y se les asignan prioridades. Los criterios para asignar dichas prioridades deben ser definidos al comienzo de esta tarea, considerando la opinión de los usuarios sobre los procesos de la organización, así como los objetivos del Plan de Sistemas de Información.

Productos de la Tarea PSI 4.3.

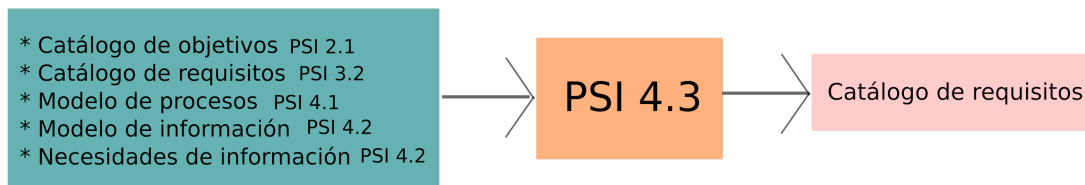


Figura 28: Productos de entrada y salida de la Tarea PSI 4.3.

3.6.3.1 Requisitos de los procesos afectados por el PSI.

Las características de una buena especificación de requisitos de software (ERS) [49] son definidas por el estándar IEEE 830-1998 [50].

- Completa. Todos los requerimientos deben estar reflejados en ella y todas las referencias deben estar definidas.
- Consistente. Debe ser coherente con los propios requerimientos y también con otros documentos de especificación.
- Inequívoca. La redacción debe ser clara de modo que no se pueda malinterpretar.
- Correcta. El software debe cumplir con los requisitos de la especificación.
- Trazable. Se refiere a la posibilidad de verificar la historia, ubicación o aplicación de un ítem a través de su identificación almacenada y documentada.
- Priorizable. Los requisitos deben poder organizarse jerárquicamente según su relevancia para el negocio y clasificándolos en esenciales, condicionales y opcionales.
- Modificable. Aunque todo requerimiento es modificable, se refiere a que debe ser fácilmente modificable.
- Verificable. Debe existir un método finito sin costo para poder probarlo.

3.6.3.2 Tipos de requisitos.

Existen varios tipos de requisitos como lo son:

- Requisitos de Usuarios: Necesidades que los usuarios expresan verbalmente.
- Requisitos del Sistema: Son los componentes que el sistema debe tener para realizar determinadas tareas.
- Requisitos Funcionales: Servicios que el sistema debe proporcionar.
- Requisitos no funcionales: Restricciones que afectarán al sistema.

3.6.3.3 Catálogo de requisitos.

La mayoría de los procesos del software comienzan con el análisis de los requisitos. Esto implica recoger y documentar las expectativas de los clientes para el software: exactamente qué se supone que tiene que hacer el software. A continuación se detallan, agrupándolos en secciones según el tipo que son.

3.6.3.3.1 Persistencia de los datos de paciente de cardiología.

- Uso de tecnologías adecuadas para la persistencia de los datos. Prioridad alta.
- Acciones crear, leer, actualizar y borrar para todas las clases definidas en el diagrama de clases del modelo de los datos. Prioridad alta.

3.6.3.3.2 Cabecera de la página de información de paciente.

- Siempre será visible la información demográfica y el estado nutricional del paciente, para saber qué paciente estamos visualizando. También se podrán ver datos de interés para la especialidad como por ejemplo, si padece o no sobrepeso. Prioridad alta.
- Los datos demográficos de paciente son: número de historia clínica, nombre, fecha de nacimiento, sexo, contacto, otro contacto y dirección postal del paciente. Estos datos son de sólo lectura, ya que se recogen de la BB.DD. de Drago, por lo tanto no se pueden modificar desde la aplicación de cardiología. Prioridad alta.
- Los datos nutricionales del paciente son: peso, estatura, el área de superficie corporal (BSA), el índice de masa corporal (IMC) y si el paciente padece sobrepeso o no. Los datos que son requeridos son el peso y la estatura ya que los otros tres atributos se calcularán a partir de éstos. Prioridad alta.

- La unidad de medida para el peso es (Kg), y para la estatura es (metros).
- Las variables IMC y BSA del estado nutricional se calcularán automáticamente a través del peso y estatura del paciente, según las fórmulas facilitadas por la OMS. Son campos de sólo lectura para el usuario. Prioridad media.

3.6.3.3.3 Navegación de la aplicación.

- Experiencia del usuario. La información relacionada ha de agruparse para mejorar la usabilidad de la aplicación. Prioridad media.

3.6.3.3.4 Sección de ficha de paciente.

En esta sección, encontraremos siete apartados sobre información de interés del paciente para la especialidad, que pueden ayudar para el diagnóstico de cardiopatías, para la realización de pruebas médicas o algún protocolo médico. Prioridad alta.

- Antecedentes referidos a la salud del paciente. Se podrá visualizar el historial de antecedentes del paciente, y además, se podrá visualizar los datos de cada antecedente, y también modificarlos o eliminarlos. Para registrar un antecedente en la BB.DD de cardiología, debe haber un formulario con tres campos:
 1. Fecha del antecedente. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido.
 2. Tipo de antecedente. El usuario podrá introducirlo a través de un desplegable el cual recoge todos los antecedentes de interés para la especialidad.
 3. Comentario. El usuario podrá introducir un comentario acerca de este antecedente en un campo de texto.
- Alergias del paciente. Se guardan en la BB.DD. de cardiología a través de un campo de texto requerido, para que el usuario escriba la alergia de manera detallada si lo desea. Se podrá visualizar el historial de alergias del paciente, y además, se podrá visualizar los datos de cada alergia, y también modificarlas o eliminarlos.
- Indicaciones a realizar por el paciente, a criterio del facultativo. Se podrá visualizar el historial de éstas, y además, se podrá visualizar los datos de cada indicación, y también modificarlas o eliminarlas. Se salvarán tres atributos sobre las indicaciones:
 1. Fecha de la indicación. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido.

2. Tipo de indicación. El usuario podrá introducirlo a través de un desplegable que recoge enfermedades, protocolos aplicados, etc del paciente de interés para la especialidad.
 3. Comentario. El usuario podrá introducir una indicación acerca del valor elegido en el desplegable en un campo de texto.
- Procedimientos realizados al paciente en el pasado de tipo pruebas médicas. Se podrá visualizar el historial de procedimientos del paciente, y además, se podrá visualizar los datos de cada procedimiento, y también modificarlos o eliminarlos. Para describir este apartado, se necesitan cuatro atributos:
 1. Fecha del procedimiento. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido.
 2. Tipo de procedimiento. El usuario podrá introducirlo a través de un desplegable que recoge las pruebas médicas de interés para la especialidad realizadas al paciente.
 3. Diagnóstico del procedimiento. Recoge el diagnóstico de la prueba realizada al paciente, mediante un campo de texto.
 4. Comentario del procedimiento. Da la posibilidad de introducir un comentario de esta prueba registrada, si el usuario lo desea, a través de un campo de texto.
 - Ingresos en los diferentes servicios de la sanidad pública. Se podrá visualizar el historial de ingresos del paciente, y además, se podrá visualizar los datos de cada ingreso, y también modificarlos o eliminarlos. Se guardarán tres campos para describir este ingreso:
 1. Fecha del ingreso. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido.
 2. Tipo de servicio. El usuario podrá introducirlo a través de un desplegable que recoge todas las áreas de servicio de la sanidad pública.
 3. Comentario. Da la posibilidad de introducir un comentario de este ingreso del paciente si el usuario lo desea, a través de un campo de texto.
 - Baja por fallecimiento. Relata información sobre el fallecimiento del paciente. Se podrá visualizar estos datos, y modificarlos o eliminarlos. Éste tiene a su vez cuatro sub-apartados:
 - Electrocardiograma basal.
 1. Ritmo. Es un desplegable.
 2. QRS. Es una medida numérica.
 3. Cicatriz. Es un desplegable.

- 4. Localización. Es un desplegable.
- Dispositivos previos.
 - 1. MCP previo. Es un desplegable.
 - 2. MCP dependiente. Es un desplegable.
 - 3. DAI previo. Es un desplegable.
 - 4. Terapias previas. Es un desplegable.
- Outcomes.
 - 1. Fecha de exitus. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido.
 - 2. Causa de exitus CV. Es un desplegable.
 - 3. Causa de exitus. Causa de la muerte. Es un desplegable.
 - 4. Respondedor. Es un desplegable.
 - 5. Remodelador. Es un desplegable.
 - 6. Vvms. Es un valor numérico.
- Comentario. Permite describir algún comentario sobre el fallecimiento del paciente. Es un campo de texto.
- Comentarios sobre el paciente. Da la posibilidad de introducir un comentario sobre el paciente si el usuario lo desea, a través de un campo de texto. Este campo se debe visualizar, y se puede sobre-escribir o eliminar. Prioridad media.

3.6.3.3.5 Sección de pruebas médicas que se realizan al paciente.

Esta sección constará de un apartado para cada tipo de prueba médica cardiológica. Actualmente, en el departamento de cardiología se desarrollan ocho tipos de pruebas diferentes. Para cada tipo de prueba existirá un formulario donde se registrarán sus parámetros indicando su unidad de medida para no falsear el diagnóstico de la prueba, y además, se podrá generar su informe automáticamente en formato PDF a través de un botón. En cada apartado de prueba médica, se podrá visualizar el historial de pruebas de ese tipo que tenga el paciente, y además, se podrá visualizar los datos de cada prueba, y también modificarlos o eliminarlos. El paciente puede realizarse 0, 1 o muchas pruebas de todas estas descritas. Prioridad alta.

- Cateterismos. Se podrá visualizar el historial de pruebas médicas de cateterismo del paciente, y además, se podrá visualizar los datos de cada prueba, y también modificarlos o eliminarlos. Para registrar una prueba cateterismo en la BB.DD de cardiología, debe haber un formulario con tres campos:

1. Fecha de realización de la prueba. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
 2. Médico. El usuario podrá introducir el médico que realiza la prueba a través de un desplegable el cual incluye todos los médicos dados de alta en la BB.DD del servicio de cardiología del HUGCDN.
 3. Informe. El usuario podrá introducir el informe de esta prueba en un área de texto, y se prefiere que éste campo disponga de editor de texto, para que el facultativo pueda darle formato al texto.
- Ecocardiografías. Se podrá visualizar el historial de pruebas médicas de ecocardiografía del paciente, y además, se podrá visualizar los datos de cada prueba, y también modificarlos o eliminarlos. Para registrar una prueba ecocardiografía en la BB.DD de cardiología, debe haber un formulario con tres campos:
 1. Fecha de realización de la prueba. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
 2. Médico. El usuario podrá introducir el médico que realiza la prueba a través de un desplegable el cual incluye todos los médicos dados de alta en la BB.DD del servicio de cardiología del HUGCDN.
 3. Plantilla. Se introducirá una plantilla (texto) automáticamente en el área de texto Informe (siguiente parámetro) al seleccionar desde este desplegable el tipo de plantilla escogido, para facilitar la redacción del informe ya que son datos genéricos que siempre aparecerán.
 4. Informe. El usuario podrá introducir el informe de esta prueba en un área de texto, y se prefiere que éste campo disponga de editor de texto, para que el facultativo pueda darle formato al texto.
 - Ecocardiografías transtorácicas. Se podrá visualizar el historial de pruebas médicas de ecocardiografías transtorácicas del paciente, y además, se podrá visualizar los datos de cada prueba, y también modificarlos o eliminarlos. Para registrar una prueba ecocardiografía transtorácica en la BB.DD de cardiología, debe haber un formulario con 18 campos:
 1. Fecha de realización de la prueba. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
 2. Médico. El usuario podrá introducir el médico que realiza la prueba a través de un desplegable el cual incluye todos los médicos dados de alta en la BB.DD

del servicio de cardiología del HUGCDN.

3. DTDVI. Es un campo numérico. La unidad de medida es (mm).
 4. Teicholtz. Es un campo numérico. La unidad de medida es (%).
 5. PP. Es un campo numérico. La unidad de medida es (mm).
 6. AI. Es un campo numérico. La unidad de medida es (mm).
 7. GM Mitral. Es un campo numérico adimensional.
 8. GM Aórtico. Es un campo numérico adimensional.
 9. E-TIV. Es un campo numérico. La unidad de medida es (mm).
 10. Onda A. Es un campo numérico adimensional.
 11. DTSVI. Es un campo numérico. La unidad de medida es (mm).
 12. TIV. Es un campo numérico. La unidad de medida es (mm).
 13. DTDVD. Es un campo numérico. La unidad de medida es (mm).
 14. Raíz Ao. Es un campo numérico. La unidad de medida es (mm).
 15. GP Aórtico. Es un campo numérico adimensional.
 16. PAPs. Es un campo numérico adimensional.
 17. Onda E. Es un campo numérico adimensional.
 18. FE. Es un campo numérico. La unidad de medida es (Simpson).
- Ergometrías. Se podrá visualizar el historial de pruebas médicas de ergometrías del paciente, y además, se podrá visualizar los datos de cada prueba, y también modificarlos o eliminarlos. Para registrar una prueba ergometría en la BB.DD de cardiología, debe haber un formulario con 14 campos:
 1. Fecha de realización de la prueba. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
 2. Médico. El usuario podrá introducir el médico que realiza la prueba a través de un desplegable el cual incluye todos los médicos dados de alta en la BB.DD del servicio de cardiología del HUGCDN.
 3. FC Basal. Es un campo numérico. La unidad de medida es (lpm).
 4. TA Basal. Es un campo numérico. La unidad de medida es (mmHg).
 5. Tiempo de ejercicio. Es un campo numérico. La unidad de medida es (min).
 6. Tiempo de ejercicio. Es un campo numérico. La unidad de medida es (seg).
 7. Resultado. Es un campo desplegable.
 8. Protocolo. Es un campo desplegable.
 9. FC Máxima. Es un campo numérico. La unidad de medida es (lpm).
 10. TA Máxima. Es un campo numérico. La unidad de medida es (mmHg).
 11. Ritmo Basal. Es un campo desplegable.
 12. Porcentaje. Es un campo numérico. La unidad de medida es (%).

13. METS. Es un campo numérico adimensional.
14. Notas. El usuario podrá introducir una nota de esta prueba en un área de texto, y se prefiere que éste campo disponga de editor de texto, para que el facultativo pueda darle formato al texto.
- Estudios electrofisiológicos. Se podrá visualizar el historial de pruebas médicas de estudios electrofisiológicos del paciente, y además, se podrá visualizar los datos de cada prueba, y también modificarlos o eliminarlos. Para registrar un estudio electrofisiológico en la BB.DD de cardiología, debe haber un formulario con tres campos:
 1. Fecha de realización de la prueba. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
 2. Médico. El usuario podrá introducir el médico que realiza la prueba a través de un desplegable el cual incluye todos los médicos dados de alta en la BB.DD del servicio de cardiología del HUGCDN.
 3. Informe. El usuario podrá introducir el informe de esta prueba en un área de texto, y se prefiere que éste campo disponga de editor de texto, para que el facultativo pueda darle formato al texto.
 - Hipertensión pulmonar. Se podrá visualizar el historial de pruebas médicas de hipertensión pulmonar del paciente, y además, se podrá visualizar los datos de cada prueba, y también modificarlos o eliminarlos. Para registrar una prueba hipertensión pulmonar en la BB.DD de cardiología, debe haber un formulario con 30 campos:
 1. Fecha de realización de la prueba. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
 2. Médico. El usuario podrá introducir el médico que realiza la prueba a través de un desplegable el cual incluye todos los médicos dados de alta en la BB.DD del servicio de cardiología del HUGCDN.
 3. AI. Es un campo numérico. La unidad de medida es (mm).
 4. DTDVI. Es un campo numérico. La unidad de medida es (mm).
 5. TIV. Es un campo numérico. La unidad de medida es (mm).
 6. E-TIV. Es un campo numérico. La unidad de medida es (mm).
 7. Teicholtz. Es un campo numérico. La unidad de medida es (mm).
 8. Frecuencia cardíaca. Es un campo numérico. La unidad de medida es (lpm).
 9. T1. Es un campo numérico adimensional.
 10. Área AD. Es un campo numérico. La unidad de medida es (cm²).

11. Área VD. Es un campo numérico. La unidad de medida es (cm²).
 12. TAPSE. Es un campo numérico. La unidad de medida es (mm).
 13. GP. Es un campo numérico adimensional.
 14. dP/dt. Es un campo numérico adimensional.
 15. G1. Es un campo numérico adimensional.
 16. G2. Es un campo numérico adimensional.
 17. Raíz Ao. Es un campo numérico. La unidad de medida es (mm).
 18. DTSVI. Es un campo numérico. La unidad de medida es (mm).
 19. PP. Es un campo numérico. La unidad de medida es (mm).
 20. DTDVD. Es un campo numérico. La unidad de medida es (mm).
 21. Arteria Pulmonar. Es un campo numérico. La unidad de medida es (mm).
 22. Tensión arterial. Es un campo numérico. La unidad de medida es (mmHg).
 23. T2. Es un campo numérico adimensional.
 24. D1. Es un campo numérico. La unidad de medida es (mm).
 25. D2. Es un campo numérico. La unidad de medida es (mm).
 26. Índice de excentr VI. Es un campo numérico adimensional.
 27. Vel E. Es un campo numérico adimensional.
 28. Vel A. Es un campo numérico adimensional.
 29. GP. Es un campo numérico adimensional.
 30. GM. Es un campo numérico adimensional.
- Holter's. Se podrá visualizar el historial de pruebas médicas de estudios holter's del paciente, y además, se podrá visualizar los datos de cada prueba, y también modificarlos o eliminarlos. Para registrar un estudio holter's en la BB.DD de cardiología, debe haber un formulario con tres campos:
 1. Fecha de realización de la prueba. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
 2. Médico. El usuario podrá introducir el médico que realiza la prueba a través de un desplegable el cual incluye todos los médicos dados de alta en la BB.DD del servicio de cardiología del HUGCDN.
 3. Informe. El usuario podrá introducir el informe de esta prueba en un área de texto, y se prefiere que éste campo disponga de editor de texto, para que el facultativo pueda darle formato al texto.
 - Resincronizaciones cardíacas. Se podrá visualizar el historial de resincronizaciones cardíacas del paciente, y además, se podrá visualizar los datos de cada prueba, y también modificarlos o eliminarlos. Para registrar una resincronización cardíaca en la BB.DD de cardiología, debe haber un formulario

con 13 campos:

1. Fecha de realización de la prueba. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
2. Médico. El usuario podrá introducir el médico que realiza la prueba a través de un desplegable el cual incluye todos los médicos dados de alta en la BB.DD del servicio de cardiología del HUGCDN.
3. Retraso interventricular (modo M). Es un campo numérico.
4. Retraso interventricular (tipo preeyectivo pulmonar) . Es un campo numérico.
5. Fin del acortamiento tras cierre valvular aórtico. Es un campo numérico.
6. Apreciación subjetiva del movimiento asíncrono del SIV (4C). Es un campo numérico.
7. Insuficiencia mitral presistólica. Es un campo numérico.
8. Insuficiencia mitral > grado II. Es un campo numérico.
9. Tiempo preeyectivo aórtico. Es un campo numérico.
10. Retraso entre el SIV y la PL. Es un campo numérico.
11. Movimiento en M del SIV por DTI color. Es un campo numérico.
12. Tiempo de llenado diastólico con respecto al RR(4C). Es un campo numérico.
13. $dP/dt(4C) < 800 \text{ mmHg/s}$. Es un campo numérico.

3.6.3.3.6 Sección de sistemas.

Esta sección constará de un apartado para cada tipo de sistema cardiológico que tenga instalado el paciente: generadores y electrodos, y además otro apartado donde se filtran los problemas activos que tiene el paciente en relación a éstos aparatos. Para cada tipo de sistema existirá un formulario donde se registrarán sus parámetros para informar del estado de éste, indicando sus unidades de medida para la mejor interpretación de los datos. En cada apartado de sistema, se podrá visualizar el historial de dispositivos de ese tipo que tenga el paciente, y además, se podrá visualizar los datos de cada sistema, y también modificarlos o eliminarlos. El paciente puede tener 0, 1 o muchos sistemas de los descritos. Prioridad alta.

- Generadores. Cada generador puede tener 0, 1 o varios tipos de complicaciones, las cuales se marcarán como checkboxes. También podrá tener 0, 1 o muchos problemas, campos para guardar sus parámetros y un campo de comentario.
 1. Modelo. Es un campo de texto para registrar el modelo del generador.
 2. Número de serie. Es un campo de texto para registrar el número de serie del

generador.

3. Fabricante. Es un campo desplegable, con todos los fabricantes disponibles.
4. Fecha de implante. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
5. Región implante. Es un campo desplegable.
6. Acceso. Es un campo desplegable.
7. Tipo generador. Es un campo desplegable.
8. Resincro. Es un campo desplegable.
9. Lugar. Es un campo desplegable.
10. Médico. Médico y médico ayudante no podrán ser la misma persona. Es un campo desplegable con los médicos de la especialidad en la BB.DD.
11. Médico ayudante. Es un campo desplegable con los médicos de la especialidad en la BB.DD.
12. Enfermero. Es un campo desplegable con los enfermeros dados de alta en la BB.DD.
13. Fecha explante. Esta fecha no puede ser menor que la Fecha Implante. Las vistas para este campo además deben añadir un calendario.
14. Causa explante. Es un campo desplegable.
15. Tipo de complicación. Son múltiples checkbox que definirán si el dispositivo tiene este tipo de complicación o no.
16. Problema del generador. En este apartado de problemas, se podrá visualizar el historial de problemas que tenga el generador, y además, se podrá visualizar los datos de cada problema, y también modificarlos o eliminarlos. El generador puede tener 0, 1 o muchos problemas. Es un formulario el cual tiene 6 campos:
 - 16.1 Fecha de comienzo del problema. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
 - 16.2 Fecha de resolución del problema. Las vistas para este campo además deben añadir un calendario. Este dato es un campo requerido.
 - 16.3 Tipo de problema. Es un campo desplegable.
 - 16.4 Grado del problema. Es un campo desplegable.
 - 16.5 Número de historia clínica. Es un campo de lectura. Indica el paciente al que está vinculado el problema.
 - 16.6 Comentario. El usuario podrá introducir un comentario sobre el problema del generador en un campo de texto.

17. Comentario del generador. El usuario podrá introducir un comentario general sobre el generador en un campo de texto.
- Electrodo. Cada electrodo tiene los siguientes parámetros:
 1. Modelo. Es un campo de texto para registrar el modelo del electrodo.
 2. Número de serie. Es un campo de texto para registrar el número de serie del electrodo.
 3. Fabricante. Es un campo desplegable, con todos los fabricantes disponibles.
 4. Fecha de implante. Las vistas para este campo además deben añadir un calendario. Es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
 5. Cámara. Es un campo desplegable. Es un campo requerido.
 6. Deflexión intrínseca. Es un campo numérico, con unidad de medida (mV/ms).
 7. Amplitud. Es un campo numérico, con unidad de medida (mV).
 8. Impedancia. Es un campo numérico, con unidad de medida (ohmios).
 9. Fecha explante. Esta fecha no puede ser menor que la Fecha Implante. Las vistas para este campo además deben añadir un calendario.
 10. Causa explante. Es un campo desplegable.
 - Problemas activos. Se visualizan los problemas activos del paciente, es decir aquellos en los que la fecha de resolución de problemas es nula. Estos datos serán sólo lectura. Se podrán visualizar el historial de problemas activos en una lista, y se podrá leer los datos de cada problema.

3.6.3.3.7 Sección de visitas.

Prioridad alta. En esta sección se podrá visualizar las visitas programadas y las no programadas del paciente por motivos cardiológicos, los datos de cada una de éstas, y también modificarlas o eliminarlas. El paciente puede tener 0, 1 o muchas visitas médicas. Para cada visita existirá un formulario donde se registrarán datos para apoyar al diagnóstico del paciente, indicando sus unidades de medida para la mejor interpretación de los datos. Cada diagnóstico puede tener 0 ó 1 evento médico, 0 ó 1 programación bradicardia, y 0 ó 1 programación taquicardia. Los datos que aparecen en el formulario son:

1. Fecha visita. Las vistas para este campo además deben añadir un calendario. Es un campo requerido, y por defecto, tomará la fecha actual en el momento de creación.
2. Tipo visita. Es un campo desplegable.

- 3. Diagnóstico:
 - 3.1 Modo taqui. Es un campo desplegable.
 - 3.2 Presencial. Es un campo desplegable.
 - 3.3 Voltaje batería. Es un campo numérico. Su unidad de medida es (V).
 - 3.4 Situación batería. Es un campo desplegable.
 - 3.5 Indicador batería. Es un campo numérico. Su unidad de medida es (%).
 - 3.6 Estado electrodo VD. Es un campo de texto.
 - 3.7 Estimulación VD. Es un campo numérico. Su unidad de medida es (%).
 - 3.8 Amplitud VD. Es un campo numérico. Su unidad de medida es (mV).
 - 3.9 Umbral VD. Es un campo numérico. Su unidad de medida es (V*ms).
 - 3.10 Impedancia. Es un campo numérico. Su unidad de medida es (Ohmios).
 - 3.11 Comentario. Es un campo de texto.
 - 3.12 Tratamiento y recomendaciones. Es un campo de texto.
 - 3.11 Tipo agenda. Es un campo desplegable.
 - 3.12 No acudió. Es un checkbox.
 - 3.13 Firma. Es un desplegable con los valores de los médicos activos en la BB.DD. de Drago.
 - 3.14 Evento médico.
 - 3.14.1 Fibrilación auricular. Es un campo desplegable.
 - 3.14.2 Taquicardia ventricular. Es un campo desplegable.
 - 3.14.3 Otras Arritmias. Es un campo de texto.
 - 3.14.4 Descargas. Es un campo desplegable.
 - 3.14.5 Comentario descargas. Es un campo de texto.
 - 3.14.6 ATP. Es un campo desplegable.
 - 3.14.7 Comentario ATP. Es un campo de texto.
 - 3.14.8 Reinicio contadores. Es un campo de texto.
 - 3.14.9 Episodios FV. Es un campo de texto.
 - 3.14.10 Episodios TV. Es un campo de texto.
 - 3.14.11 Episodios TV-1. Es un campo de texto.
 - 3.14.12 Episodios TNVS. Es un campo de texto.
 - 3.14.13 Cambios RTA. Es un campo de texto.
 - 3.14.14 Episodios FA. Es un campo de texto.
 - 3.14.15 Episodios TSV. Es un campo de texto.
 - 3.15 Programación bradicardia.
 - 3.15.1 Modo estimulación. Es un campo desplegable.
 - 3.15.2 LIF. Es un campo numérico.
 - 3.15.3 LMS. Es un campo numérico.

- 3.15.4 LSF. Es un campo de texto.
 - 3.15.5 Tipo estimulación ventricular. Es un campo desplegable.
 - 3.15.6 Retraso ventrículos. Es un campo numérico.
 - 3.15.7 Intervalo AV estimulación. Es un campo numérico.
 - 3.15.8 Intervalo AV detección. Es un campo numérico.
 - 3.15.9 Ciclos búsqueda AV. Es un campo numérico.
 - 3.15.10 Histéresis AV. Es un campo numérico.
 - 3.15.11 Sensibilidad VD. Es un campo numérico.
 - 3.15.12 Salida VD. Es un campo numérico.
 - 3.15.13 Config. Estimulación/Detección VD. Es un campo numérico.
- 3.16 Programación taquicardia.
- 3.16.1 Zona TV-1. Es un campo numérico.
 - 3.16.2 Zona TV . Es un campo numérico.
 - 3.16.3 Zona FV . Es un campo numérico.
 - 3.16.4 ATP TV-1. Es un campo desplegable.
 - 3.16.5 ATP TV . Es un campo desplegable.
 - 3.16.6 Núm. ráfagas ATP1 TV-1. Es un campo numérico.
 - 3.16.7 Núm. ráfagas ATP1 TV . Es un campo numérico.
 - 3.16.8 ATP2 TV-1. Es un campo numérico.
 - 3.16.9 ATP2 TV . Es un campo numérico.
 - 3.16.10 Núm. ráfagas ATP2 TV-1. Es un campo numérico.
 - 3.16.11 Núm. ráfagas ATP2 TV . Es un campo numérico.
 - 3.16.12 Descarga TV-1. Es un campo numérico.
 - 3.16.13 Descarga TV . Es un campo numérico.
 - 3.16.14 Descarga FV . Es un campo numérico.
 - 3.16.15 Descarga 2 TV-1. Es un campo numérico.
 - 3.16.16 Descarga 2 TV . Es un campo numérico.
 - 3.16.17 Descarga 2 FV . Es un campo numérico.
 - 3.16.18 Máx. energía TV-1. Es un campo numérico.
 - 3.16.19 Máx. energía TV . Es un campo numérico.
 - 3.16.20 Máx. energía FV . Es un campo numérico.
 - 3.16.21 Prórroga TV-1. Es un campo numérico.
 - 3.16.22 Prórroga TV . Es un campo numérico.
 - 3.16.23 Prórroga FV . Es un campo numérico.
 - 3.16.24 Modo taqui. Es un campo desplegable.
 - 3.16.25 Comentario. Es un campo de texto.

3.6.3.3.8 Acceso a la aplicación.

- Cada usuario, del tipo administración, facultativo, o informático, debe acceder a la aplicación con autenticación de usuario y contraseña. De no ser así, no se permitirá el acceso a ésta, ni tampoco a los datos de paciente.
- Cada usuario autenticado, una vez dentro de la aplicación, puede ver a todos los pacientes de la BB.DD de Drago, y puede realizar búsquedas de pacientes filtrando por los siguientes atributos de paciente: por número de historia clínica, por nombre, por teléfonos de contacto, por dirección postal, por peso, por estatura, por BSA, por IMC, o por sobrepeso. Prioridad alta.

3.6.3.3.9 Roles de usuario.

Prioridad alta.

- Usuario de administración. Sólo podrá leer cualquier información de todos los pacientes, e imprimir los informes de pruebas médicas y visitas médicas registradas en la base de datos.
- Usuario facultativo. Puede crear, modificar y leer cualquier información de todos los pacientes, e imprimir los informes de pruebas médicas y visitas médicas registradas en la base de datos. No podrá borrar ningún registro, si lo desea, lo hará bajo órdenes de mantenimiento al área de informática del HUGCDN.
- Usuario informático. Puede crear, modificar, leer y borrar cualquier información de todos los pacientes, e imprimir los informes de pruebas médicas y visitas médicas registradas en la base de datos.

3.6.3.3.10 Auditoría de los datos.

- Registro de la actividad de los usuarios en la aplicación para la auditoría de los datos. Cada vez que se grabe, modifique, lea o elimine alguna información, se registrará en una tabla, guardándose qué usuario fue, qué acción realizó, y qué información se manipuló. El identificador único para este registro está compuesto por el usuario, fecha y hora, y qué información maneja. Prioridad alta.

3.7 Evaluación de riesgos.

En cualquier proyecto de desarrollo, los fallos de especificación de diseño pueden ser difíciles de encontrar hasta que se comienza a construir el software, o el usuario final llega a interactuar con él.

Por otra parte, cuanto más tarde (dentro del proceso de desarrollo) se detecta un error, más coste conlleva corregirlo. Por ejemplo, si se encuentra un error de requerimiento durante el proceso de prueba, se tienen que modificar los requisitos adecuadamente, ajustar el diseño completo en función de estos y, finalmente, corregir todo el código que se ve afectado por el cambio de diseño.

Con el fin de reducir los riesgos e inconvenientes que podían suceder en el desarrollo de esta aplicación, se propuso seguir la iteración en espiral para construir software definida en el bloque 2.

En lugar de penalizar los cambios como sucede con el proceso de cascada, este proceso busca descubrir cualquier hipótesis de diseño que falte o requisitos incorrectos u otros defectos, tan pronto como sea posible durante el proceso. De hecho, permite el cambio en cada iteración, haciendo que sea parte del propio proceso de desarrollo.

3.7.1 Identificación de los riesgos [51].

Existen dos tipos distintos de riesgos: riesgos genéricos y riesgos específicos del producto [52].

Los riesgos genéricos son una amenaza potencial a todo proyecto de software. Por otro lado, los riesgos específicos del producto pueden identificarse solamente por quienes tienen clara comprensión de la tecnología, el personal y el entorno específico del software que se construye.

Tipo de riesgo	Nombre	Posibles riesgos
Personal	RG1	Algún perfil no disponible para el desarrollo del Software.
	RG2	Algún perfil sin conocimientos requeridos para el desarrollo.
Requisitos	RG3	Cambios de requisitos que obligan a hacer grandes cambios en el diseño, código etc de la aplicación.
	RG4	No satisfacción de las necesidades de los usuarios del HUGCDN.
Tecnológico	RG5	Incompatibilidad con las tecnologías empleadas.
	RG6	Gestión de la información protegida.

Tabla 8: Tabla de riesgos generales.

Tipo de riesgo	Nombre	Posibles riesgos
Entorno de desarrollo	RE1	Riesgos asociados con la disponibilidad y calidad de las herramientas por usar para construir la aplicación.
Tamaño y experiencia del personal	RE2	Riesgos asociados con la experiencia técnica y de proyecto global de los ingenieros del software que hacen el trabajo.
Impacto empresarial	RE3	Riesgos asociados con restricciones impuestas por la Administración o por el mercado.

Tabla 9: Tabla de riesgos específicos.

3.7.2 Estimación del riesgo [53].

La proyección del riesgo, también llamada estimación del riesgo, intenta calificar cada riesgo en dos formas: 1) la posibilidad o probabilidad de que el riesgo sea real y 2) las consecuencias de los problemas asociados con el riesgo, en caso de que ocurra.

Componentes		Rendimiento	Apoyo	Costo	Calendario
Categoría					
Catastrófico	1	La falla para satisfacer el requisito resultaría en fallo en la misión		La falla da como resultado aumento de costos y demoras en el calendario, con valores esperados en exceso de US\$500K	
	2	Degradación significativa para no lograr el rendimiento técnico	Software que no responde o no puede tener apoyo	Significativos recortes financieros, probable agotamiento de presupuesto	IOC inalcanzable
Crítico	1	Falla para satisfacer el requisito degradaría el rendimiento del sistema hasta un punto donde el éxito de la misión sería cuestionable		La falla da como resultado demoras operativas y/o aumento de costos con valor esperado de US\$100K a US\$500K	
	2	Cierta reducción en rendimiento técnico	Demoras menores en modificaciones de software	Cierto recorte de recursos financieros, posible agotamiento	Posible deterioro en IOC
Marginal	1	Falla para satisfacer los requisitos resultaría en degradación de misión secundaria		Costos, impactos y/o calendario recuperable se deterioran con valor esperado de US\$1K a US\$100K	
	2	Reducción mínima a pequeña en rendimiento técnico	Apoyo de software receptivo	Suficientes recursos financieros	Calendario realista, alcanzable
Despreciable	1	Falla para satisfacer requisitos crearía inconvenientes o impacto no operativo		Error da como resultado costo menor y/o impacto en calendario con valor esperado de menos de US\$1K	
	2	No reducción en rendimiento técnico	Software fácilmente soportable	Posible subejercicio de presupuesto	IOC alcanzable con facilidad

Nota: 1) La consecuencia potencial de errores o fallos de software no detectados.
2) La consecuencia potencial si el resultado deseado no se alcanza.

Figura 29: Valoración de impacto. Fuente: [54].

Nombre	Evaluación de riesgo	Probabilidad
RG1	Crítico	40%
RG2	Crítico	60%
RG3	Crítico	20%
RG4	Catastrófico	5%
RG5	Catastrófico	20%
RG6	Catastrófico	1%
RE1	Catastrófico	10%
RE2	Crítico	10%
RE3	Marginal	5%

Tabla 10: Tabla de riesgos, probabilidades aproximadas de que suceda, y tipo de riesgo.

3.7.3 Planificación de los riesgos [55].

Este paso tiene como objetivo desarrollar una estrategia para tratar los riesgos. Si el equipo de trabajo adopta un enfoque proactivo frente al riesgo, evitarlo será siempre la mejor estrategia. Esto se consigue desarrollando los planes de reducción del riesgo y de contingencia.

En la planificación de riesgos se considera cada uno de los riesgos claves identificados y las estrategias para administrarlos, que vendrán dadas por el juicio y la experiencia del administrador del proyecto.

Nombre	Estrategia
RG1	Reorganizar el equipo de tal forma que se solapen el trabajo y los miembros comprendan el trabajo de los demás.
RG2	Formación adecuada de los perfiles implicados en el desarrollo.
RG3	Rastrear la información para valorar el impacto de los requerimientos, maximizar la información oculta en ellos.
RG4	Continuo feedback con los cardiólogos y jefe de servicios del HUGCDN para recogida de información sobre el uso del prototipo de la aplicación, de sus necesidades y expectativas de la aplicación web.
RG6	Aplicación de los protocolos y catálogo de normas. Auditorías.
RG5, RE1	Documentación sobre las tecnologías empleadas para saber su aplicación en el desarrollo. Ejemplos de uso.
RE2	Formación adecuada de los perfiles implicados en la gestión del proyecto.
RE3	Rastrear la información para valorar el impacto de las restricciones impuestas. En su caso, establecer las medidas correctoras oportunas. Establecer el ente responsable de su cumplimiento.

Tabla 11: Tabla de planificación de los riesgos.

3.7.4 Supervisión de los riesgos [56].

En esta sección se trata qué acciones y decisiones se tomarán ante un problema que ya ha sido identificado, proyectado y evaluado.

La supervisión de riesgos valora cada uno de los riesgos identificados para decidir si es más o menos probable y cuándo han cambiado sus posibles efectos. Hay que controlar factores que pueden indicar cambios en la probabilidad y el impacto.

Nombre	Indicador
RG1	Bajas, ausencia de los perfiles.
RG2	Retrasos en la planificación del proyecto.
RG3	Peticiones de muchos cambios en los requisitos.
RG4	Quejas de los usuarios de la aplicación web.
RG5	Existencia de informes sobre problemas tecnológicos.
RG6	Acceso no autorizado a los datos protegidos.
RE1	Rechazo de los miembros del equipo a utilizar las herramientas.
RE2	Cumplimiento de objetivos.
RE3	Cambios de restricciones en el proyecto por parte de la Administración.

Tabla 12: Supervisión de los riesgos.

3.8 Definición de la arquitectura tecnológica.

En este PFC y basado en las consultas hechas a los implicados, se optó por la solución más económica, apoyada en herramientas de trabajo cedidas gratuitamente y licencias ya utilizadas en el Departamento de Informática del Hospital Universitario Doctor Negrín, como por ejemplo. *Openxava* [57], marco de trabajo Ajax para desarrollo rápido de aplicaciones web empresariales, el uso de sus *frameworks* [58], conjuntos estandarizados de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, y el gestor de base de datos *PostgreSQL* [59] fundamentalmente.

En el siguiente bloque 4 se detallará en profundidad cada una de las tecnologías empleadas.

4. Tecnologías empleadas.

4. Tecnologías empleadas.

4.1 Bases de datos relacionales.

Para tener una visión general de cómo fluyen los datos entre el nivel de cliente (navegador), el nivel de aplicación (aplicación web de cardiología), y finalmente en el nivel de datos (BB.DD. objeto-relacional), se representa la siguiente imagen:

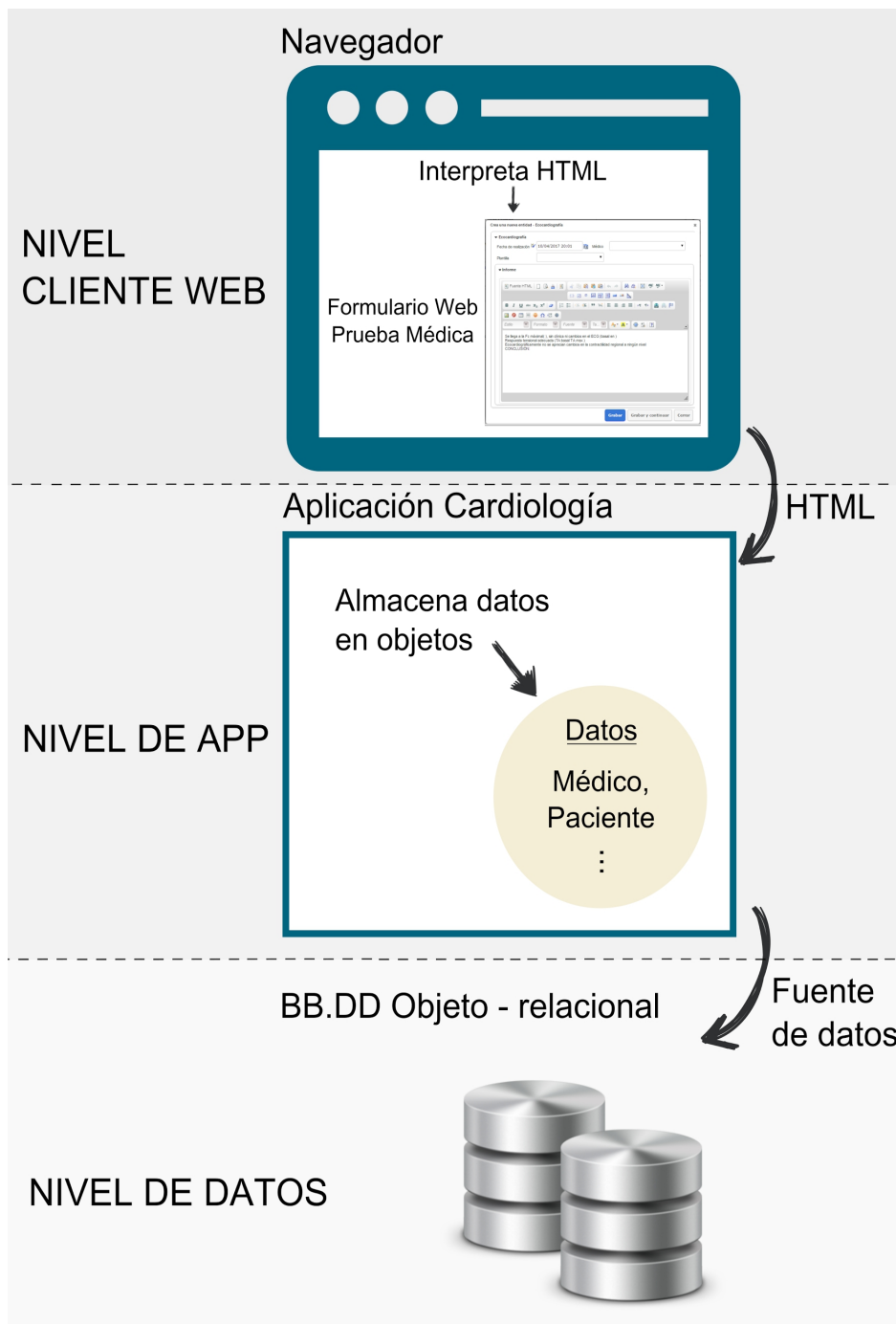


Figura 30: Flujo de datos en una aplicación web.

Como se puede observar, en el lado del cliente, el navegador interpreta código HTML para representar un formulario web. Luego, en el nivel de aplicación se almacena los datos proporcionados en este formulario como objetos, y finalmente, a través de las fuentes de datos (*data source*) [60], o también llamada capa de persistencia, se graban los datos en la base de datos para poder acceder a ellos en la siguiente sesión.

Normalmente se desea que los datos utilizados en una aplicación web persistan de una sesión web a la siguiente. Las bases de datos (BB.DD. referidas de ahora en adelante) relacionales se utilizan comúnmente para este propósito. Ejemplos de datos persistentes son los datos de usuario que definen un perfil, por ejemplo en un portal de empleo, en una página de ventas online, etc.

Estas BB. DD. almacenan colecciones de relaciones entre tablas, cada una con sus respectivos registros y atributos. Cada tabla representa una entidad (clase Java), cada fila de la tabla corresponde a un registro, y sus columnas corresponden a los atributos o campos de este registro.

	id [PK] character varying	fechavisita date	tipovisita character varying	iddiagnostico character varying	numeronhc integer
1	4028804054a972fc0154a9a09aa10004	2016-05-13	0	4028804054a972fc0154a9a09a910003	2
2	4028804054c06c640154c06dbe960005	2016-05-17		4028804054c06c640154c06dbe7c0004	2
3	402880435534a0ba015534c085000005	2016-06-09		402880435534a0ba015534c084e70004	2
4	4028804453c705260153c705fec0001	2016-03-08	0	4028804453c705260153c705fec50000	1
5	4028804453c705260153c7068cc10003	2016-03-21	0	4028804453c705260153c7068cac0002	1
*					

Captura 38: Ejemplo de los registros de la tabla *visita* de la BB.DD. de pruebas de cardiología.

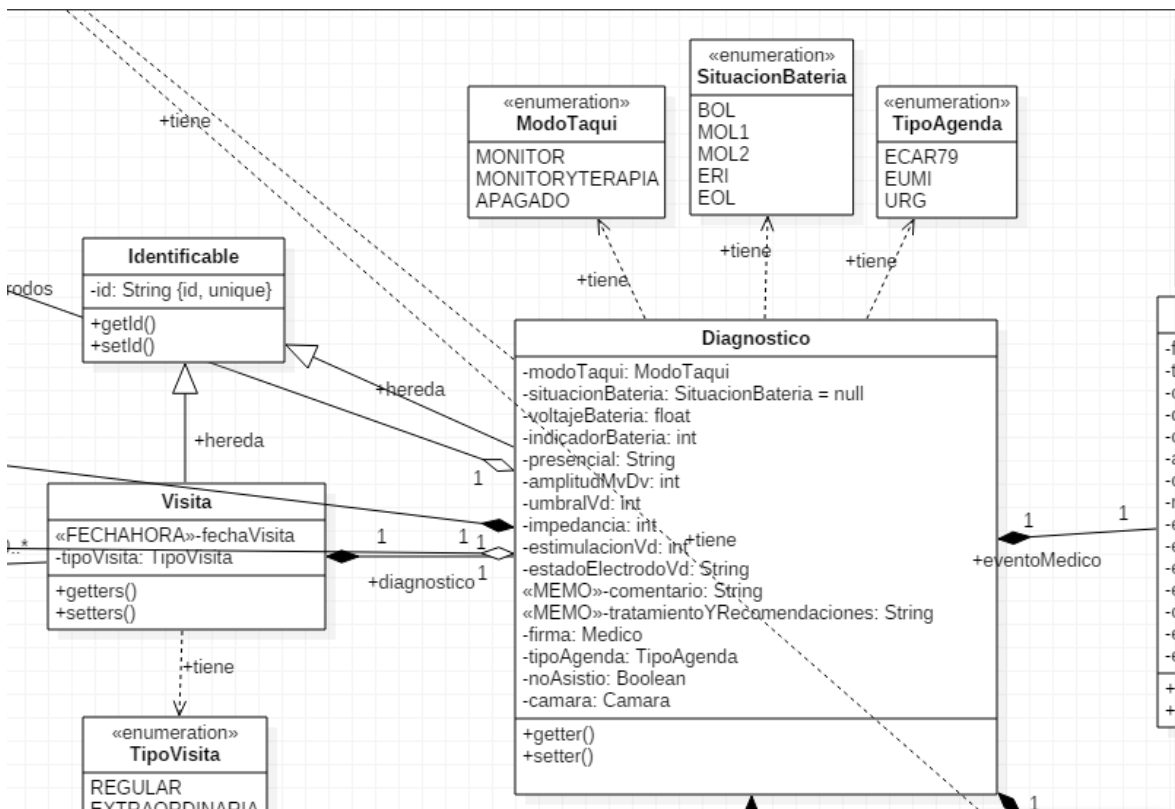
Cada fila corresponde a un registro, a excepción de la primera fila que representa los nombres de los atributos de los registros. Los atributos en este caso, por orden son el identificador *id* para identificar de forma unívoca cada registro en la tabla, la fecha en la que se realiza la visita médica *fechavisita*, el tipo de visita médica que puede ser desde tipo urgencia o programada *tipovisita*, el identificador único *iddiagnostico* que apunta al diagnóstico de esa visita almacenado en la tabla *diagnostico*, y *numeronhc* que corresponde con el número de historia clínica del paciente para saber a qué persona corresponde ese registro.

El identificador único, o la clave primaria [61] (*primary key*) de un registro también es necesario para que los registros de otras tablas puedan apuntarlo. Por ejemplo, vemos

como en la tabla *diagnostico*, existe el id señalado en rojo en la tabla *visita* anterior, ya que existe una relación 1 a 1 entre ambas entidades: una visita tiene un diagnóstico;

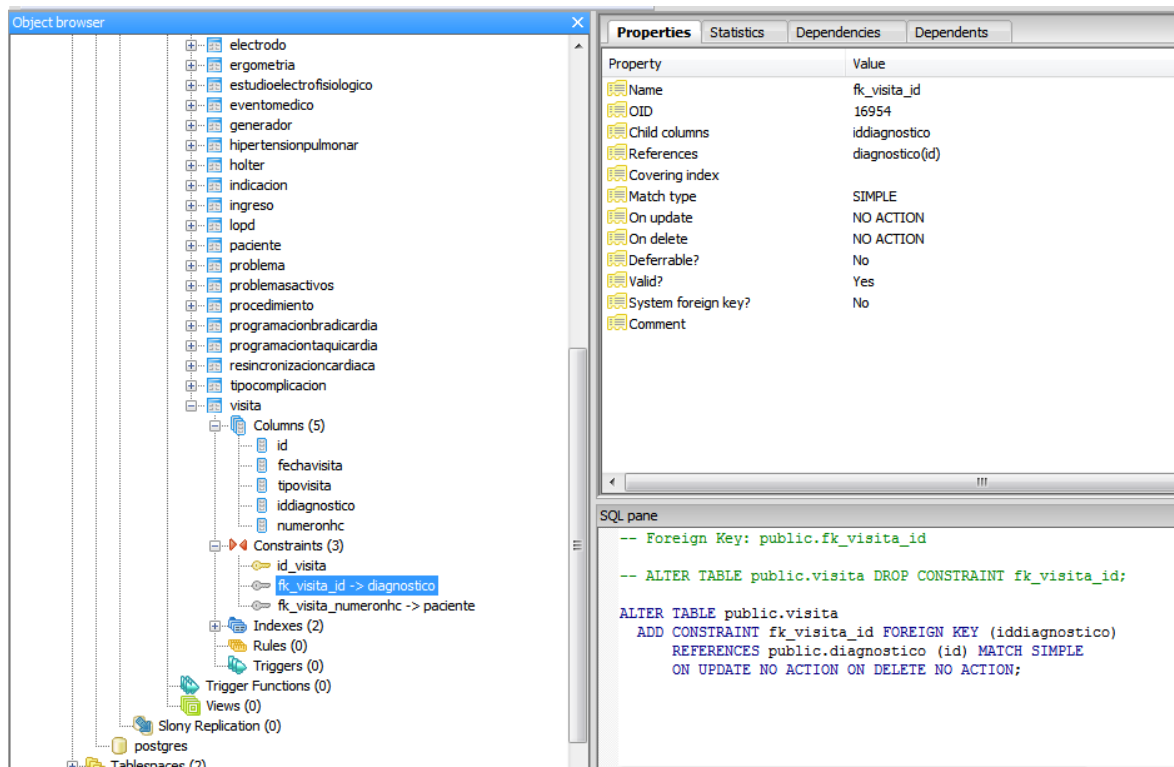
	id [PK] character varying	modotaqui character varying	situacionbateria character varying	voltajebateria numeric	indicadorbateria integer	presencial character varying	amplitud integer
1	4028804054a972fc0154a9a09a910003	0	0	22	32	0	23
2	4028804054c06c640154c06dbe7c0004			0	0		0
3	402880435534a0ba015534c084e70004	1	0	0	0	1	0
4	4028804453c705260153c705fec50000	0	1	33	222	0	11
5	4028804453c705260153c7068cac0002	1	4	333	22	1	3
*							

Captura 39: Datos de la tabla *diagnostico* de la BB.DD de prueba.



Captura 40: Fragmento de diagrama UML donde se aprecia la relación 1 a 1 entre las clases *Visita* y *Diagnostico*.

Para vincular estas tablas, hacemos uso de las claves foráneas [62]. Mostramos cómo se han unido estas dos tablas en el gestor de base de datos utilizado:



Captura 41: Captura de pantalla de las propiedades de la tabla *visita*.

En el siguiente apartado, se explica el gestor de la BB.DD utilizado para realizar este proyecto; *PostgreSQL*.

4.2 Gestión de Bases de Datos con *PostgreSQL*.

Como ya adelantamos en capítulos anteriores, el HUGCDN ha apostado por *PostgreSQL* como sistema de gestión de base de datos objeto-relacional porque es de código abierto, de alta calidad y muy popular.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará al resto y el sistema continuará funcionando.

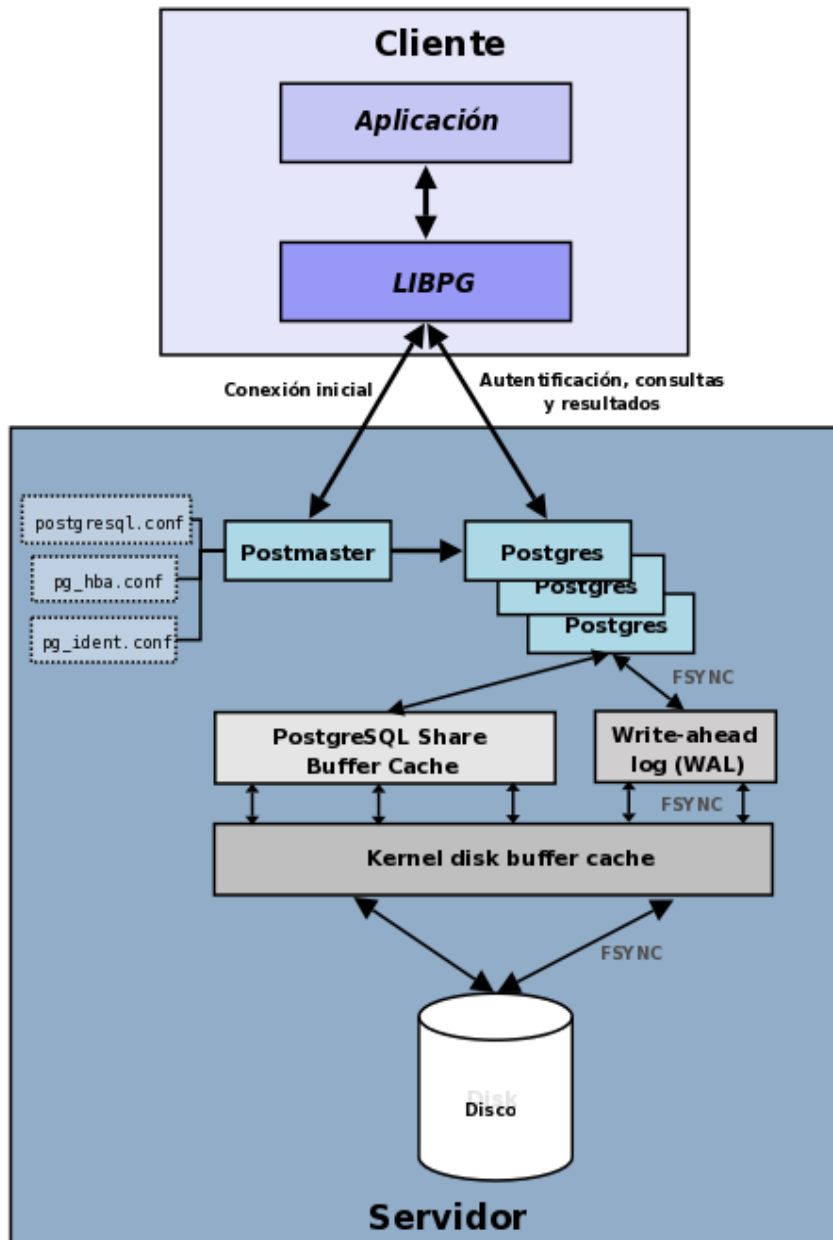


Figura 31: Componentes más importantes en un sistema PostgreSQL [63].

- **Aplicación cliente:** Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir vía TCP/IP o sockets locales.
- **Demonio postmaster:** Este es el proceso principal de PostgreSQL. Es el encargado de escuchar por un puerto/socket por conexiones entrantes de clientes. También es el encargado de crear los procesos hijos que se encargarán de autenticar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes.
- **Ficheros de configuración:** Los 3 ficheros principales de configuración utilizados por PostgreSQL, *postgresql.conf*, *pg_hba.conf* y *pg_ident.conf*.

- **Procesos hijos postgres:** Procesos hijos que se encargan de autenticar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes.
- **PostgreSQL share buffer cache:** Memoria compartida usada por PostgreSQL para almacenar datos en caché.
- **Write-Ahead Log (WAL):** Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO).
- **Kernel disk buffer cache:** Caché de disco del sistema operativo.
- **Disco:** Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione.

Para facilitar la gestión de la Base de Datos *PostgreSQL*, se usó la interfaz gráfica pgAdmin III, la cual se puede ejecutar desde cualquier plataforma, y sus funciones van desde consultas SQL hasta el desarrollo de BB.DD. muy complejas.

4.3 Hibernate como herramienta para la persistencia de los datos.

Para vincular los objetos del modelo de datos usados en nuestra aplicación de cardiología con la base de datos relacional que almacena toda la información necesaria, hacemos uso de *Hibernate* [64], una herramienta de mapeo objeto-relacional *ORM* para la plataforma Java. Este propósito lo conseguimos mediante archivos declarativos *xml* o también, a través de anotaciones en las clases *java* que permiten establecer estas relaciones. En el capítulo de Java explicaremos mejor la estructura de estas clases *java*.

Las anotaciones de *Hibernate Validator* (que no son el motor de persistencia de *Hibernate*) son reconocidas por nuestro marco de trabajo *OpenXava*, por tanto se pueden usar todas sus anotaciones predefinidas en la aplicación de cardiología:

Anotación	Aplica a	Validación
@Length(min=, max=)	Propiedad (cadena)	Comprueba que la longitud de la cadena esté dentro del rango
@Max(value=)	Propiedad (numérica o cadena representando un valor numérico)	Comprueba que el valor sea igual o menor al máximo
@Min(value=)	Propiedad (numérica o cadena representando un valor numérico)	Comprueba que el valor sea igual o mayor al mínimo
@NotNull	Propiedad	Comprueba que el valor no sea nulo
@NotEmpty	Propiedad	Comprueba que la cadena no sea nulo ni vacía
@Past	Propiedad (fecha o calendario)	Comprueba que la fecha esté en el pasado
@Future	Propiedad (fecha o calendario)	Comprueba que la fecha esté en el futuro
@Pattern(regex="reg exp", flag=)*	Propiedad (cadena)	Comprueba que la propiedad cumpla la expresión regular dado un <i>match flag</i>

Figura 32: Algunas de las anotaciones que son utilizables desde Openxava [65].

Además, *Openxava* define anotaciones de validación propias con *Hibernate Validator*, como por ejemplo:

Anotación	Aplica a	Validación
@Required	Propiedad	Comprueba si la propiedad tiene valor
@PropertyValidator	Propiedad	Permite definir una lógica de validación personalizada
@EntityValidator	Entidad	Permite definir una lógica de validación personalizada
@RemoveValidator	Entidad	Permite definir una lógica de validación personalizada al borrar

Figura 33: Validaciones predefinidas de *Openxava*.

Como ejemplo del uso de estas anotaciones, observamos un fragmento de la clase paciente:

```

61 @Table(name = "paciente")
62 public class Paciente {
63
64     //PROPIEDADES
65     @Id // La propiedad numero es la clave. Las claves son obligatorias (required) por defecto
66     @Column(name="NUMERONHC", length=6) // La longitud de columna se usa a nivel UI y a nivel DB
67     public int numeroNhc;
68
69     @Column(name="NOMBRE") // La longitud de columna se usa a nivel UI y a nivel DB
70     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
71     private String nombre;
72
73     @Column(name="FECHANACIMIENTO")
74     @Stereotype("FECHAHORA")
75     //@Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
76     private Date fechaNacimiento;
77

```

Captura 42: Fragmento de código desde la línea 61 a 77 de la clase *Paciente.java*.

Observamos que, en la línea 70 resaltada, la columna *nombre* de la tabla *paciente* es un atributo obligatorio, y por lo tanto no se dejará que se persistan los datos si ésta no tiene valor. El efecto que tiene en la aplicación al intentar grabar el paciente sin el nombre es éste:

KHArдиоLOGía - Paciente ☆

« ‹ › » Nuevo Grabar Borrar Q Buscar ↻ Refrescar

✖ Es obligado que Nombre en Paciente tenga valor

▼ Datos demográficos

Número de Historia Nombre

Fecha de nacimiento Sexo

Contacto Otro contacto

Dirección

Captura 43: Captura de pantalla de la sección datos demográficos del paciente de la aplicación web.

Al intentar grabar un paciente sin el nombre, no permitirá grabar este registro en la BB.DD. hasta que la casilla Nombre con el icono de obligatorio tenga algún valor.

4.3.1 Arquitectura de *Hibernate*.

Con *Hibernate* se crea una capa entre la base de datos y la aplicación donde carga los detalles de la configuración, como la cadena de conexión de base de datos, clases de entidad, asignaciones, etc.

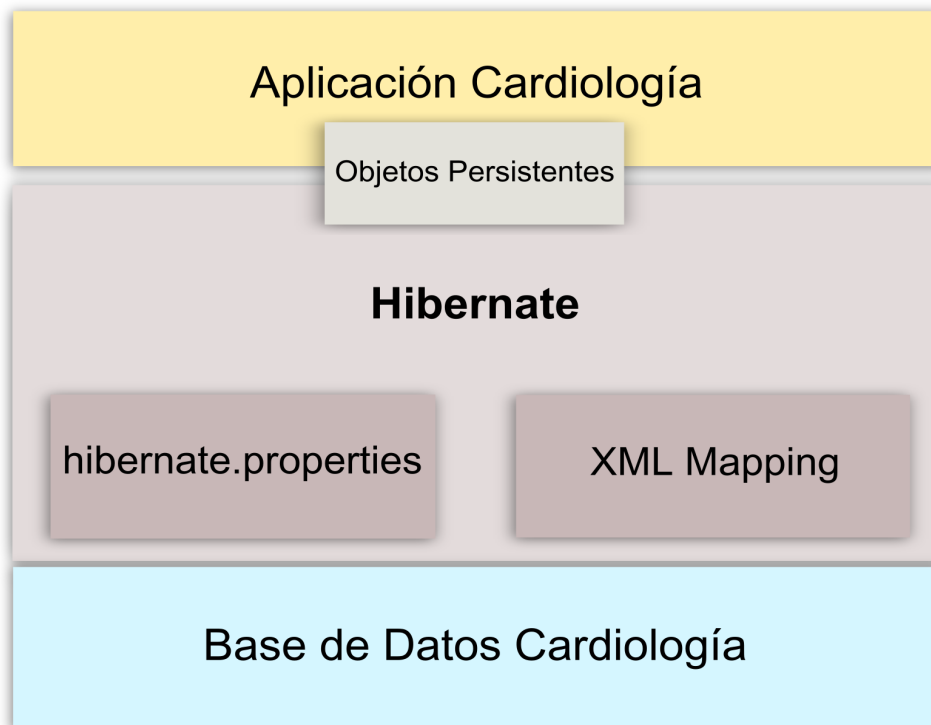


Figura 34: Objetos persistentes que sincronizan los datos entre la aplicación y la base de datos [66].

La herramienta *Hibernate* utiliza varios API de Java existentes, como *Java Transacción API (JTA)*, *Java Naming and Directory Interface (JNDI)* y *JDBC*. JNDI y JTA permiten a *Hibernate* integrarse con servidores de aplicaciones J2EE, y JDBC [67] es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL [68] del modelo de base de datos que se utilice.

Veamos como *Hibernate* interactúa entre la aplicación de cardiología y la BB.DD:

1. Se crea una instancia de clase de entidad (clase Java mapeada con la capa de base de datos). Este objeto se llama **objeto transitorio**, ya que aún no está asociado con la sesión o no se conserva en la base de datos *cardiología*.
2. Para guardar este objeto en la base de datos, antes se crea la instancia de la interfaz **SessionFactory**, la cual con la ayuda de **TransactionFactory** y **ConnectionProvider** implementa todos los ajustes de configuración para la base de datos.
3. Cada conexión a la base de datos en *Hibernate* se crea mediante la creación de una instancia de la interfaz **Session**. Ésta, representa una única conexión con la

base de datos.

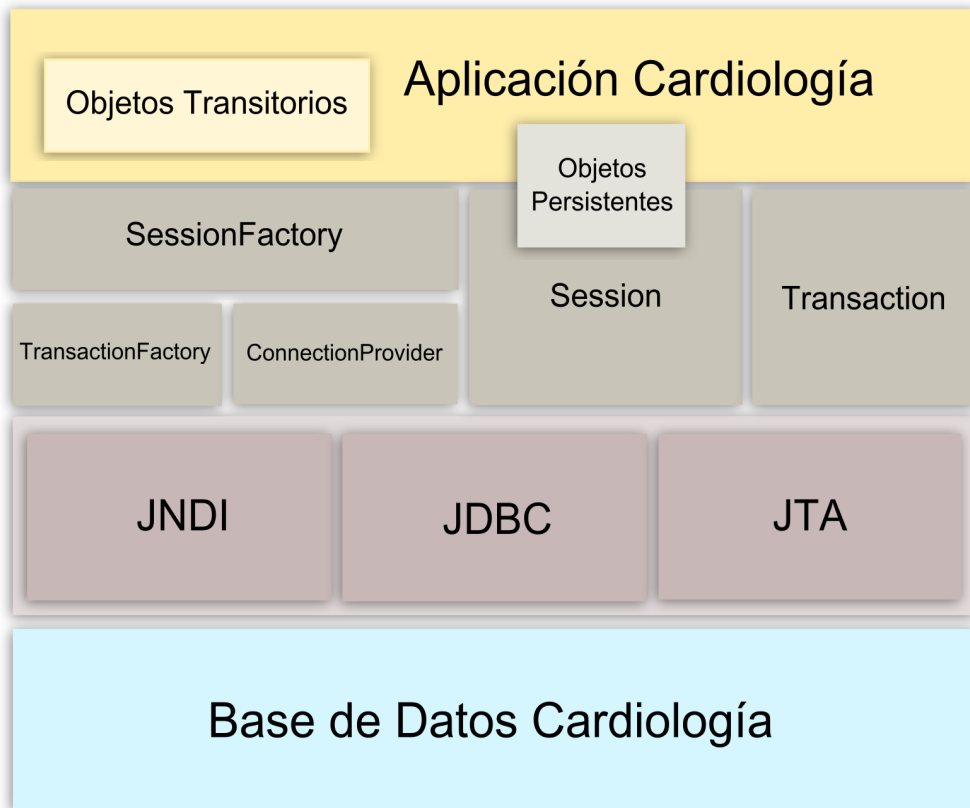


Figura 35: Arquitectura de *Hibernate* en nuestra aplicación de cardiología.

4.3.2 Ventajas de usar *Hibernate* como herramienta de persistencia.

Las principales ventajas del uso de *Hibernate* en este proyecto son:

- Consultas automáticas a la BB.DD. para convertir los registros en objetos (y viceversa).
- Validación de los datos de nuestros objetos Java antes de que se persistan en la base de datos a través de una API de metadatos sobre las clases java de cada entidad.
- Compatibilidad con nuestro gestor de la BB.DD. *PostgreSQL*.

Es por esto que seguimos con la filosofía del proyecto; el desarrollo ágil de aplicaciones, donde el programador no se ocupa de escribir las consultas a la BB.DD.

4.4 Eclipse como herramienta IDE.

Para este proyecto se escogió *Eclipse* [69] como la IDE más apropiada para desarrollar la aplicación de cardiología. En este capítulo describiremos con más detalle esta herramienta y su configuración para su uso.

El entorno de desarrollo

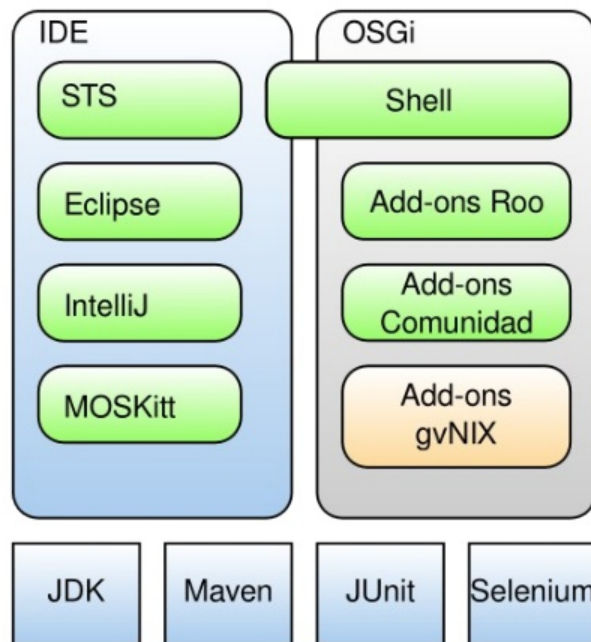
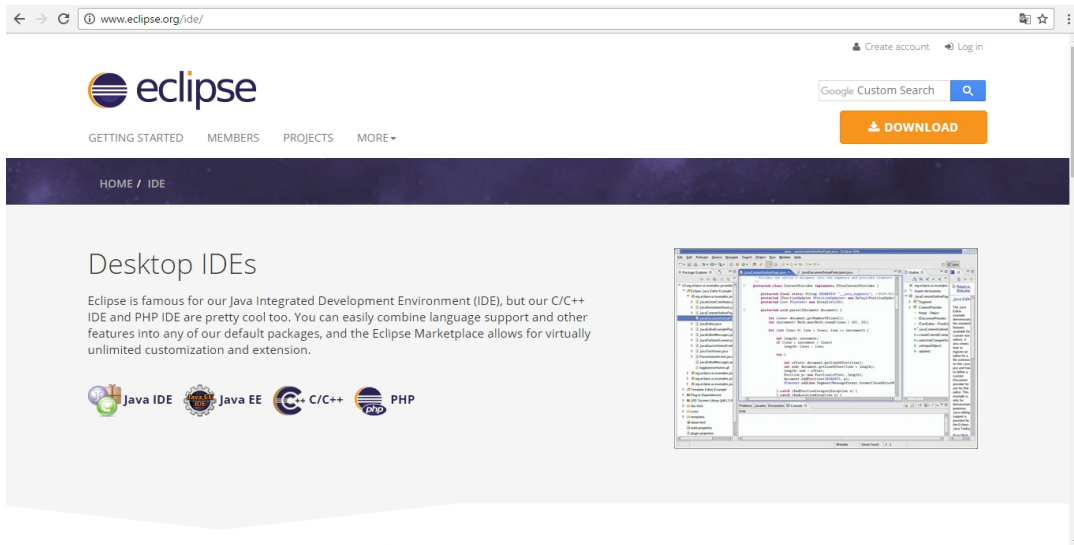


Figura 36: Representación del entorno de desarrollo.

Eclipse es una plataforma de desarrollo de código abierto basada en *Java* la cual incluye servicios para la construcción del entorno de desarrollo de escritorio, web y de la nube.

Además, brinda soporte para varios lenguajes de programación como *Java*, *C/C++*, *JavaScript* y *PHP*, entre otros, y ofrece una amplia colección de herramientas adicionales disponibles para los desarrolladores de software, donde añadiendo o modificando los paquetes predeterminados, y el entorno de *Eclipse* se consigue una personalización y extensión de sus utilidades prácticamente ilimitadas.



Captura 44: Página oficial de descarga para los diferentes IDE's de escritorio *Eclipse* [69].

Para nuestro proyecto, elegimos *Eclipse Java EE IDE* como herramienta de trabajo.

4.4.1 Arquitectura de *Eclipse*.

La plataforma *Eclipse* está estructurada como subsistemas que se implementan en uno o más plug-ins. Los subsistemas se construyen sobre un pequeño motor de tiempo de ejecución. La figura siguiente muestra una vista simplificada:

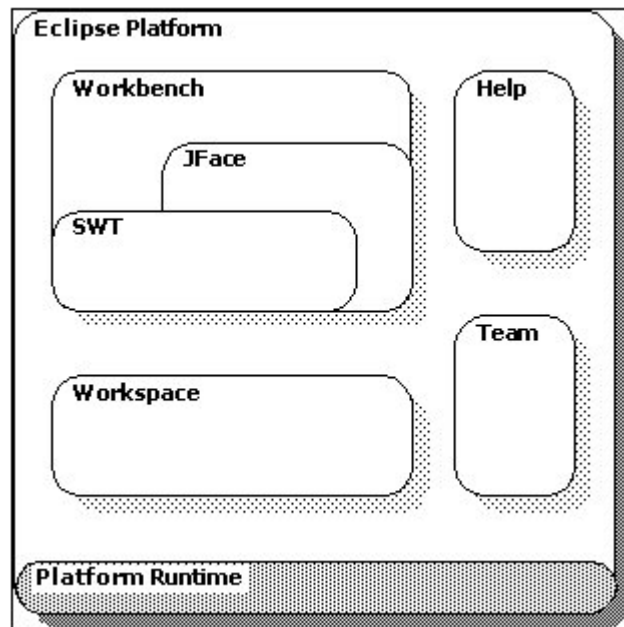


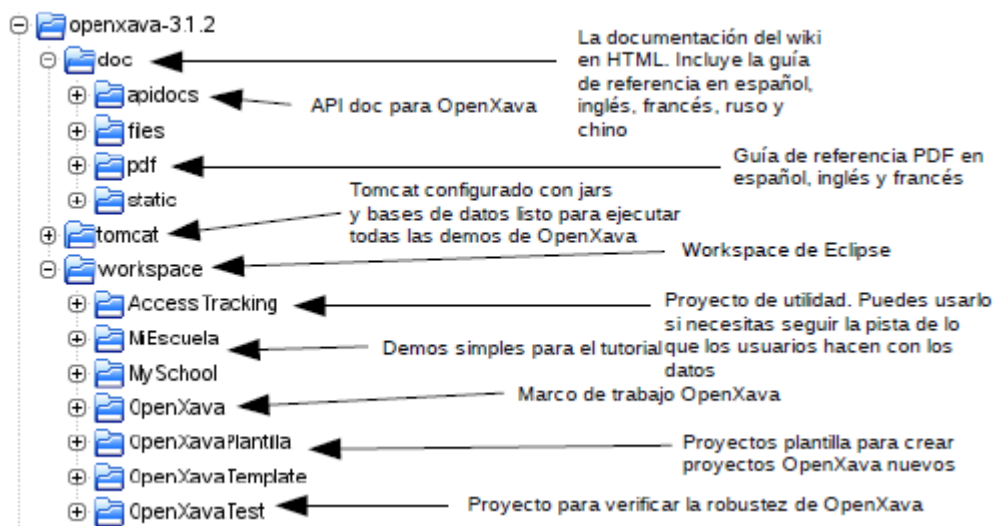
Figura 37: Representación de la estructura de *Eclipse* [70].

4.4.1.1 *Workspace*.

Cada proyecto de desarrollo tiene sus propias características, y por ello hay que gestionar un *workspace* del proyecto.

Workspace, que traducido al español significa el espacio de trabajo de *eclipse*, contiene recursos como proyectos, archivos y carpetas con una estructura jerárquica. Los proyectos están en el nivel superior de la jerarquía y dentro de ellos puede tener archivos y carpetas. Los complementos utilizan una API proporcionada por el complemento de recursos para administrar los recursos en el espacio de trabajo.

A continuación se muestra la estructura del *workspace* donde trabajaremos en *Eclipse* para este proyecto fin de carrera.



Captura 45: Contenido de la distribución de *Openxava*.

4.4.1.2 *Workbench*.

Este término se refiere al entorno de desarrollo de escritorio; vistas, editores, perspectivas y asistentes.

Cada ventana de *Workbench* contiene una o más perspectivas. Las perspectivas contienen vistas y editores y controlan lo que aparece en ciertos menús y barras de herramientas. Pueden existir más de una ventana de *Workbench* en el escritorio en un momento dado.

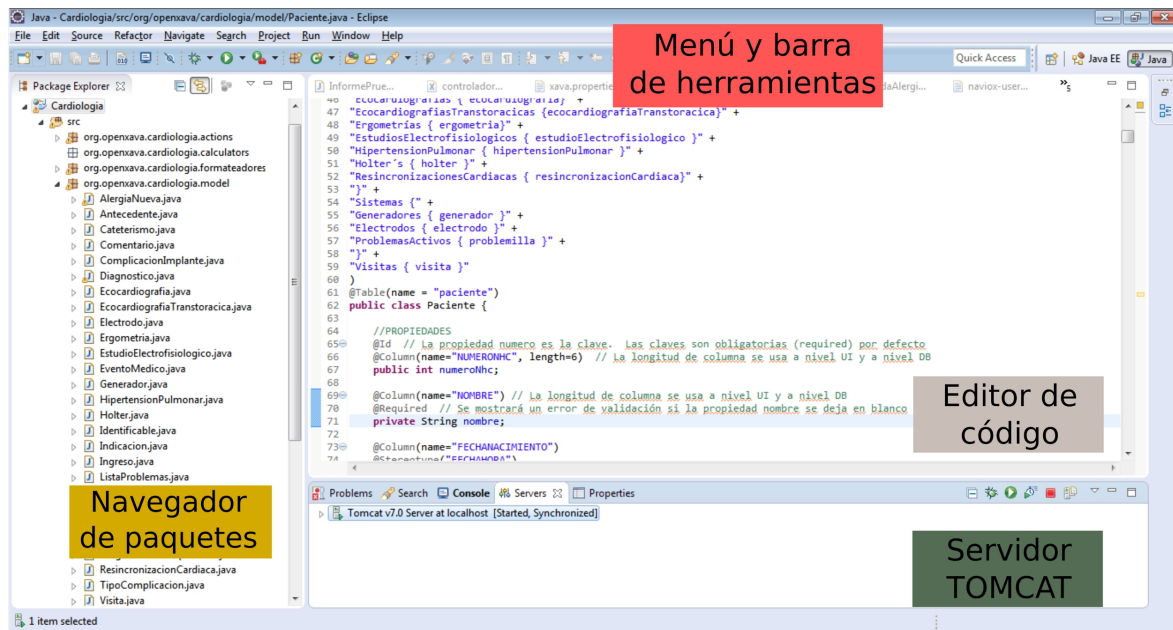
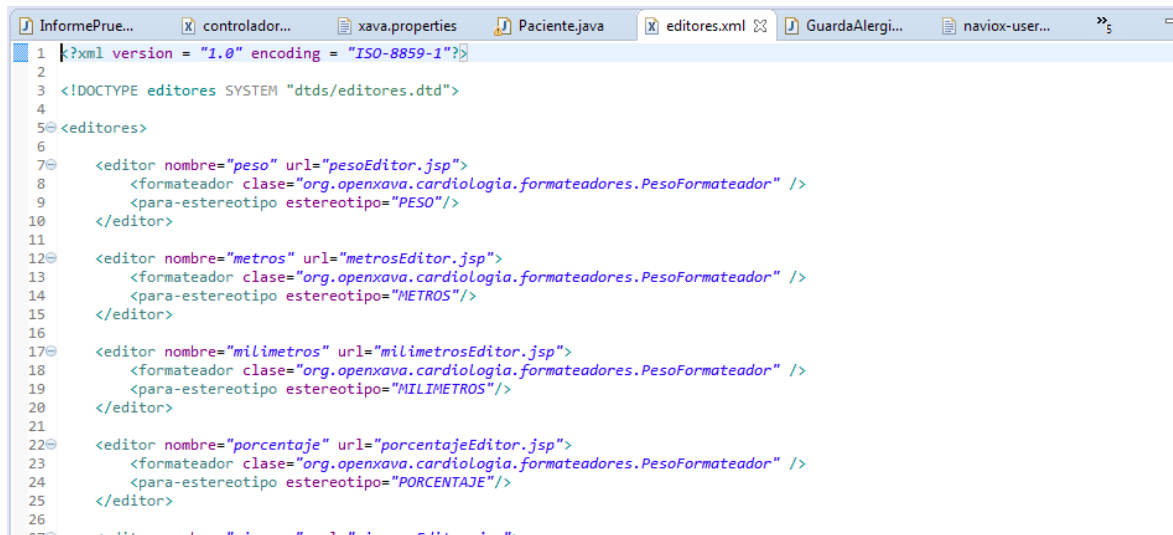


Figura 38: Perspectiva para el desarrollo de la aplicación de cardiología.

4.4.2 Ventajas del uso de esta herramienta.

4.4.2.1 Editor de texto para diferentes lenguajes.

En la misma ventana, podemos editar ficheros de distinta extensión, con analizador sintáctico:



Captura 46: Sección de editor de texto de la perspectiva anterior representada.

Como vemos, las pestañas del área del editor indican los nombres de los recursos que están abiertos para la edición; archivos de código de diferentes lenguajes de

programación, como *java*, *xml*, etc de nuestra aplicación en desarrollo.

4.4.2.2 Barras de Menús y herramientas ajustadas a los lenguajes de programación utilizados.

Como podemos ver a continuación, la barra de menú principal y la barra de herramientas de la ventana *Workbench* contienen operaciones que son aplicables al editor activo.

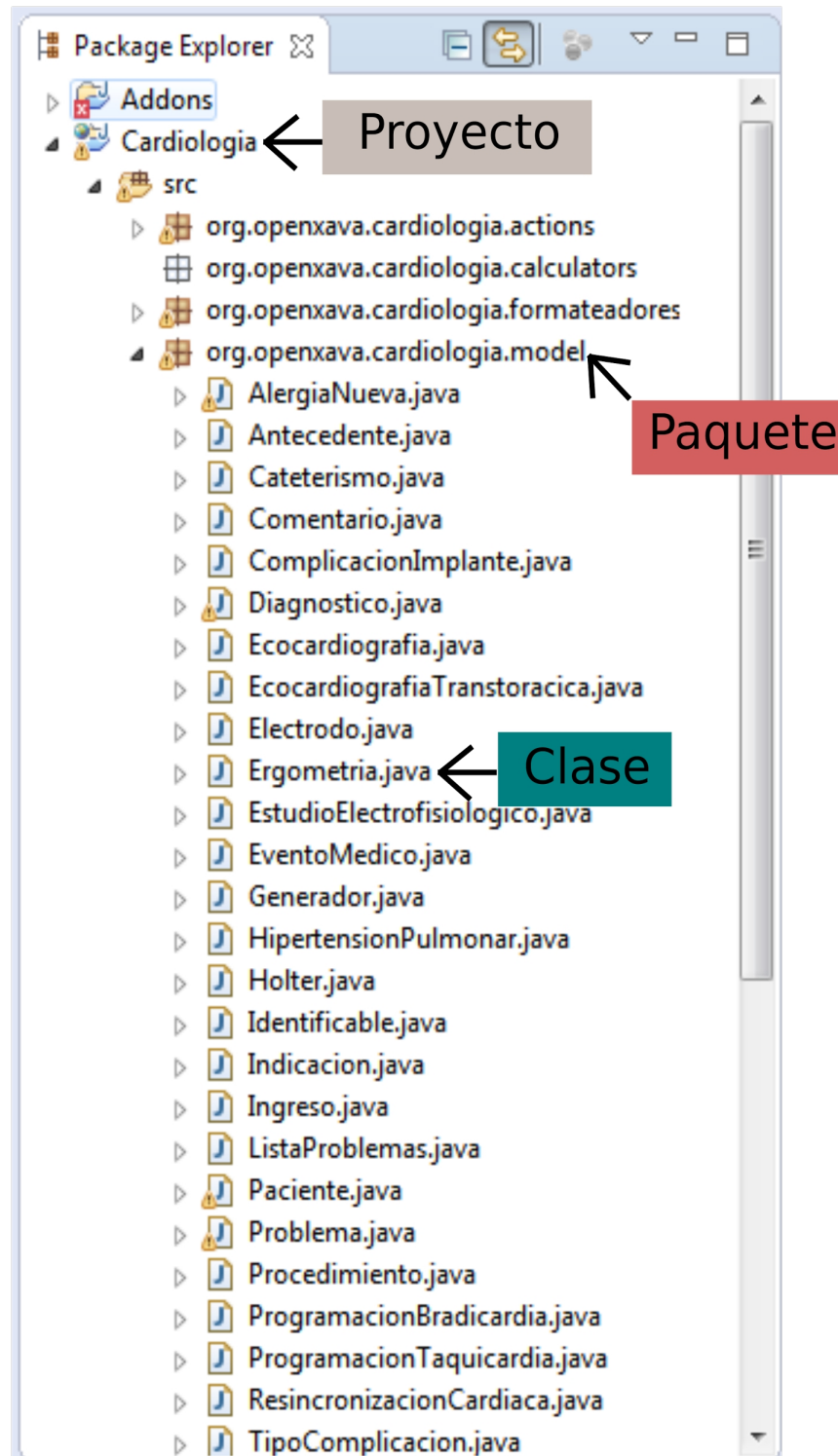


Captura 47: Barra de menú en *Eclipse*.

Se resalta el botón que crea una nueva clase Java.

4.4.2.3 Explorador de proyectos.

Gracias a él, podemos navegar por los distintos proyectos en los que estemos trabajando, y dentro de cada uno, en sus paquetes correspondientes. Por ejemplo, dentro del proyecto cardiología, accedemos a los diferentes paquetes que engloban las acciones, o el modelo de los datos, o los calculadores, etc que explicaremos en el capítulo dedicado a *Openxava* y sus correspondientes clases. En el bloque 5, se explicarán con más detalle estas clases java.

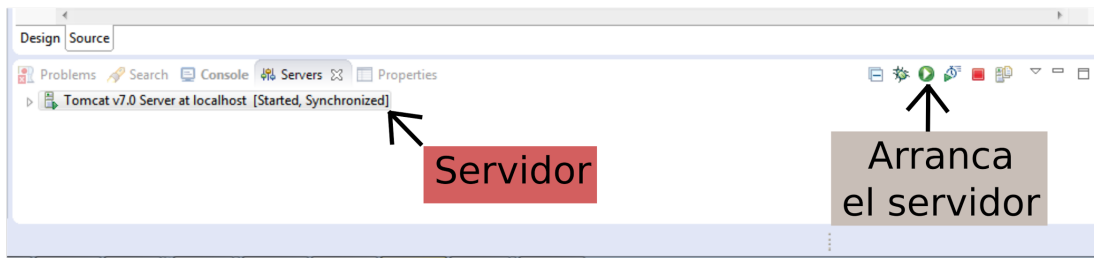


Captura 48: Captura de pantalla del explorador de proyectos de *Eclipse*.

4.4.2.4 Servidores Web Tomcat.

Arrancar el *Tomcat* desde dentro del *Eclipse* tiene varias ventajas, como poder depurar, ver los mensajes de log y trazas dentro del *Eclipse*, ir desde una traza al código con un

solo click, etc.



Captura 49: Sección del servidor *Tomcat* incorporado para el despliegue del servidor web de desarrollo *Tomcat*.

4.5 *Tomcat*: servidor para aplicaciones.

Apache Tomcat es un software desarrollado con *Java* (con lo cual puede funcionar en cualquier sistema operativo, con su máquina virtual *java* correspondiente) que sirve como servidor web por sí mismo, ya que trae incluido el compilador *Jasper*, que compila JSP's convirtiéndolas en *servlets*, combinado con el servidor *Web Apache*. A partir de la versión 4.0, *Tomcat* utiliza el contenedor de *servlets Catalina*, capaz de recibir las peticiones que se realizan a través de la aplicación de cardiología y redireccionar estas peticiones a un objeto *Servlet*.

Como ya indicamos en capítulos anteriores, el marco de trabajo *Openxava* tiene incluido un servidor *Tomcat* listo para usar. Veamos qué contiene el directorio raíz del servidor en el siguiente apartado.

4.5.1 Arquitectura del servidor *Tomcat*.

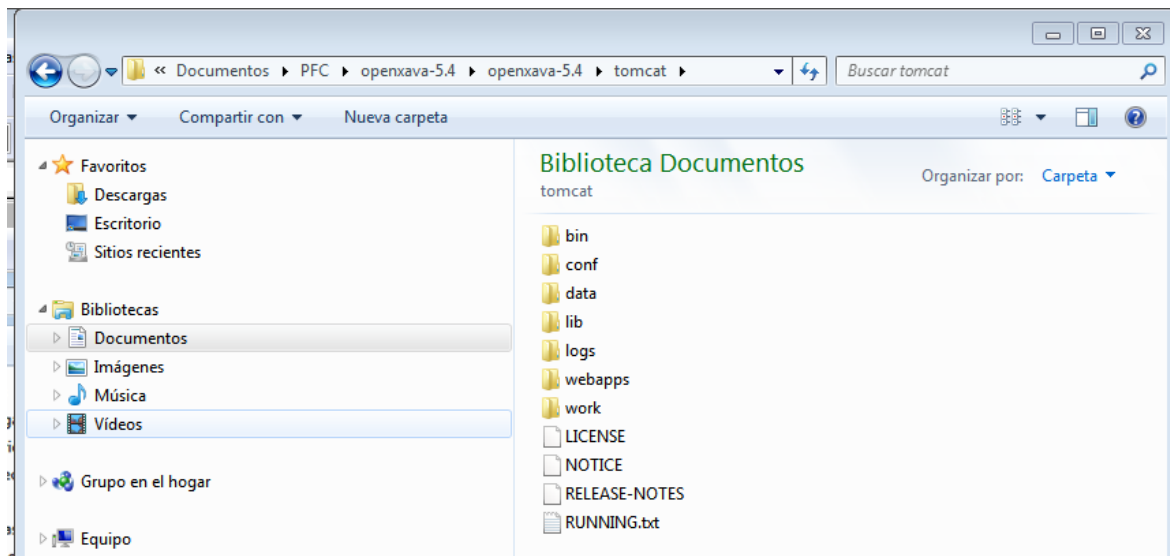
El funcionamiento del servidor *Apache Tomcat* para nuestra aplicación es el siguiente:

1. El usuario de la aplicación, a través del navegador (lado cliente) pide una página al servidor *Tomcat* contenedor de *servlets*.
2. Este contenedor delega la petición a un *servlet Catalina* en particular elegido de entre los *servlets* que contiene.
3. Este *servlet* escogido, que es un objeto *java*, se encarga de generar el texto de la página web que se entrega al contenedor.
4. El contenedor devuelve la página web al navegador (cliente) que la solicitó,

normalmente en HTML.

En el siguiente capítulo dedicado a la tecnología *JavaServer Pages*, se explicará gráficamente este servidor y cómo interactúa con esta tecnología para ver el funcionamiento en su conjunto.

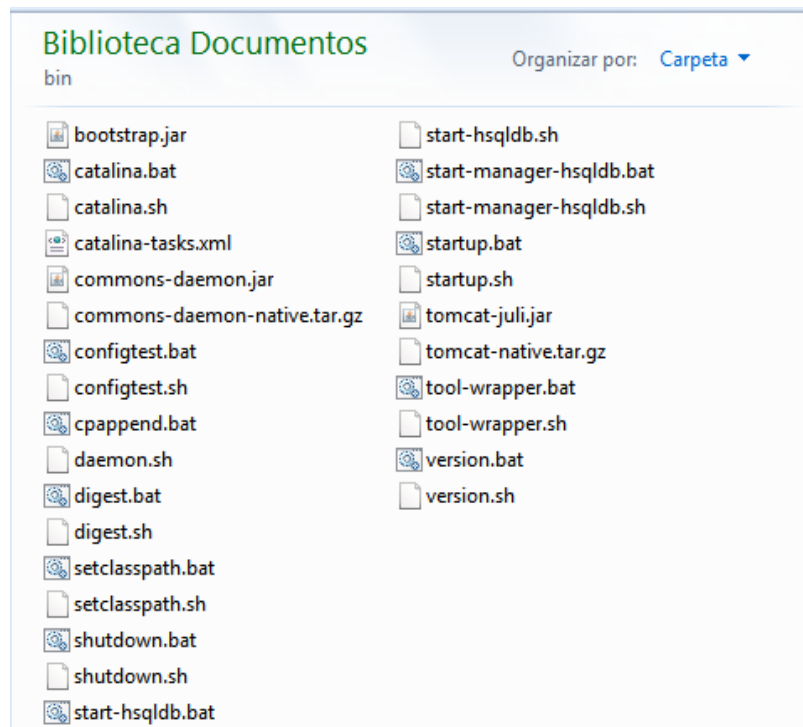
En este proyecto, el directorio raíz del servidor Tomcat utilizado es el siguiente:



Captura 50: Captura de pantalla de los principales directorios Tomcat.

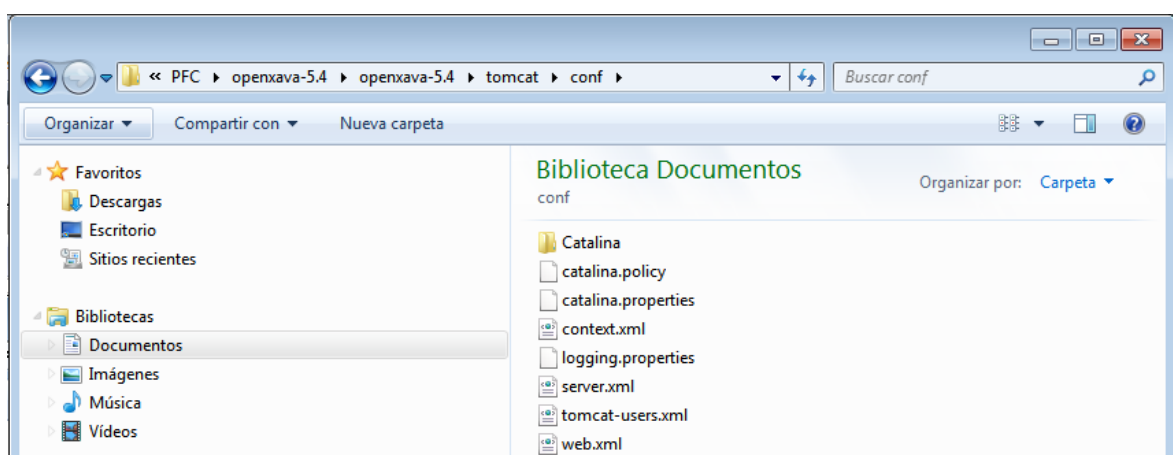
Dentro de éste, encontramos los directorios del servidor *Tomcat* [71] siguientes:

- **bin**. Inicio, apagado del servidor y otros scripts. Los archivos * .sh (para sistemas *Unix*) son duplicados funcionales de los archivos * .bat (para sistemas *Windows*). Dado que la línea de comandos de *Win32* carece de ciertas funcionalidades, hay algunos archivos adicionales aquí.



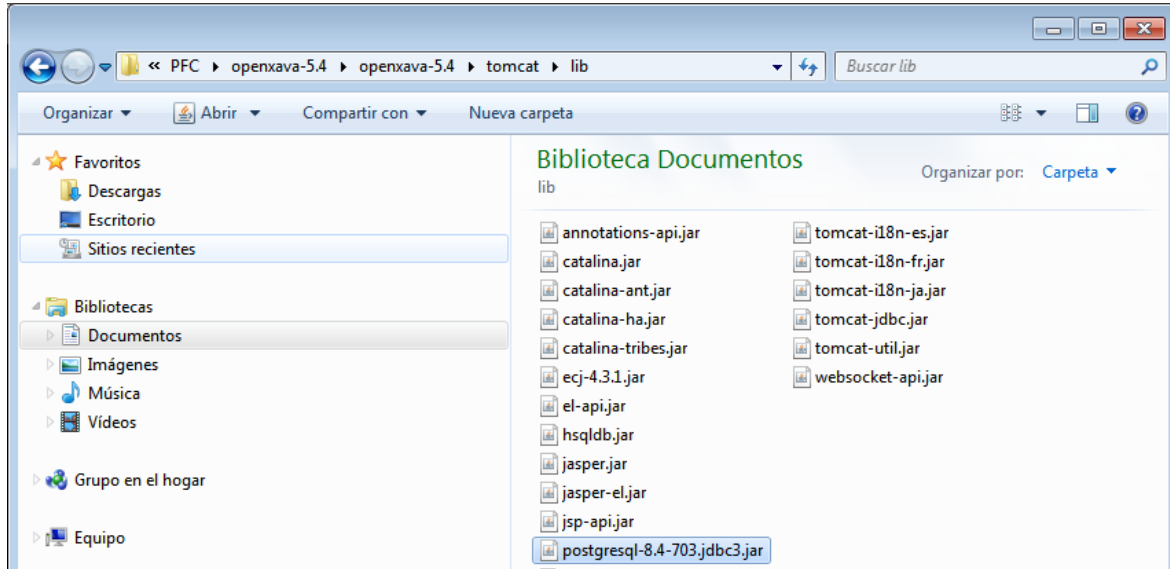
Captura 51: Captura de pantalla del contenido del directorio *bin*.

- **conf.** Archivos de configuración y archivos que definen la estructura de documentos XML relacionados (DTD's) [72]. El archivo más importante aquí es *server.xml*. Es el archivo de configuración principal del contenedor. Vemos también la ubicación del archivo *context.xml*; un archivo de configuración del servidor que modificaremos en el siguiente apartado ya que guarda relación con la fuente de datos a utilizar.



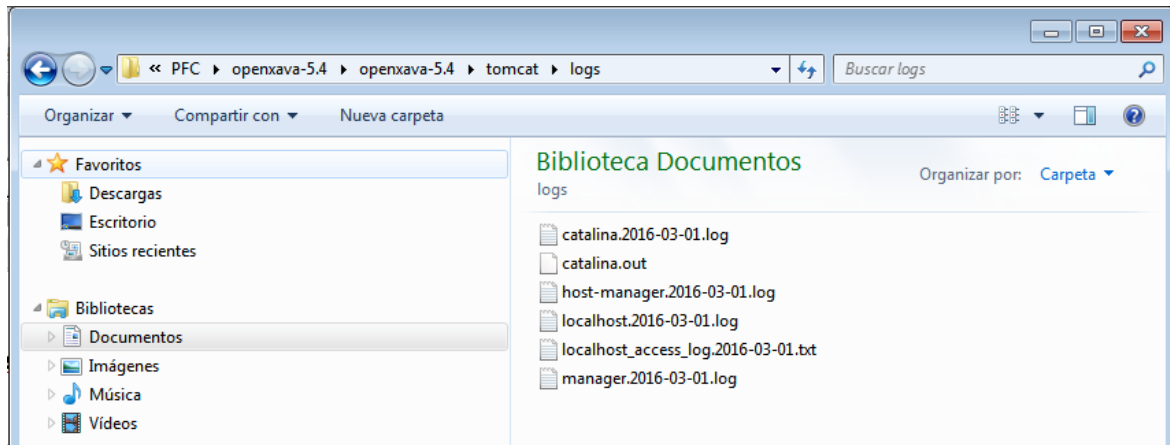
Captura 52: Captura de pantalla del contenido del directorio *conf*.

- **lib.** En este directorio encontramos los archivos adicionales, controladores, librerías que necesita el servidor. Vemos señalado el controlador *jdbc* que también añadiremos en la parte de configuración que trata sobre JDBC en el Bloque 5.



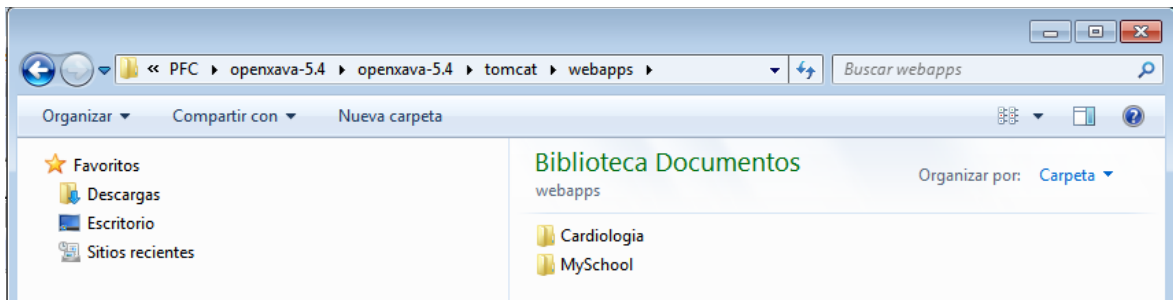
Captura 53: Captura de pantalla del contenido del directorio *lib*.

- **logs.** Los archivos de registro los localizamos aquí de forma predeterminada.



Captura 54: Captura de pantalla del contenido del directorio *logs*.

- **webapps.** Aquí se ubica la aplicación de cardiología.



Captura 55: Captura de pantalla del contenido del directorio *webapps*.

- **Variables de entorno:**

`$CATALINA_HOME`: representa la ubicación del directorio raíz de la distribución "binaria" de *Tomcat*.

`$CATALINA_BASE`: representa la ruta de configuración del servidor.

Un ejemplo de su uso es para arrancar el servidor, ejecutando el siguiente comando en Windows:

```
%CATALINA_HOME%\bin\startup.bat
```

4.6 Java Databases Connectivity (JDBC): API para acceder al sistema de gestión de BB.DD (DBMS).

La API de JDBC [73] es una API de *Java* que puede acceder a cualquier base de datos relacional.

Como se ha comentado en apartados anteriores, este conector nos ayuda a:

- Conectar a una fuente de datos, como una base de datos.
- Enviar consultas y actualizar instrucciones a la base de datos.
- Recuperar y procesar los resultados recibidos de la base de datos en respuesta a la consulta realizada.

4.6.1 Arquitectura JDBC.

La API de JDBC admite un modelo de procesamiento de tres niveles para el acceso a la

base de datos:

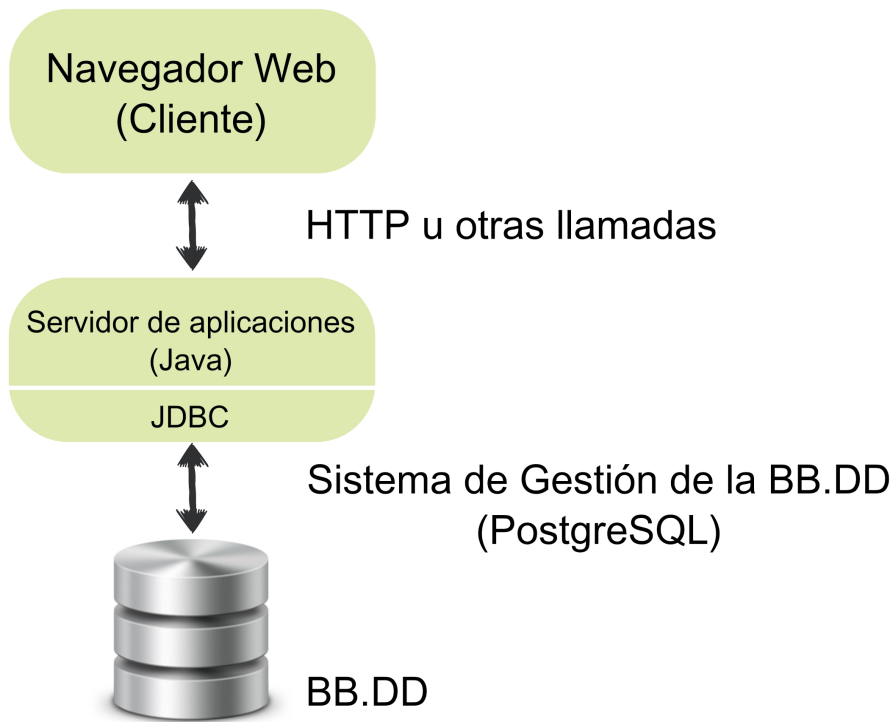


Figura 39: Representación del modelo a tres niveles del JDBC.

En el modelo de tres niveles, los comandos se envían a un "nivel medio" de servicios, que luego envía los comandos al origen de datos. La fuente de datos procesa los comandos y envía los resultados al nivel intermedio, que luego los envía al usuario de cardiología.

la API JDBC se utiliza cada vez más en el nivel medio de una arquitectura de tres niveles. Algunas de las características que convierten a JDBC en una tecnología de servidor son su compatibilidad para el agrupamiento de conexiones, las transacciones distribuidas... La API JDBC es también lo que permite el acceso a una fuente de datos desde un nivel intermedio de Java.

4.7 Java Server Pages (JSP).

En este proyecto, para crear la aplicación web de cardiología que maneja solicitudes al servidor y genera contenido dinámico recurrimos a la tecnología *JavaServer Pages (JSP)* [74], una plataforma Java. Ésta es similar a PHP [75], un lenguaje de 'scripting', o lenguaje interpretado comando a comando, de propósito general y de código abierto que está especialmente pensado para el desarrollo web y que puede ser embebido en páginas HTML, pero usando código Java que se ejecutará en el lado del servidor para

añadirse al código cliente (HTML) ya existente.

En definitiva, una página JSP es un documento textual que describe cómo crear un objeto de respuesta desde un objeto de petición para un protocolo dado. El procesamiento de la página JSP puede implicar la creación y / o el uso de otros objetos.

4.7.1 Arquitectura JSP.

Las páginas JSP y los *servlets* son denominados conjuntamente componentes web:

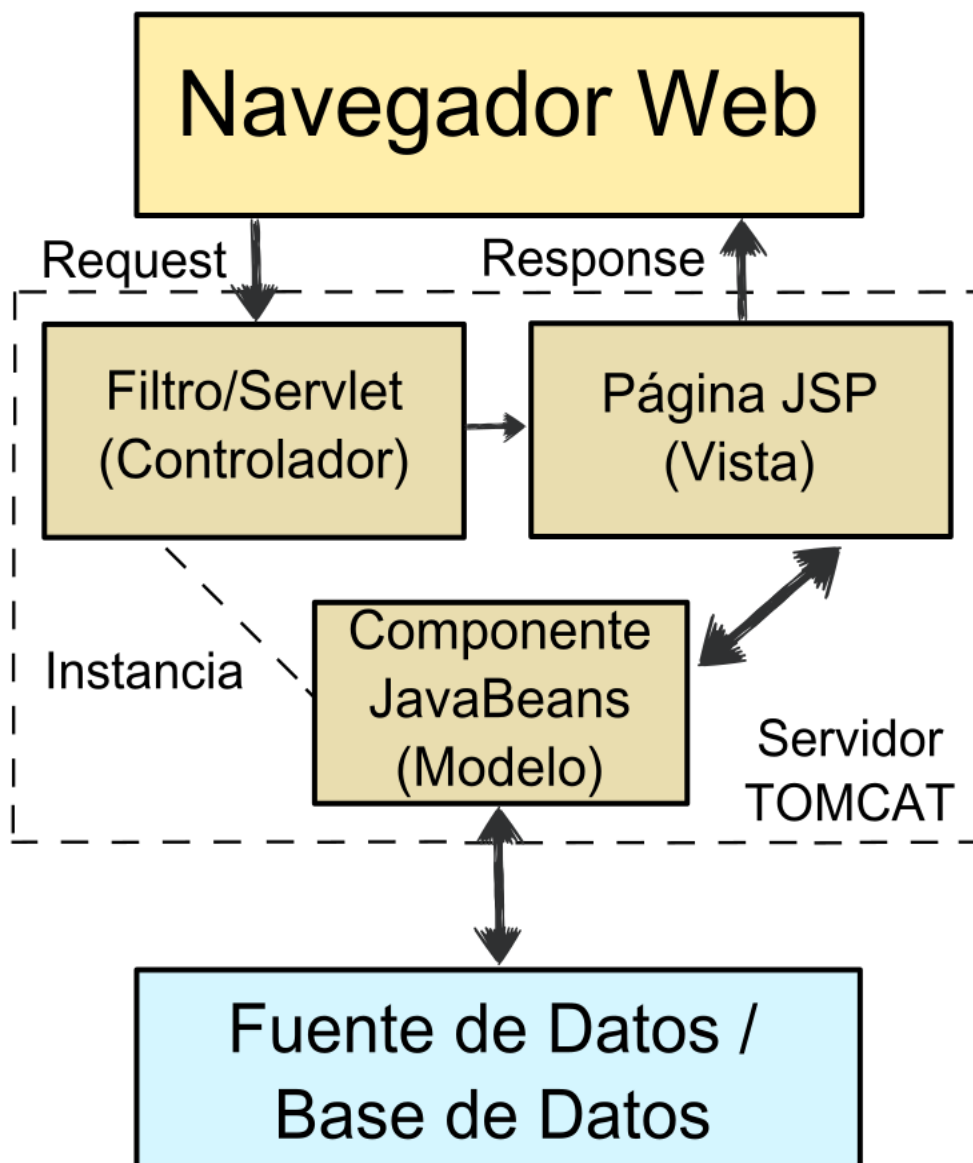


Figura 40: Arquitectura del *Model 2 de JavaServer Pages* [76].

4.7.1.1 Filtros.

Los filtros que aparecen en el nivel servidor, son componentes que nos permite interceptar el *request* antes de que llegue éste al *servlet* o JSP correspondiente, y también nos permite interceptar el *response* del *servlet* o JSP antes de que llegue al cliente.



Figura 41: Filtros que aparecen en el nivel servidor.

4.7.1.2 Servlets.

Recordemos que una de las tareas del contenedor de *servlets* catalina del servidor TOMCAT es la conexión a la BB. DD de cardiología mediante el conector JDBC.

4.7.1.3 Páginas JSP.

Son componentes textuales que pasan por dos fases: la fase de traducción y la fase de solicitud. La traducción se realiza una vez por página y la fase de solicitud se realiza una vez por solicitud.

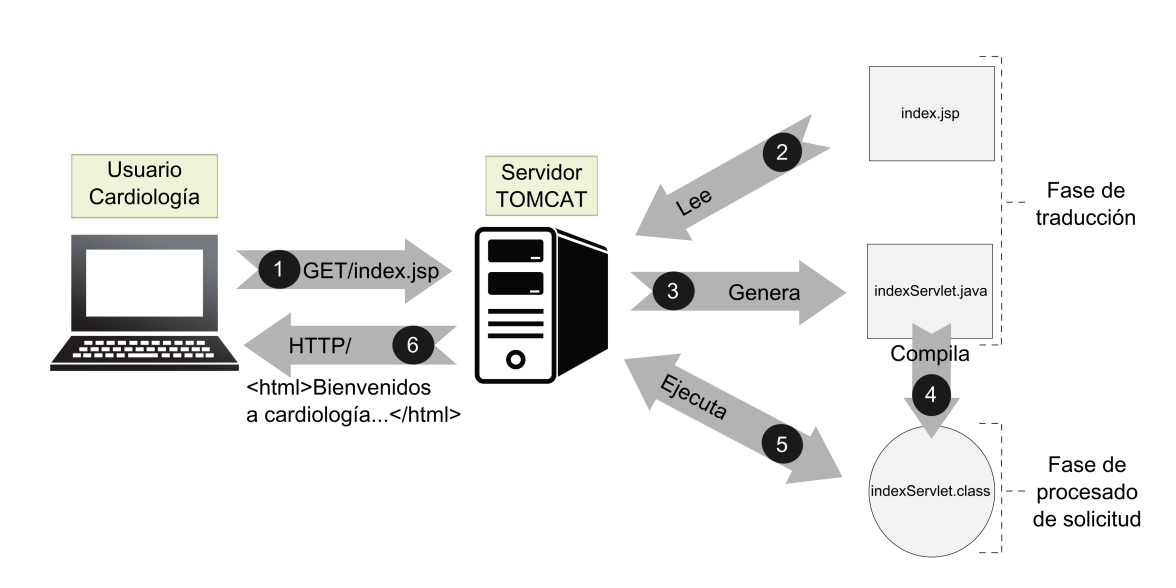


Figura 42: Representación del modelo web cliente/servidor.

En los pasos 2,3 y 4 se observa como al desplegar una JSP e iniciarse el servidor, éste crea el código *java* para un *servlet* a partir del contenido de la JSP y lo compila.

4.7.1.4 Servidor.

Para desplegar y correr *JavaServer Pages*, se requiere un servidor web compatible con contenedores *servlet* que gestionan las peticiones y generan contenido dinámico. Es por esto que es necesario el servidor *Apache Tomcat* descrito anteriormente.

4.7.2 Ventajas.

Esta tecnología permite la fácil creación de páginas web que crean contenido dinámico con máxima potencia y flexibilidad. Las ventajas que ofrece son muchas:

4.7.2.1 Portabilidad.

Las páginas JSP se pueden desplegar en cualquier plataforma, ejecutar en cualquier servidor web o servidor de aplicaciones habilitado para web, y se puede acceder desde cualquier navegador web.

4.7.2.2 Soporte de herramientas de alta calidad.

La independencia de la plataforma permite al desarrollador elegir las mejores herramientas.

4.7.2.3 Separación de roles.

Dependiendo de qué componentes se desarrolle o modifique el usuario tendrá un rol u otro. En la especificación se habla de hasta seis tipos: desplegados, autores...

4.7.2.4 Reutilización de componentes y bibliotecas de etiquetas.

Con el objetivo de simplificar el desarrollo y la composición de la página JSP, la tecnología de JavaServer Pages enfatiza el uso de componentes reutilizables tales como componentes JavaBeans, componentes Enterprise JavaBeans y etiquetas. Veamos algunos ejemplos de algunos tipos que hemos usado en este proyecto.

4.7.2.4.1 Variables y objetos implícitos.

Éstos están predefinidos con nombres estándares, y ya están listos para usarse, sin tener que ser declarados ni configurados.

Variable	Clase
pageContext	javax.servlet.jsp.PageContext
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
session	javax.servlet.http.HttpSession
config	javax.servlet.ServletConfig
application	javax.servlet.ServletContext
out	javax.servlet.jsp.JspWriter
page	java.lang.Object
exception	java.lang.Exception

Figura 43: Objetos y variables privilegiadas ya incluidas listas para usar [77].

4.7.2.4.2 Directivas.

Etiquetas a partir de las cuales se genera información que puede ser utilizada por el motor de JSP .

- **include** : Incluye el contenido de un fichero en la página mediante el atributo *file*.
`<%@ include file="cabecera.html" %>`
- **taglib** : Importa bibliotecas de etiquetas (Tag Libraries)
`<%@ taglib uri="/tags/struts-html" prefix="html" %>`
- **page** : Especifica atributos relacionados con la página a procesar. Los atributos son:

Atributo	Sintaxis	Utilización
import	<code><%@ page import="class; class" %></code>	Importa clases y paquetes Java para ser utilizadas dentro del fichero JSP.
session	<code><%@ page session="false" %></code>	Especifica si utiliza los datos contenidos en sesión; por defecto "true".
contentType	<code><%@ page contentType="class; class" %></code>	Especifica el tipo MIME del objeto "response"; por defecto "text/html, charset=ISO-8859-1".
buffer	<code><%@ page buffer="12KB" %></code>	Buffer utilizado por el objeto writer "out"; puede tomar el valor de "none"; por defecto "8KB".
errorPage	<code><%@ page errorPage="/path_to_error_page" %></code>	Especifica la ruta de la página de error que será invocada en caso de producirse una excepción durante la ejecución de este fichero JSP.
isErrorPage	<code><%@ page isErrorPage="true" %></code>	Determina si este fichero JSP es una página que maneja excepciones. Únicamente a este tipo de páginas pueden acceder a la variable implícita "exception", que contiene la excepción que provocó la llamada a la página de error.

Figura 44: Directivas disponibles para JSP [77].

Por ejemplo, si analizamos el contenido del fichero *index.jsp* ubicado en la ruta *...openxava-5.4\workspace\Cardiologia\web\naviox\index.jsp*, vemos como aparece código JSP embebido en código HTML, aparecen distintas directivas, y objetos implícitos:

```

1  <% Servlets.setCharacterEncoding(request, response) %>
2
3  <%@include file="../xava/imports.jsp"%>
4
5  <%@page import="org.openxava.web.servlets.Servlets"%>
6  <%@page import="org.openxava.util.Locales"%>
7  <%@page import="com.openxava.naviox.web.NaviOXStyle"%>
8
9  <jsp:useBean id="context" class="org.openxava.controller.ModuleContext" scope="session"/>
10 <jsp:useBean id="modules" class="com.openxava.naviox.Modules" scope="session"/>
11
12 <%
13 String app = request.getParameter("application");
14 String module = context.getCurrentModule(request);
15 Locales.setCurrent(request);
16 String retainOrder = request.getParameter("retainOrder");
17 boolean retainOrder = "true" equals(retainOrder);
18 modules.setCurrent(request.getParameter("application"), request.getParameter("module"), retainOrder);
19 String oxVersion = org.openxava.controller.ModuleManager.getVersion();
20 %>
21
22 <!DOCTYPE html %>
23
24 <head %>
25 <title><%=modules.getCurrentModuleDescription(request) %></title>
26 <link href="<%=request.getContextPath() %>/naviox/style/naviox.oss?ox=<%=oxVersion %>" rel="stylesheet" type="text/oss">
27 <link rel="stylesheet" href="<%=request.getContextPath() %>/xava/style/materialdesignicons.oss?ox=<%=oxVersion %>">
28 <script type="text/javascript" src="<%=request.getContextPath() %>/xava/js/dwr-engine.js?ox=<%=oxVersion %>"></script>
29 <script type="text/javascript" src="<%=request.getContextPath() %>/dwr/interface/Modules.js?ox=<%=oxVersion %>"></script>
30 <script type="text/javascript" src="<%=request.getContextPath() %>/dwr/interface/Folders.js?ox=<%=oxVersion %>"></script>
31 </head %>
32
33 <body <%=NaviOXStyle.getBodyClass(request) %> %>
34
35 <div id="main navigation">

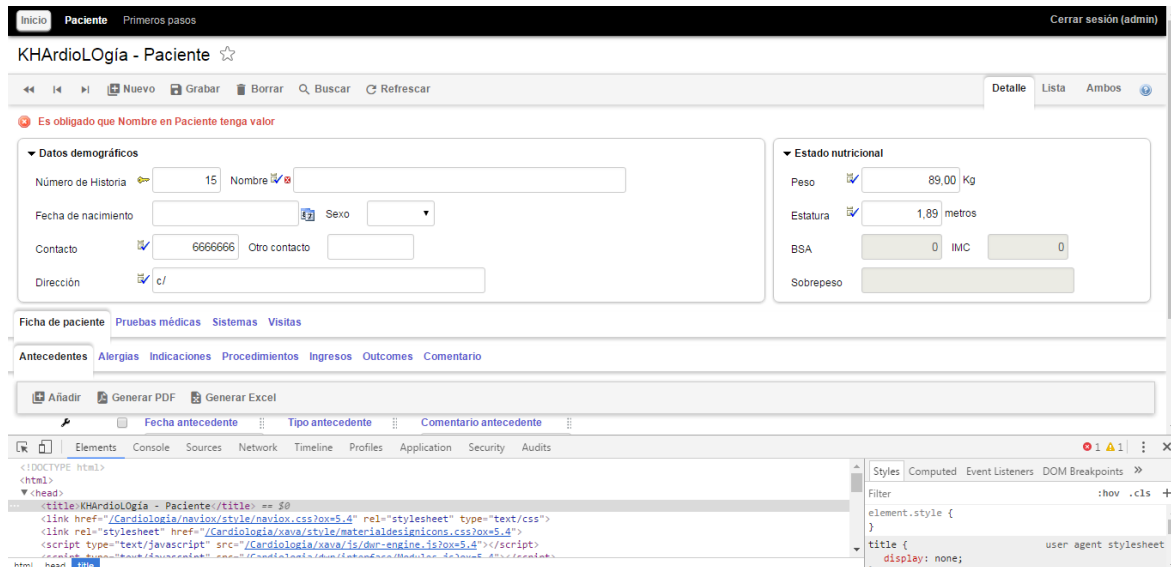
```

Captura 56: Captura de pantalla de código del fichero *index.jsp*.

- ➔ En la línea 5, con la directiva `page` se importa el paquete propio de *Openxava* ya creado por este marco de trabajo.
- ➔ En la línea 25, se embebe código JSP, entre la etiqueta título `<title>` y fin de título `</title>`.

</title> de código HTML para poner como título dinámico de página el módulo de aplicación donde se encuentra.

- ➔ En la línea 25, se hace uso de la variable *request*, la cual representa la petición que es recibida en la página, y con ello los parámetros y valores que llegan.



Captura 57: Captura de pantalla de la página que representa el código anterior.

Al inspeccionar con las herramientas del navegador utilizado, vemos que efectivamente el título de la página es el nombre del módulo en el que se encuentra **Paciente**, valor recogido gracias al código JSP.

4.7.2.4.3 Declaraciones.

Gracias a ellas, podemos declarar variables, funciones y datos estáticos.

4.7.2.4.4 Scriptlets.

Código Java incrustado entre los elementos estáticos de la página.


```

1  <% Servlets.setCharacterEncoding(request, response); %>
2
3  <%@include file="../xava/imports.jsp"%>
4
5  <%@page import="org.openxava.web.servlets.Servlets"%>
6  <%@page import="org.openxava.util.Locales"%>
7  <%@page import="com.openxava.naviox.web.NavioxStyle"%>
8
9  <jsp:useBean id="context" class="org.openxava.web.servlets.Servlets" scope="session"/>
10 <jsp:useBean id="modules" class="org.openxava.controller.ModuleManager" scope="session"/>
11
12 <%
13 String app = request.getParameter("application");
14 String module = context.getCurrentModule(request);
15 Locales.setCurrent(request);
16 String sretainOrder = request.getParameter("retainOrder");
17 boolean retainOrder = "true".equals(sretainOrder);
18 modules.setCurrent(request.getParameter("application"), request.getParameter("module"), retainOrder);
19 String oxVersion = org.openxava.controller.ModuleManager.getVersion();
20 %>
21
22 <DOCTYPE html>
23
24 <head>
25 <title><%=modules.getCurrentModuleDescription(request) %></title>
26 <link href="<%=request.getContextPath() %>/naviox/style/naviox.css?ox=<%=oxVersion %>" rel="stylesheet" type="text/css">
27 <link rel="stylesheet" href="<%=request.getContextPath() %>/xava/style/materialdesignicons.css?ox=<%=oxVersion %>">
28 <script type="text/javascript" src="<%=request.getContextPath() %>/xava/js/dwr-engine.js?ox=<%=oxVersion %>"></script>
29 <script type="text/javascript" src="<%=request.getContextPath() %>/dwr/interface/Modules.js?ox=<%=oxVersion %>"></script>
30 <script type="text/javascript" src="<%=request.getContextPath() %>/dwr/interface/Folders.js?ox=<%=oxVersion %>"></script>
31 </head>
32
33 <body <%=NavioxStyle.getBodyClass(request) %>
34
35 <div id="main_navigation">
36 <!--include page=mainNavigation.jsp-->

```

Captura 58: Captura de pantalla del fichero *index.jsp* con *scriptlets*.

En la anterior captura de pantalla, la banda azul engloba todo el código Java incrustado. En él, se declaran y dan valor a variables tipo string y booleanas:

```

<%
String app = request.getParameter("application");
String module = context.getCurrentModule(request);
Locales.setCurrent(request);
String sretainOrder = request.getParameter("retainOrder");
boolean retainOrder = "true".equals(sretainOrder);
modules.setCurrent(request.getParameter("application"),
request.getParameter("module"), retainOrder);
String oxVersion = org.openxava.controller.ModuleManager.getVersion();
%>

```

4.7.2.4.5 Expresiones.

Se evalúan dentro del servlet.

```

25 String viewObject = request.getParameter("viewObject");
26 viewObject = (viewObject == null || viewObject.equals(""))?"xava_view":viewObject;
27 View view = (View) context.get(request, viewObject);
28 String moduleName = view.getValueString("module.name");
29 String applicationName = request.getParameter("application");
30 MetaModule module = MetaApplications.getMetaApplication(applicationName).getMetaModule(moduleName);
31 int i=0;
32 <%>
33 <table width="100%"><tr>
34 <%>
35 for (Object ocontroller: module.getControllersNames()) {
36 MetaController controller = MetaControllers.getMetaController((String) ocontroller);
37 for (Object oaction: controller.getAllNotHiddenMetaActions()) {
38 MetaAction action = (MetaAction) oaction;
39 if (action.getMetaController().getName().equals("Navigation")) continue;
40 String checked = actions.contains(action.getQualifiedName())?"checked='true'":"";
41 <%>
42 <td>
43 <INPUT name="<%=action.getQualifiedName()%>" class="<%=style.getEditor()%>"
44 type="checkbox"
45 value="<%=action.getQualifiedName()%>"
46 <%=checked%>
47 <%=disabled%>
48 <%=script%>
49 />
50 <%=action.getLabel()%>
51 <%= if (++i % 4 == 0) { %></tr><tr><% } %>
52 </td>
53 <%>
54 }
55 }
56 <%>
57 </tr></table>
58 <%>
59 if (!editable) {

```

Expresiones

Captura 59: Expresiones JSP.

4.7.2.4.6 Etiquetas JSP.

Éstas pertenecen a la especificación JSP y facilitan una funcionalidad básica.

Funcionalidad a nivel de página

Etiqueta	Funcionalidad
<code><jsp:forward></code>	Redirige la request a otra URL.
<code><jsp:include></code>	Incluye el texto de un fichero dentro de la página.
<code><jsp:plugin></code>	Descarga un plugin de Java (una applet o un Bean).

Componentes JavaBean

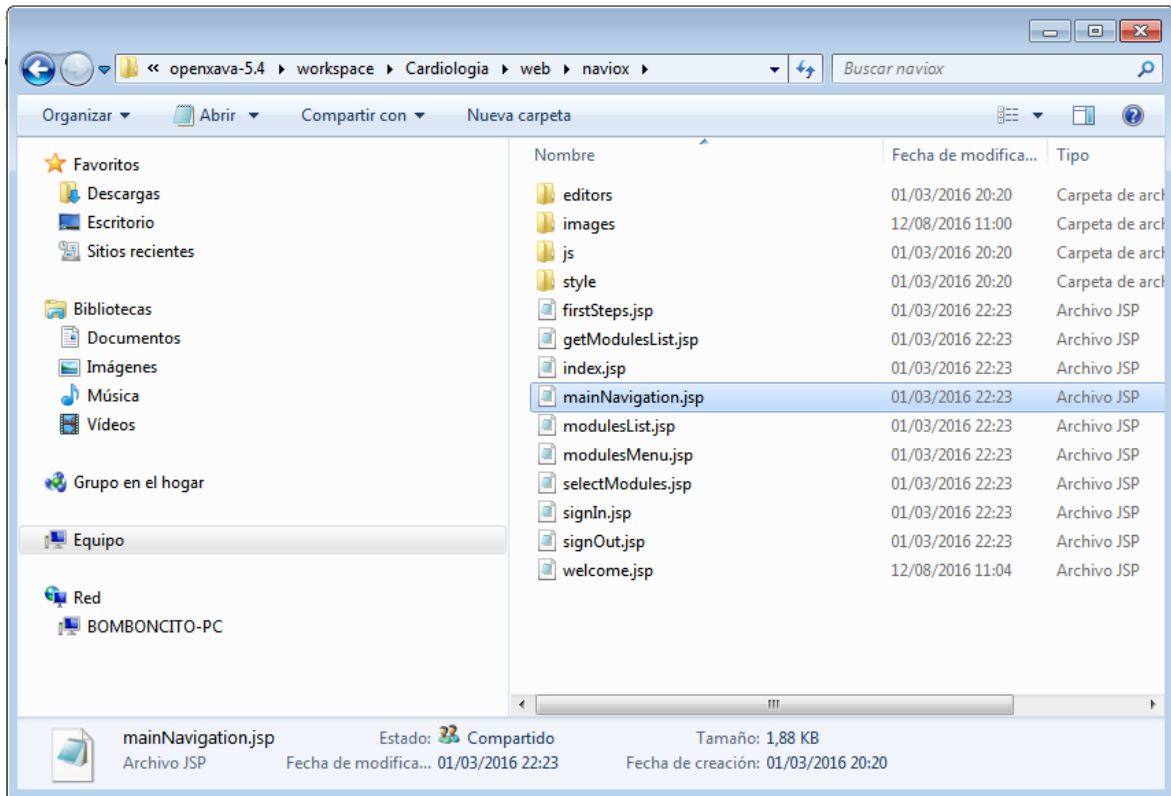
Etiqueta	Funcionalidad
<code><jsp:useBean></code>	Permite manipular un Bean (si no existe, se creará el Bean) especificando su ámbito (scope), la clase y el tipo.
<code><jsp:getProperty></code>	Obtiene la propiedad especificada de un Bean previamente declarado y la escribe en el objeto response.
<code><jsp:setProperty></code>	Establece el valor de una propiedad de un Bean previamente declarado.

Figura 45: Tipos de etiquetas JSP.

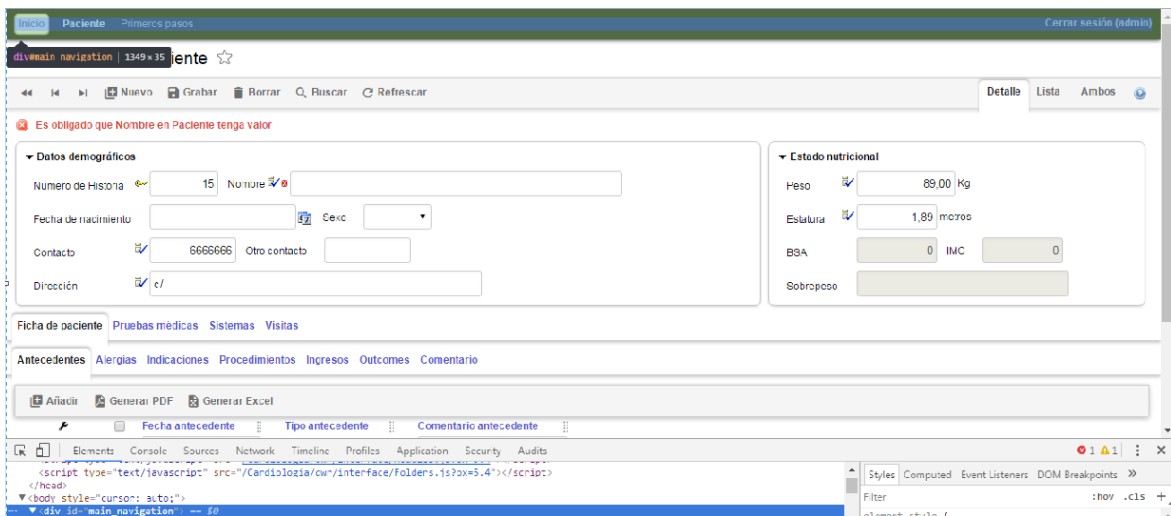
```
22 <!DOCTYPE html>
23
24
25 <title><%=modules.getCurrentModuleDescription(request) %></title>
26 <link href="<%=request.getContextPath() %>/naviox/style/naviox.css?ox=<%=oxVersion %>" rel="stylesheet" type="text/css">
27 <link rel="stylesheet" href="<%=request.getContextPath() %>/xava/style/materialdesignicons.css?ox=<%=oxVersion %>">
28 <script type="text/javascript" src="<%=request.getContextPath() %>/xava/js/dwr-engine.js?ox=<%=oxVersion %>"></script>
29 <script type="text/javascript" src="<%=request.getContextPath() %>/dwr/interface/Modules.js?ox=<%=oxVersion %>"></script>
30 <script type="text/javascript" src="<%=request.getContextPath() %>/dwr/interface/Folders.js?ox=<%=oxVersion %>"></script>
31 </head>
32
33 <body <%=NavioxStyle.getBodyClass(request) %>>
34
35 <div id="main_navigation">
36 <jsp:include page="mainNavigation.jsp"/>
37 </div>
38
39 <table width="100%">
40 <tr>
41 <td id="modules_list">
42 <div id="modules_list_popup">
43 
44 <div id="modules_list_outbox">
45 <table id="modules_list_box">
46 <tr id="modules_list_content">
47 <td>
48 <jsp:include page="modulesMenu.jsp"/>
49 </td>
50 </tr>
51 </table>
52 </div>
53 </div>
54 </td>
55 </tr>
56 </table>
```

Captura 60: Captura de pantalla de fragmento de código *index.jsp*.

En la anterior fotografía, en la línea de código resaltada, se está incluyendo el contenido del fichero *mainNavigation.jsp* dentro de esta página *index.jsp*. Este fichero contiene el código del menú principal.



Captura 61: Captura de pantalla del directorio *naviox* donde figura el fichero que se está incluyendo.



Captura 62: Inspeccionando elemento con el navegador web.

Al inspeccionar con las herramientas del navegador utilizado, vemos que efectivamente el menú **Inicio** de la aplicación web es el arriba señalado, con `id=main_navigation`, descrito en el fichero *mainNavigation.jsp*.

4.7.2.4.7 Soporte para acciones, expresiones y secuencias de comandos.

Admite elementos de *scripting* y acciones. Gracias a la especificación JSP 2.0 podemos

escribir acciones utilizando la tecnología JSP directamente como ya vimos anteriormente.

Una vez revisada toda la teoría referida a la persistencia de la BB. DD de la aplicación, servidor de aplicaciones utilizado, conectividad a la fuente de datos para acceder a los datos, y cómo generamos el contenido dinámico de esta aplicación web, se introduce al lector en la teoría necesaria para saber cómo desarrollar el código del lado del cliente de la aplicación web; clases del modelo de los datos, los formularios web y sus estilos, la comunicación asíncrona con el servidor en segundo plano para mejorar la experiencia del usuario, y demás tecnologías.

4.8 El concepto de componente de negocio.

OpenXava, el marco de trabajo de desarrollo ágil de aplicaciones escogido para este proyecto, está basado en un refinamiento de conceptos preexistentes, algunos populares y otros no tanto. El más popular es el Desarrollo Dirigido por el Modelo (Model-Driven Development, MDD), que *OpenXava* usa de una manera ligera. El otro concepto, el Componente de Negocio, es raíz y principio básico de *OpenXava*, además de ser la alternativa opuesta a MVC.

4.8.1 Desarrollo dirigido por el modelo ligero.

Básicamente, MDD establece que únicamente se ha de desarrollar la parte del modelo de una aplicación, y el resto se generará a partir de este modelo. A continuación se explica de manera gráfica:

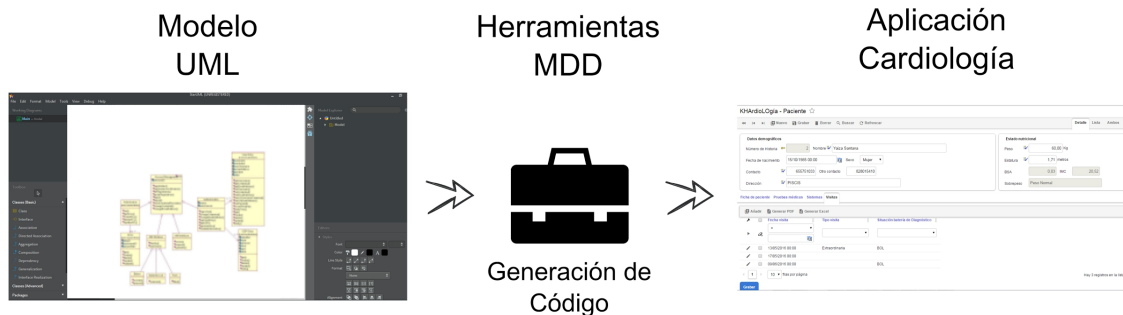


Figura 46: Desarrollo dirigido por el modelo.

En el contexto de MDD el modelo es el medio para representar los datos y la lógica de la aplicación. Puede ser, bien mediante una notación gráfica, como UML, o bien mediante una notación textual como un Lenguaje Específico del Dominio (Domain-Specific Language, DSL).

Por desgracia, el uso de MDD [78] es muy complejo. Requiere de una gran cantidad de tiempo, pericia y herramientas. Aun así la idea tras MDD sigue siendo muy buena, por lo tanto OpenXava toma esa idea de una manera simplificada. Usa simples clases de Java con anotaciones para definir el modelo, y no usa generación de código.

	Definición del modelo	Generación de la aplicación
MDD clásico	UML/DSL	Generación de código
OpenXava	Simple clases Java	Dinámicamente en tiempo de ejecución

Figura 47: Comparación Openxava y MDD.

Podemos decir pues, que *OpenXava* es un marco de trabajo ligero dirigido por el modelo.

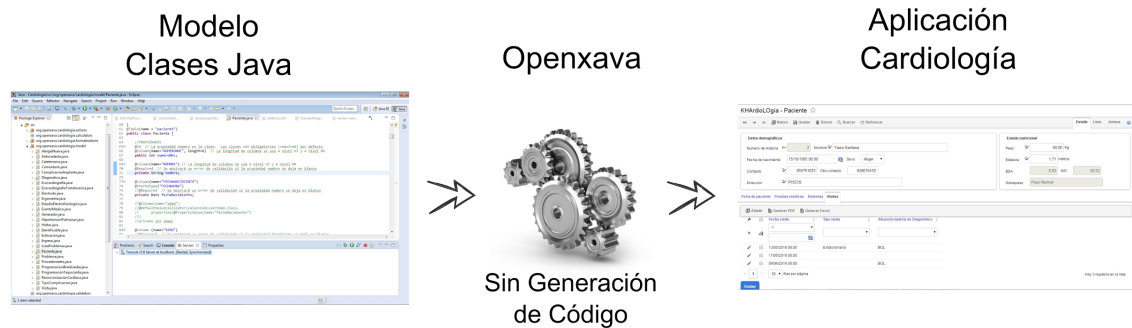


Figura 48: Esquema simple de cómo se desarrolla la aplicación de cardiología.

Así, a modo de resumen, podemos decir que, a partir de clases Java simples, obtienes una aplicación lista para usar. El siguiente apartado sobre el concepto de componente de negocio revelará algunos detalles importantes sobre la naturaleza de estas clases.

4.8.2 Componente de negocio.

Un Componente de Negocio consiste en todos los componentes software relacionados con un concepto de negocio. Los componentes de negocio son tan sólo una forma de organizar el software. La otra forma de organizar software es MVC (*Model-View Controller*), dónde se clasifica el código por datos (modelo), interfaz de usuario (vista) y lógica (controlador).

Con el enfoque Modelo Vista Controlador se organizan los componentes software en una aplicación MVC. Todos los componentes para la interfaz de usuario de la aplicación, tales como páginas web con código HTML y etiquetas especiales y código *Java* (JSP) [79], JSF [80], la biblioteca gráfica para *Java*; *Swing* [81], etc. están en el mismo lugar, la capa de la vista. Lo mismo ocurre para el modelo y el controlador.

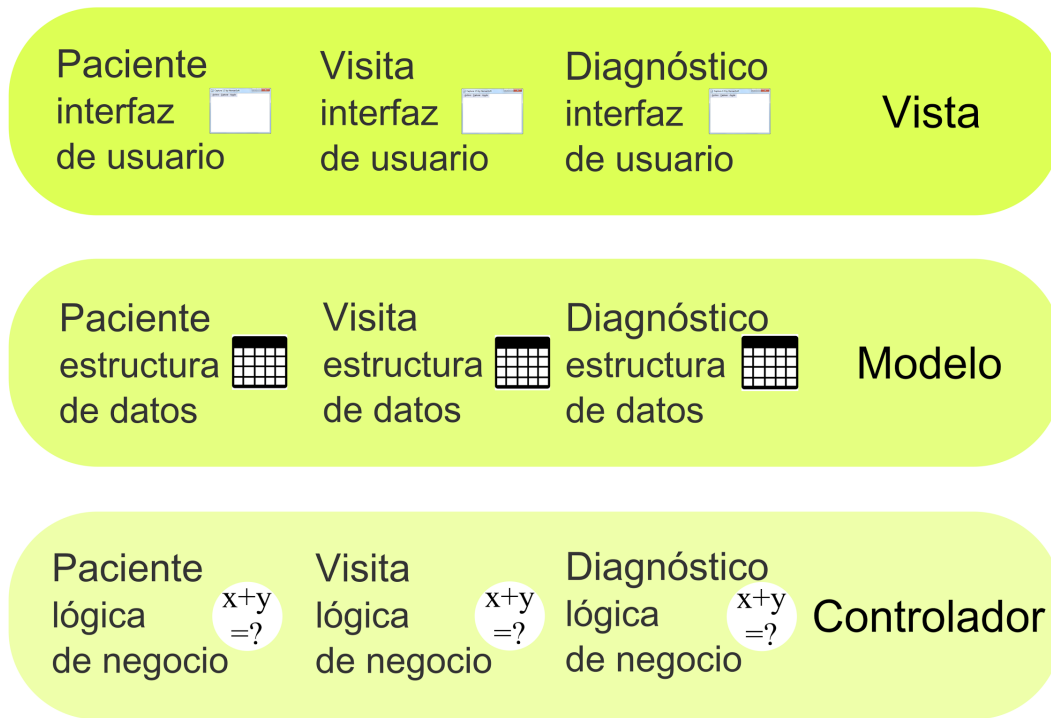


Figura 49: Modelo Vista Controlador (MVC).

Esto contrasta con una arquitectura basada en componentes de negocio donde los componentes software se organizan alrededor de los conceptos de negocio.

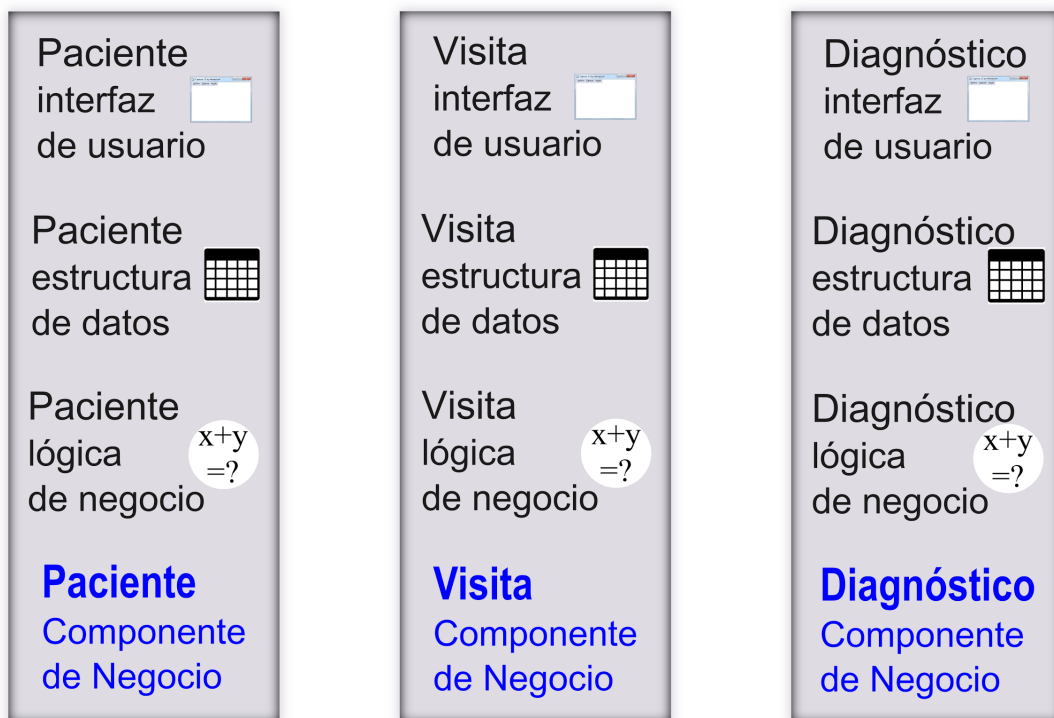
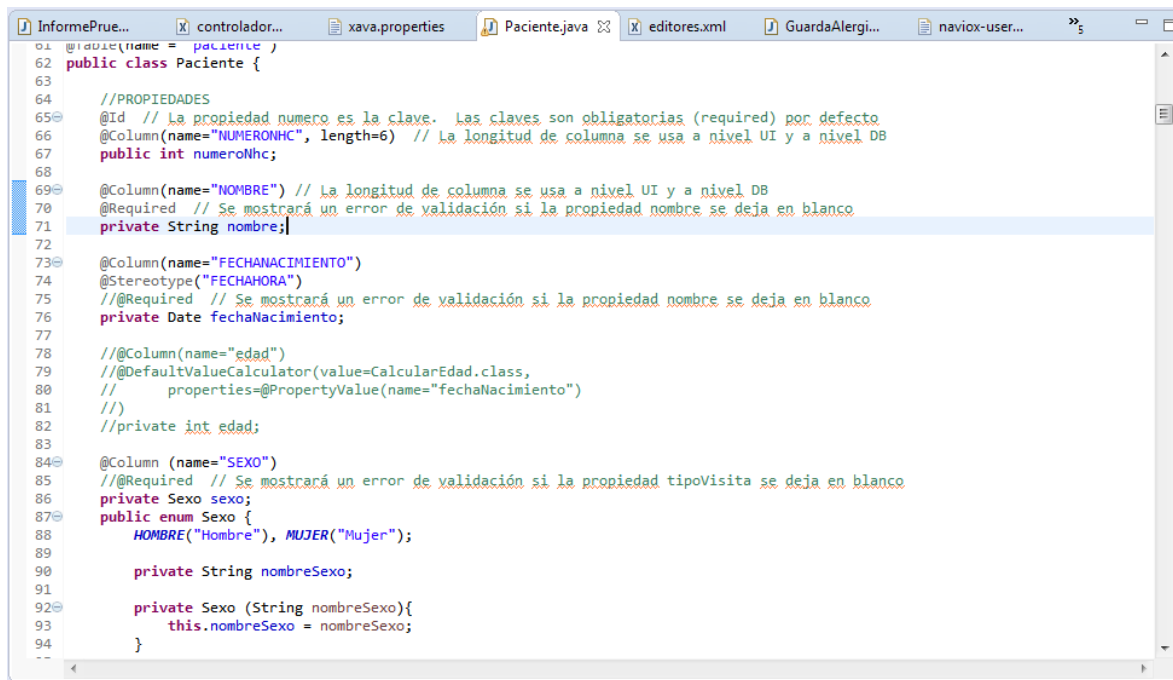


Figura 50: Componente de negocio.

Aquí, todos los componentes software acerca del concepto de Paciente, como la interfaz de usuario, acceso a base de datos, lógica de negocio, etc. están en un mismo lugar.

Al tener que hacer cambios en la estructura de los datos y la lógica de negocio entonces la opción de los componentes de negocio frente a MVC es muy práctica, porque todas las cosas que necesitas modificar cuando haces un cambio están en el mismo ámbito y no esparcidas por multitud de archivos.

La pieza básica para desarrollar aplicaciones *OpenXava* es el componente de negocio, y la forma de definir un componente de negocio en *OpenXava* es usando una simple clase Java con anotaciones:



```
01 @table(name = "paciente")
02 public class Paciente {
03
04     //PROPIEDADES
05     @Id // La propiedad numero es la clave. Las claves son obligatorias (required) por defecto
06     @Column(name="NUMERONHC", length=6) // La longitud de columna se usa a nivel UI y a nivel DB
07     public int numeroNhc;
08
09     @Column(name="NOMBRE") // La longitud de columna se usa a nivel UI y a nivel DB
10     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
11     private String nombre;
12
13     @Column(name="FECHANACIMIENTO")
14     @Stereotype("FECHAHORA")
15     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
16     private Date fechaNacimiento;
17
18     // @Column(name="edad")
19     // @DefaultValueCalculator(value=CalcularEdad.class,
20     //     properties=@PropertyValue(name="fechaNacimiento"))
21     //)
22     //private int edad;
23
24     @Column(name="SEXO")
25     @Required // Se mostrará un error de validación si la propiedad tipoVisita se deja en blanco
26     private Sexo sexo;
27     public enum Sexo {
28         HOMBRE("Hombre"), MUJER("Mujer");
29
30         private String nombreSexo;
31
32         private Sexo (String nombreSexo){
33             this.nombreSexo = nombreSexo;
34         }
35     }
36 }
```

Captura 63: Fragmento de código de la clase java Paciente.

Como puede observarse, todo acerca del concepto de negocio sobre el paciente se define en un único lugar, la clase *Paciente.java*. En esta clase se definen elementos de base de datos, estructura de los datos, lógica de negocio, interfaz de usuario, validación, etc. Esto se hace usando la facilidad de metadatos de Java, las famosas anotaciones que explicaremos más adelante en el código de las clases *Java* del proyecto “cardiología”.

En definitiva, la ventaja de usar *Openxava* y los componentes de negocio es que se puede producir una aplicación funcional a partir de componentes de negocio.

4.9 *Openxava*: marco de trabajo.

Como venimos indicando, los componentes de negocio son los elementos básicos para construir la aplicación web de cardiología. Definiendo únicamente estos componentes de negocio tendríamos la aplicación básica, aunque no con todas las funcionalidades que se requieren ya descritas en los requerimientos pedidos por el HUGCDN, como por ejemplo, que los dos médicos que figuran en una prueba médica nunca sean la misma persona.

Para desarrollar todas las funcionalidades para la aplicación de este proyecto, hemos recurrido a otros elementos como los módulos, editores, validadores y calculadores.

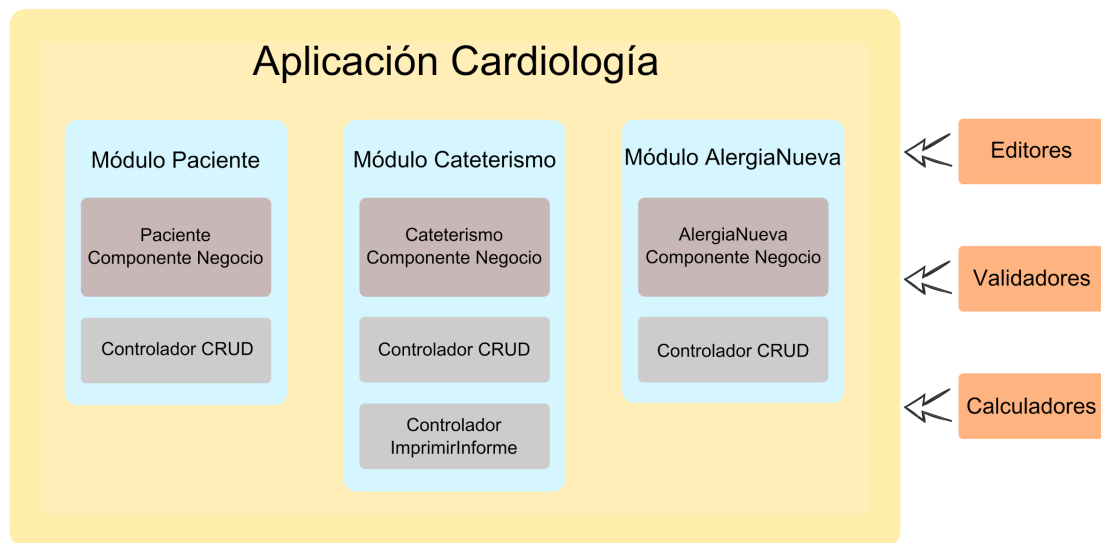


Figura 51: Representación de algunos elementos de la aplicación de cardiología.

A continuación se definen los diferentes elementos que nos encontraremos en la aplicación del HUGCDN.

4.9.1 Elementos *Openxava*.

4.9.1.1 Componentes de negocio.

Clases de Java que describen los componentes de negocio de la aplicación en todos sus aspectos.

Por ejemplo, se añade el código del componente de negocio *AlergiaNueva.java*. Este código representa una alergia que pueda padecer el paciente, y la relación que guarda el paciente con ella es de 1 a muchos, es decir el paciente puede tener 0, 1 o muchas alergias.

```
package org.openxava.cardiologia.model;
```

```
import javax.persistence.*;
```

```

import org.openxava.annotations.*;
import org.openxava.util.*;

@Entity
@Table(name = "alergianueva")
public class AlergiaNueva extends Identificable{

    //PROPIEDADES
    @Required
    @Column(name="NUEVAALERGIA")
    private String nuevaAlergia;

    // @ReadOnly
    //private String userId;

    //REFERENCIAS
    @ManyToOne // La referencia tiene que tener valor siempre. No permite
AlergiaNueva sin Paciente
    @JoinColumn(
        name="numeroNhc",
        nullable = true,
        foreignKey = @ForeignKey(name = "fk_alergianueva_numeronhc"))
        // fkpaciente es la columna para la clave foránea
    private Paciente alergiasNuevasLocal; // Una referencia Java convencional

    //MÉTODOS
    public Paciente getAlergiasNuevasLocal() {
        return alergiasNuevasLocal;
    }

    public void setAlergiasNuevasLocal(Paciente alergiasNuevasLocal) {
        this.alergiasNuevasLocal = alergiasNuevasLocal;
    }

    public void setNuevaAlergia(String nuevaAlergia) {
        this.nuevaAlergia = nuevaAlergia;
    }
}

```

```

    }

    public String getNuevaAlergia() {
        return nuevaAlergia;
    }

    //public String getUserId() {
    //    return userId;
    //}
}

```

4.9.1.2 Módulos.

Un módulo engloba al componente de negocio y sus controladores. En la barra de menú principal aparecería un enlace a cada módulo (clase) de la aplicación que nos llevaría a su página correspondiente.

Se muestra el módulo que se genera al crear la clase Paciente:



Captura 64: Barra de módulos que se generan automáticamente en *Openxava*.

4.9.1.3 Controladores.

Son clases con lógica que representan una colección de acciones y por lo tanto, el comportamiento de la aplicación. Éstos se representan para el usuario de la aplicación de cardiología mediante botones o vínculos que él puede pulsar, y tras ello, se realiza la acción.

4.9.1.4 Editores.

Son componentes de la interfaz de usuario para definir la forma en que los miembros de un componente de negocio son visualizados y editados. Es una manera de personalizar la generación de la interfaz de usuario.

OpenXava usa editores predefinidos para los tipos básicos y permite la creación de

editores propios, gracias a lenguajes JSP, *JavaScript*, HTML, *Ajax*, o la tecnología de presentación web que queramos, asignándolos al atributo que deseemos.

Para poder describir estos editores, *Openxava* utiliza los estereotipos [82].

4.9.1.5 Validadores.

Corresponde a la lógica de validación que puede usarse en cualquier componente de negocio.

4.9.1.6 Calculadores.

Lógica de negocio reutilizable que se puede usar en algunos puntos de los componentes de negocio.

Un calculador muy recurrido para este proyecto fue el calculador de la fecha actual, por ejemplo, para poner como valor por defecto en las pruebas médicas, o en la instalación del equipo generador cardiológico, el día en curso.

4.9.2 Ventajas de usar el marco de trabajo *Openxava*.

- **Alta productividad [83].** Solamente tuvimos que programar la lógica de negocio y la estructura de los datos. No escribimos HTML, JavaScript, CSS, SQL, etc para la gran mayoría de la funcionalidad de la aplicación web de cardiología. La interfaz de usuario y la lógica de base de datos se proveen automáticamente.
- **Aplicaciones con mucha funcionalidad.** Interfaz de usuario *Ajax* sin recarga de página. Modo lista con paginación, ordenación, filtrado, añadir/quitar/mover columnas, informes PDF, exportación a Excel, formato tarjetas, gráficos, etc. Modo detalle con pestañas, marcos, diálogos, editores para referencias y colecciones, disposición adaptable, etc.
- **Portabilidad.**
 - ➔ **Navegadores:** Internet Explorer, Chrome, Firefox y Safari.
 - ➔ **Bases de datos:** Cualquiera soportada por *Hibernate*, es decir, *Oracle*, DB2, AS/400, Informix, PostgreSQL, MySQL, MS SQL Server y practicamente todas las bases de datos relacionales.
 - ➔ **Sistemas operativos:** Cualquier con soporte de Java 6 (o superior), es decir, Windows, Linux, Mac, Unix, AS/400, z/OS, etc.
 - ➔ **Servidores de aplicaciones:** Cualquiera con soporte de Servlets 2.5 (o

superior), incluyendo Tomcat, JBoss, WebSphere, Glassfish, WebLogic, etc.

→ **Portales empresariales:** Cualquiera con soporte de JSR-168 o JSR-286 incluyendo WebSphere Portal y Liferay.

- **Código abierto.**
- **Proyecto activo.** 6 versiones al año, con resolución de fallos y nuevas funcionalidades.
- **Basado en estándares Java.**
 - JSR-317. Java Persistence 2.0 [84].
 - JSR-303. Bean Validation [85].
 - JSR-330. Dependency Injection for Java [86].
 - JSR-220. Enterprise JavaBean 3.0 [87].
 - JSR-153. Enterprise JavaBean 2.1 [88].
 - JSR-168. Portlet Specification [89].
 - JSR-286. Portlet Specification 2.0 [90].

En este proyecto, el alumno tuvo una curva de aprendizaje corta para el lenguaje de programación *Java*, necesario para describir las clases Java que definen la aplicación, las relaciones entre ellas, etc. Es por ello que en el siguiente capítulo se mencionan las anotaciones que se pueden usar con *OpenXava* como por el ejemplo el estándar *Java* JSR-317, ya que formó parte del trabajo para poder desarrollar la aplicación web.

4.10 Java: clases, objetos y herencias.

Como ya mencionamos anteriormente, en este proyecto usamos la tecnología *Hibernate* para solucionar el problema de desajuste de impedancia que se produce porque las bases de datos relacionales tienen una estructura, tablas y columnas con datos simples, y las aplicaciones orientadas a objetos otra, clases con referencias, colecciones, interfaces, herencia, etc. En lugar de escribir código SQL para la persistencia de los datos, usamos *Java Persistence API (JPA)* [91] para ello.

A la hora de describir las clases java del modelo de datos, usamos varios tipos de recursos; anotaciones JPA para indicar que las clases son persistentes y para dar información sobre el mapeo entre las clases y las tablas de la BB.DD, y luego una API para leer y escribir objetos desde la aplicación de cardiología.

En el siguiente bloque, se explica cómo se define una clase persistente, cómo se

representan sus propiedades y cómo se relacionan con las tablas de la BB.DD., cómo relacionamos las tablas, etc gracias a las anotaciones JPA y *Openxava*.

En el siguiente apartado hablaremos sobre Ajax, tecnología usada para hacer nuestra aplicación de cardiología interactiva; el navegador web del usuario mantiene una comunicación asíncrona con el servidor en segundo plano para no tener que recargar la página.

4.11 Ajax: Asynchronous JavaScript And XML.

La tecnología *Ajax* es un acrónimo de Asíncrono, JavaScript y XML, la cual se encarga de unir varias tecnologías con el fin de mejorar la experiencia del usuario, y hacer la aplicación de cardiología más interactiva.

Sin *Ajax*, con la aplicación web tendríamos que esperar a refrescar la página completamente para poder recibir la respuesta de la solicitud aunque sea un sólo elemento el que se modifique, y no podríamos seguir interactuando con la aplicación. Con *Ajax* cambiamos totalmente esta situación.

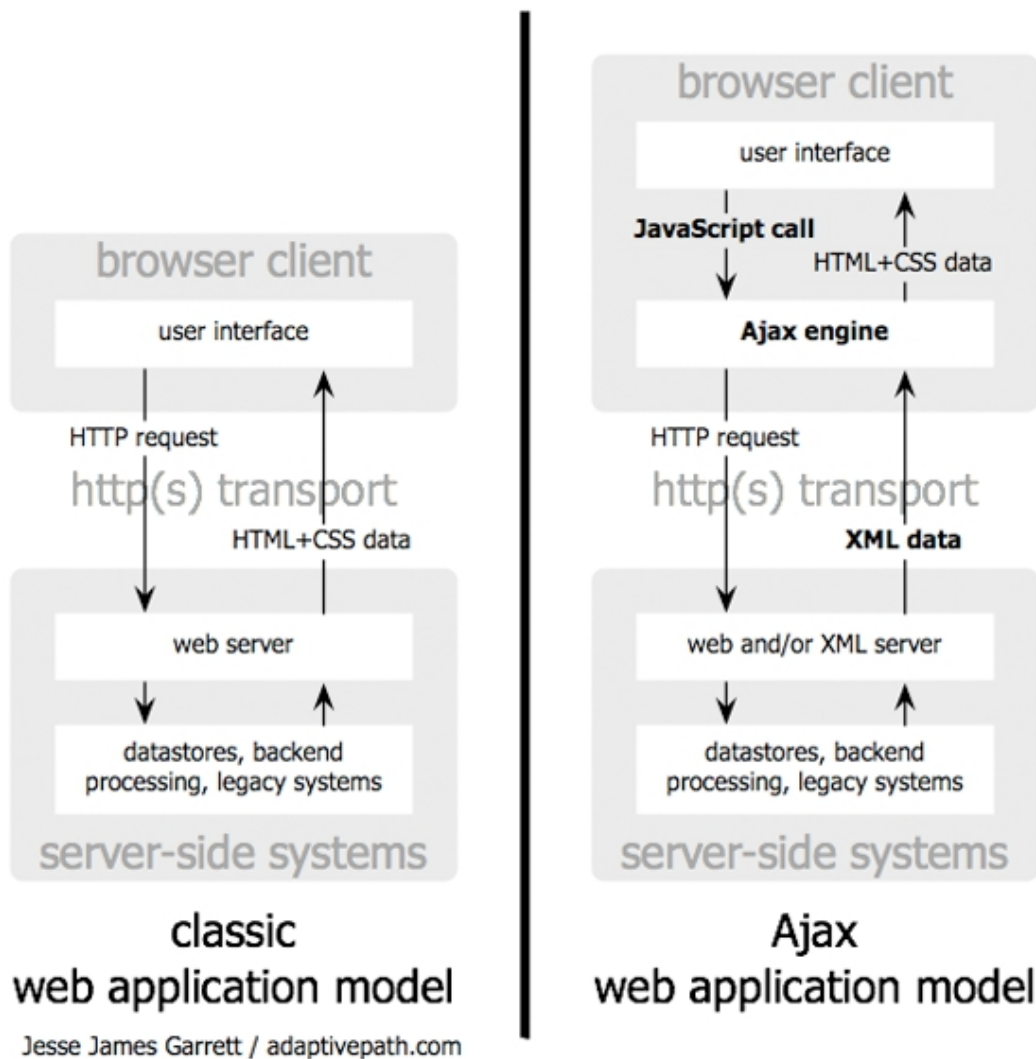


Figura 52: Comparativa de una aplicación web clásica y una aplicación web con tecnología Ajax [92].

Gracias a la API *XMLHttpRequest* (XHR) [93] se envían peticiones HTTP al servidor asíncronas que ya no tienen que esperar a que recargue la página completa, refrescando sólo los elementos que cambian.

4.11.1 Funcionamiento del ciclo Ajax.

Al incluir Ajax como tecnología, la interfaz gráfica de usuario además de contar con el navegador web ahora cuenta con un motor Javascript.

Cuando el usuario hace una llamada a la acción a través de un botón o enlace, le lleva a un evento de Javascript local, es decir, ubicado en la máquina del usuario de cardiología.

Cuando tiene lugar este evento de javascript que se entrega a esta XHR, lo que sucede es que se envía la petición y los datos al servidor.

La API HXR devuelve entonces los datos de respuesta al usuario en algún tipo de formato; texto, JSON o XML, y mientras que estos datos llegan y se actualizan gracias a funciones JavaScript que actualizan el árbol de componentes DOM [94], el usuario puede seguir utilizando la página web donde esté trabajando.

A continuación mostramos lo descrito de manera gráfica, en la línea del tiempo:

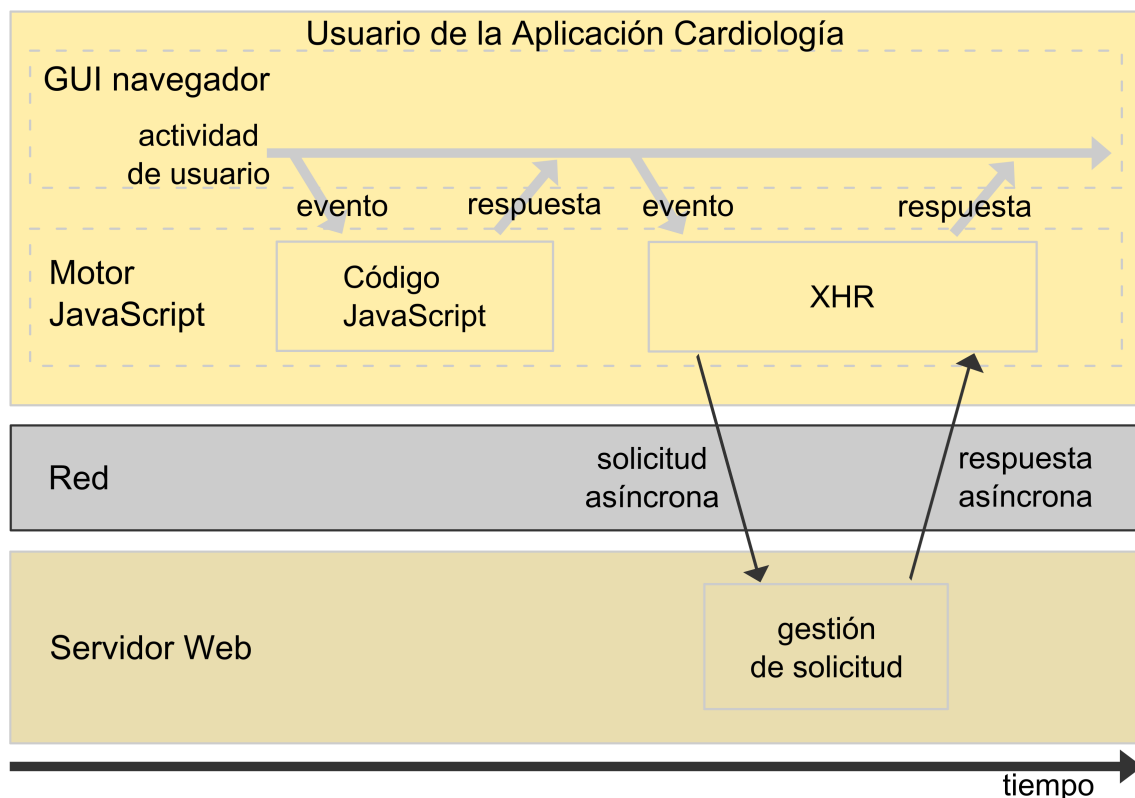


Figura 53: Representación del ciclo Ajax en la aplicación de cardiología.

4.11.2 Ventajas.

- **Aplicación interactiva.** La actividad de los usuarios no se interrumpe nunca.
- **Mejor usabilidad.** No se pierde tanto tiempo innecesario esperando a recargar la página, y por lo tanto la experiencia del usuario es mejor.

Una vez comprendido como la aplicación web de cardiología solicita y recibe datos al/del servidor, hablaremos sobre los lenguajes de programación usados en el nivel de presentación para visualizar los datos; HTML y CSS.

4.12 Nivel de presentación de los datos.

El nivel de presentación constituye el nivel de cliente y la lógica de presentación para que los usuarios de cardiología puedan ver los datos e interactúen con la aplicación a través del navegador.

En este capítulo explicamos cómo nuestra aplicación web especifica la estructura de las páginas web a través del lenguaje HTML, cómo representa los datos a través de las hojas de estilo del lenguaje CSS o le añade comportamiento a las páginas web de la aplicación de cardiología con *JavaScript*.

4.12.1 HTML5.

Con este lenguaje, *Openxava* define las páginas web, formularios web etc en los que se representan los datos a enviar y recibir.

En los *scripts* donde se describen estos formularios encontraremos código HTML que define cada elemento HTML, que consta de:

- Una etiqueta de inicio que contiene el nombre del elemento y demás atributos.
- Una etiqueta final cuyo nombre coincide con el de la etiqueta de inicio.
- El contenido el cual se ubica entre las etiquetas de inicio y final.

Si observamos el ejemplo de código fuente siguiente, vemos su significado comentado en color verde:

```

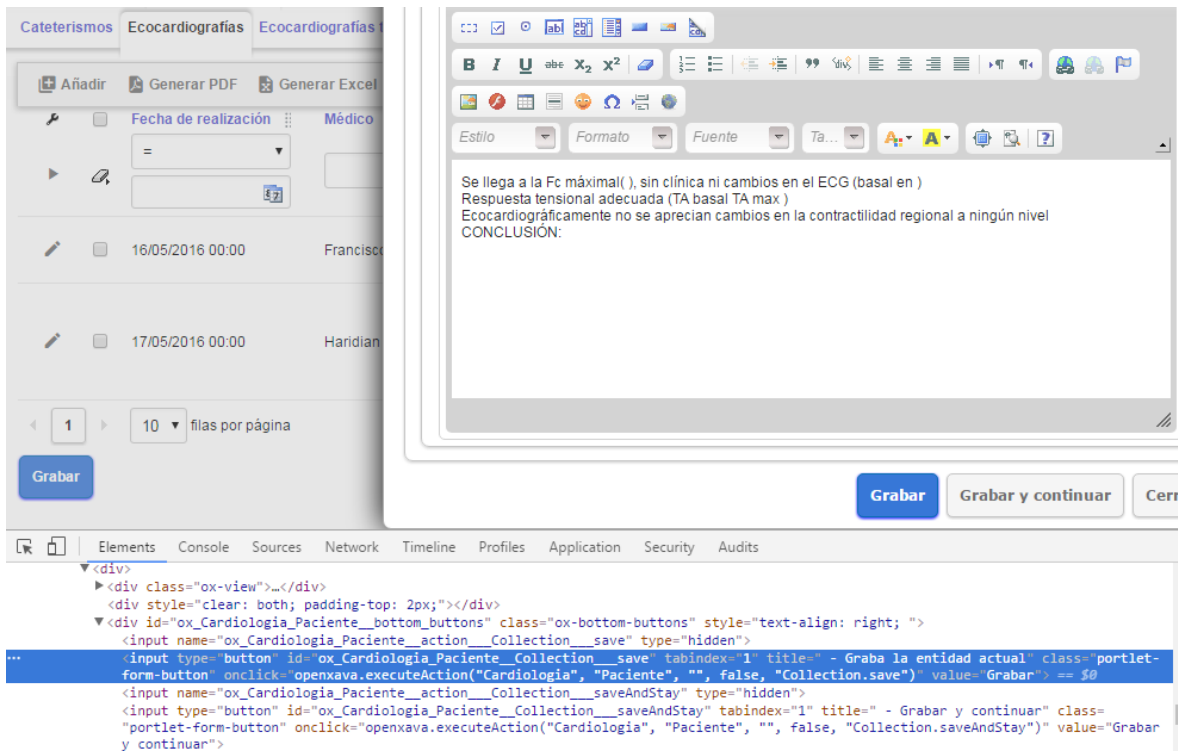
1
2
3 <a href="/Cardiologia/m/AlergiaNueva"> <!-- enlaza el div con id="AlergiaNueva_module" de la línea 2 con el módulo AlergiaNueva-->
4 <div id="AlergiaNueva_module" class="module-row " onclick="$('#AlergiaNueva_loading').show()"><!-- etiqueta de inicio del div con
5 id="AlergiaNueva_module", se definen los estilos en el atributo class, y con el atributo onclick, se llama a la función JavaScript que
6 muestra el div con id=AlergiaNueva_loading que es el que permite registrar una nueva alergia-->
7 <div class="module-name"><!-- etiqueta de inicio del div que muestra el nombre del módulo-->
8 Alergia nueva
9
10 
11 </div><!-- etiqueta de fin del div con class="module-name"-->
12 <div class="module-description">Alergia nueva</div><!-- etiqueta de inicio y fin del div. Entre ellos está el contenido de la
13 descripción del módulo-->
14 </div><!-- etiqueta de fin del div con id="AlergiaNueva_module"-->
15 </a><!-- Etiqueta de fin del enlace al módulo AlergiaNueva -->
16
17
18
19
20
21
22
23

```

Captura 65: Fragmento de código del código fuente de la aplicación web.

Captura 66: div que se muestra gracias a la línea 3 del código anterior.

Inspeccionando con la herramienta del navegador web el formulario Añadir ecocardiografía, el cual ejemplifica cómo *Openxava* envía los datos mediante *Ajax*:



Captura 67: Captura de pantalla de la página web para *ecocardiograma* de la aplicación web.

Al inspeccionar el botón “Grabar”, vemos que gracias al atributo *onclick*, se llama a la función *openxava.executeAction* a la cual se le pasan los parámetros correspondientes para grabar el registro en la tabla *ecocardiografia*.

Cabe destacar, que *Openxava* permite tanto modificar el código HTML del marco de trabajo como crear contenido nuevo, sólo debemos tener extremo cuidado de no modificar, eliminar, código que comprometa el funcionamiento de *Openxava*.

4.12.2 CSS3.

Este lenguaje se usa para especificar la semántica de presentación o estilo del documento HTML, y para ello se crean las reglas que especifican cómo debe aparecer cada elemento.

Por ejemplo, si analizamos de nuevo el código de *cabecera.jsp*, nos fijamos que en la línea 14, al título de la cabecera se le aplica la clase *Estilo1* definida desde la línea 3 hasta la 7. Esto quiere decir, que este título tendrá un color determinado (línea 4), aparece como texto resaltado (línea 5), y que el tamaño de la fuente es de 25 px (línea 6).

```
enlaces bt | prueba.jsp | prueba.java | cabecera.jsp | paciente.java | generador.java | visita.java | guardaAlergia.java | diagnostico.java |
1 <style type="text/css">
2
3 .Estilol {
4     color: #000066;
5     font-weight: bold;
6     font-size: 25px;
7 }
8 </style>
9
10 <table width="100%" height="100" border="0" align="center"><!--etiqueta de inicio de la tabla con una fila y tres columnas -->
11 <tr><!--etiqueta de inicio de la fila de la tabla-->
12 <td width="15%" height="100"><div align="left"></div></td><!--etiqueta de inicio, contenido y etiqueta de fin de la 1ª columna de la tabla-->
14 <td width="85%"><div align="center"><span class="Estilol"> CARDIO WEB-ESTIM <BR />Gestión de Pruebas Especiales e Implantes
15 Cardiológicos </span></div></td><!--etiqueta de inicio, contenido y etiqueta de fin de la 2ª columna de la tabla-->
16 <td width="25%"><div align="right"></div></td>
17 </tr><!--etiqueta de fin de la fila de la tabla-->
</table><!--etiqueta de fin de la tabla con una fila y tres columnas -->
```

Captura 68: Captura de pantalla de fragmento de código de *cabecera.jsp*.

El objetivo de este capítulo no trata de nombrar cada uno de las etiquetas HTML que se usan ni para qué se aplican, ni la nomenclatura para escribir clases CSS, pero sí persigue que se comprenda el uso de estas tecnologías empleadas, y cómo modificando código de ambos lenguajes, podemos hacer que el marco de trabajo se adapte a las necesidades y requerimientos del proyecto fin de carrera.

4.13 Uso de *Ireport* y librerías JasperReport para generar informes pdf.

Uno de los requisitos para esta aplicación web, es que se puedan imprimir los informes médicos, informes de las pruebas médicas, etc de manera automática.

La herramienta escogida para ello es *Ireport* [95], una interfaz para crear reportes en diversos formatos e imprimirlos de una forma simple y flexible. Esta interfaz utiliza *JasperReports* [96] para crear reportes para java, es decir, crea un archivo con extensión *.jrxml* [97] el cual contiene la estructura del diseño del informe seleccionado en formato XML , e *iText* [98], una serie de librerías para exportar a formato PDF.

El ejemplo que analizaremos en el siguiente bloque es la generación automática de informes para las pruebas médicas de cateterismo.

4.14 El estudio de la experiencia de usuario como valor añadido.

Aplicar principios de usabilidad en los productos tecnológicos es cada vez más usual. La usabilidad es la disciplina que estudia la forma de diseñar productos del entorno web para que los usuarios puedan interactuar con ellos de la forma más fácil, cómoda e intuitiva posible. Ésto lo conseguimos realizando un diseño centrado en el usuario, y no centrado en la tecnología o las tendencias, por ejemplo.

En este proyecto fin de carrera se ha intentado mejorar la experiencia del usuario con la aplicación de cardiología, aplicando tecnologías como *Ajax*, o funcionalidades de la aplicación con el fin de lograr este objetivo.

Las funcionalidades que se han desarrollado para ello son, aplicar etiquetas para la internacionalización de la aplicación, describir los estereotipos para indicar la unidad de medida de los valores a insertar que lo requieran, cargar plantillas en informes dependiendo de los valores tomados por algunos campos del formulario, o no salir del formulario del objeto al guardar, para que el usuario pueda salvar la información y seguir trabajando con el objeto.

Cómo se consiguió, y el código de la aplicación dedicado a la usabilidad web de la aplicación se explicará en el siguiente apartado.

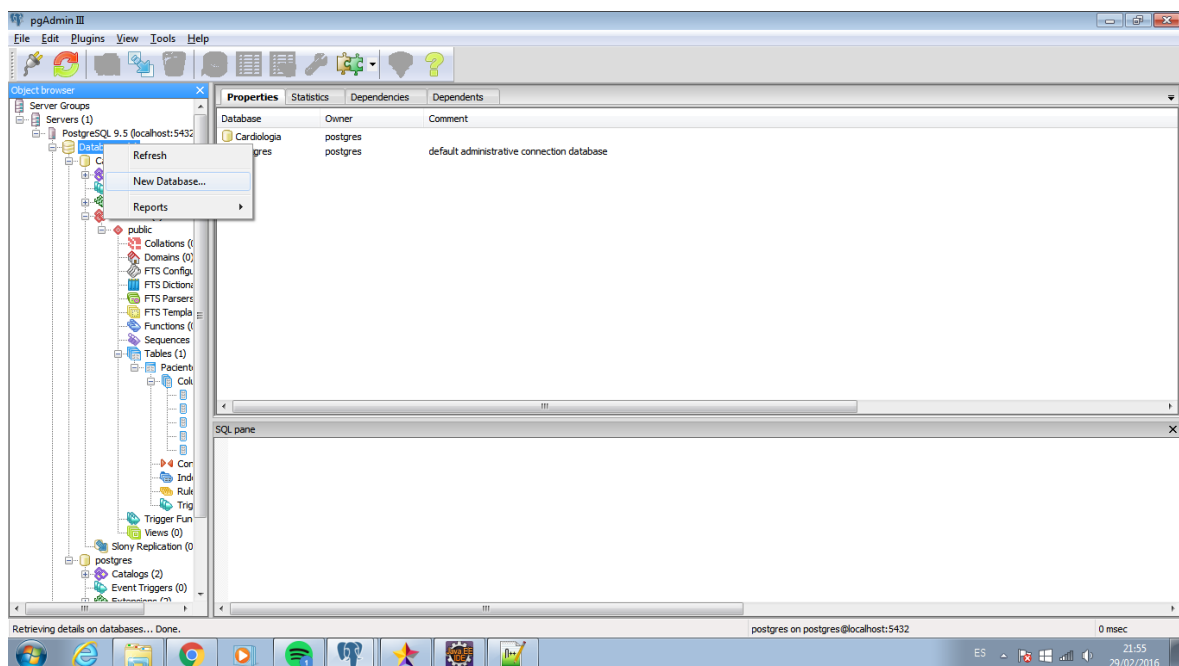
5. Aplicación Cardiología.

5. Aplicación Cardiología.

5.1 Configuración de las tecnologías empleadas.

5.1.1 Gestión de la Base de Datos con PostgreSQL.

Para poder comenzar a desarrollar la aplicación, uno de los primeros pasos a seguir es conectar el servidor (introduciendo los datos de autenticación) para acceder a los datos, crear la base de datos “cardiología”, que contendrá todas las tablas de las clases de nuestra aplicación, y conectar el marco de trabajo *Openxava* con PostgreSQL a través de la fuente de datos; el conector *JDBC*.



Captura 69: Creación de la nueva base de datos *cardiologia*.

Un detalle a tener en cuenta es que todas las tablas y parámetros que creemos en la BB.DD irán en minúsculas, porque aunque veamos más adelante en Eclipse que cada clase puede tener alguna letra en mayúscula (parámetro *numeroNhc* de la clase *Paciente*), *Hibernate*, herramienta usada para el mapeo objeto-relacional que se explicará más adelante, lo traduce todo a minúsculas (*numeronhc*):

numeroNhc (paciente.java) -> numeronhc (parámetro de la BB.DD en postgresSQL)

A medida que en el desarrollo necesitemos tablas con sus parámetros, las introducimos a través de pgAdmin III, teniendo en cuenta la indicación anterior, o bien manualmente, o a través de herramientas adicionales automáticas de Eclipse, la herramienta *IDE* escogida.

5.1.2 Configuración de *Hibernate*.

Para configurar nuestro marco *Openxava* e integrar en él *Hibernate*, tuvimos que modificar desde la herramienta IDE utilizada; *Eclipse*, los ficheros XML de configuración de *Hibernate* como *hibernate.cfg.xml* dentro del proyecto “cardiología” para conectar correctamente con la BB.DD requerida.

```
5 <!--
6 This hibernate configuration is used only for IMAGES_GALLERY stereotype,
7 because this stereotype is implemented using hibernate native APIs.
8 If you do not use IMAGE_GALLERY you do not need this file.
9
10 The datasource configured here is the datasource in where the images will be
11 store, and usually match with the main datasource of application defined in
12 persistence.xml.
13 -->
14
15 <hibernate-configuration>
16
17   <session-factory>
18
19     <property name="hibernate.connection.datasource">java:comp/env/jdbc/CardiologiaDS</property>
20     <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
21     <!-- Tomcat + Hypersonic -->
22     <property name="hibernate.connection.datasource">java://comp/env/jdbc/CardiologiaDS</property>
23     <property name="hibernate.dialect">org.hibernate.dialect.HSQLDialect</property>
24     <property name="hibernate.jdbc.use_get_generated_keys">>false</property>
25     <property name="hibernate.show_sql">>false</property>
26     <property name="hibernate.connection.release_mode">after_transaction</property>
27
28     <!-- WebSphere + AS/400
29     <property name="hibernate.connection.datasource">jdbc/CardiologiaDS</property>
30     <property name="hibernate.dialect">org.hibernate.dialect.DB2400Dialect</property>
31     <property name="hibernate.jdbc.use_get_generated_keys">>false</property>
32     <property name="hibernate.show_sql">>false</property>
33     <property name="hibernate.connection.release_mode">after_transaction</property>
34     -->
35
36     <!-- GalleryImage is needed only if you uses IMAGES_GALLERY/GALERIA_IMAGENES stereotype -->
37     <mapping resource="GalleryImage.hbm.xml"/>
38
39   </session-factory>
40
41 </hibernate-configuration>
```

Captura 70: Captura de pantalla del contenido del fichero de configuración predefinido *hibernate.cfg.xml*.

En las líneas resaltadas de la captura de pantalla anterior, vemos la configuración del fichero *hibernate.cfg.xml* para conectar con la fuente de datos de la BB.DD de *cardiología* y especificar el dialecto de ésta; *PostgreSQLDialect*.

También configuramos el fichero de configuración *persistence.xml*:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3
4 <persistence xmlns="http://java.sun.com/xml/ns/persistence"
5   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6   xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
7   version="1.0">
8
9   <!-- Tomcat + PostgreSQL -->
10  <persistence-unit name="default">
11    <provider>org.hibernate.ejb.HibernatePersistence</provider>
12    <non-jta-data-source>java:comp/env/jdbc/CardiologiaDS</non-jta-data-source>
13    <class>org.openxava.session.GalleryImage</class>
14    <properties>
15      <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>
16    </properties>
17  </persistence-unit>
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71 <persistence-unit name="junit">
72   <properties>
73     <!-- Clase del controlador de PostgreSQL -->
74     <property name="hibernate.connection.driver_class"
75       value="org.postgresql.Driver"/>
76     <!-- Dialecto de PostgreSQL -->
77     <property name="hibernate.dialect"
78       value="org.hibernate.dialect.PostgreSQLDialect"/>
79
80     <property name="hibernate.connection.url"
81       value="jdbc:postgresql://localhost:5432/cardiologia"/>
82
83     <!-- El usuario de la base de datos -->
84     <property name="hibernate.connection.username" value="openxava"/>
85
86     <!-- La contraseña del usuario de la base de datos -->
87     <property name="hibernate.connection.password" value="openxava"/>
88   </properties>
89 </persistence-unit>
90
91 </persistence>

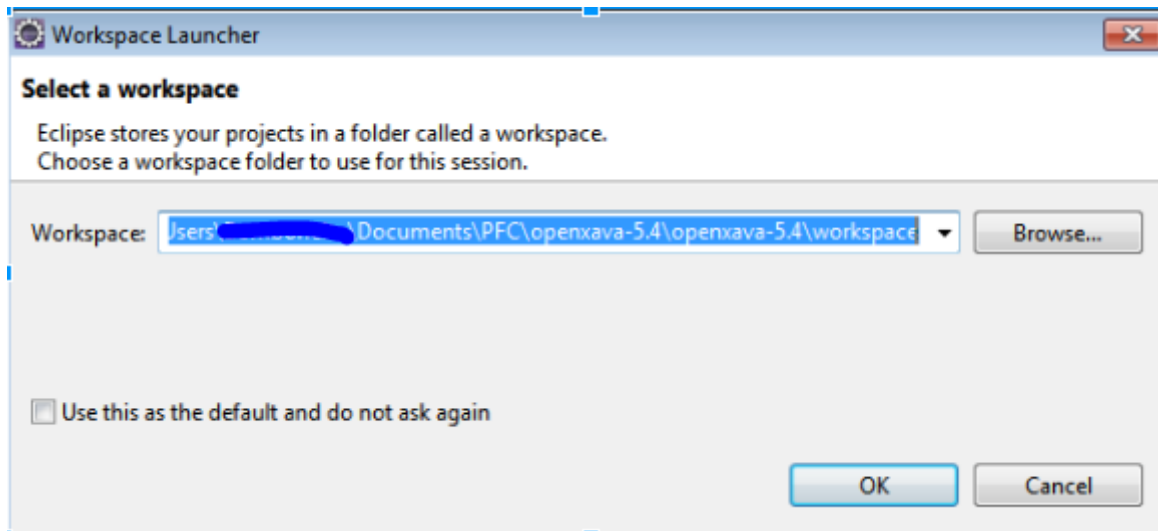
```

Captura 70: Capturas de pantalla del contenido del fichero de configuración predefinido *persistence.xml*.

Este fichero contiene, entre otros, datos de conexión con la BB.DD. como en la línea 84 (usuario) o en la línea 87 (contraseña), o añade el driver de PostgreSQL para poder usarlo.

5.1.3 Configuración de la herramienta IDE.

Al ejecutar Eclipse, antes de llegar al *Workbench*, nos pide que definamos el *workspace* donde trabajaremos. Éste, lo hemos descargado previamente de la página oficial de *Openxava* ya que la distribución de *OpenXava* incluye un servidor *Tomcat* y un *workspace* de *Eclipse*, todo configurado y listo para usar.



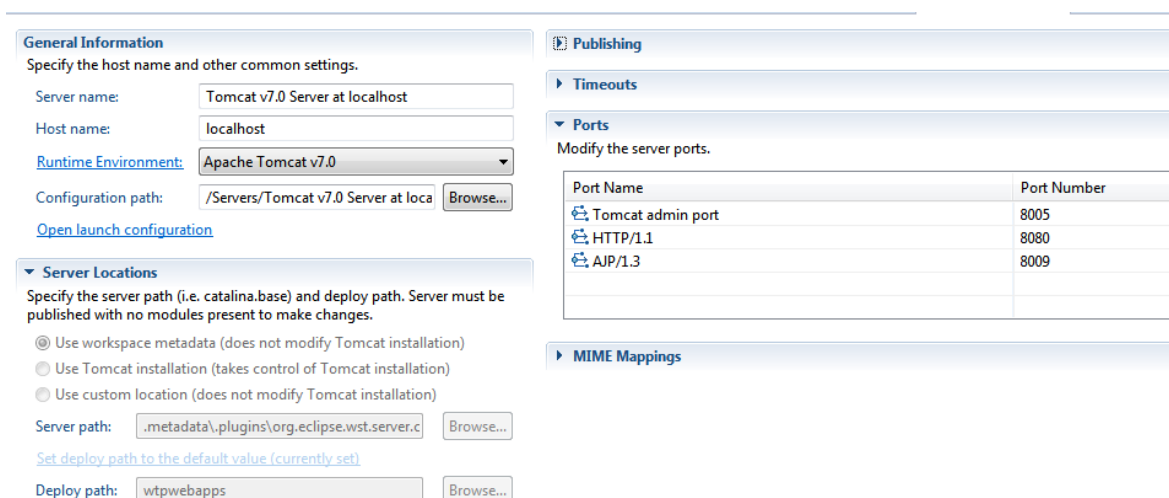
Captura 71: Ventana de inicio de Eclipse para seleccionar el *workspace* deseado.

Una vez seleccionado el *workspace*, ya podemos proceder con las siguientes configuraciones.

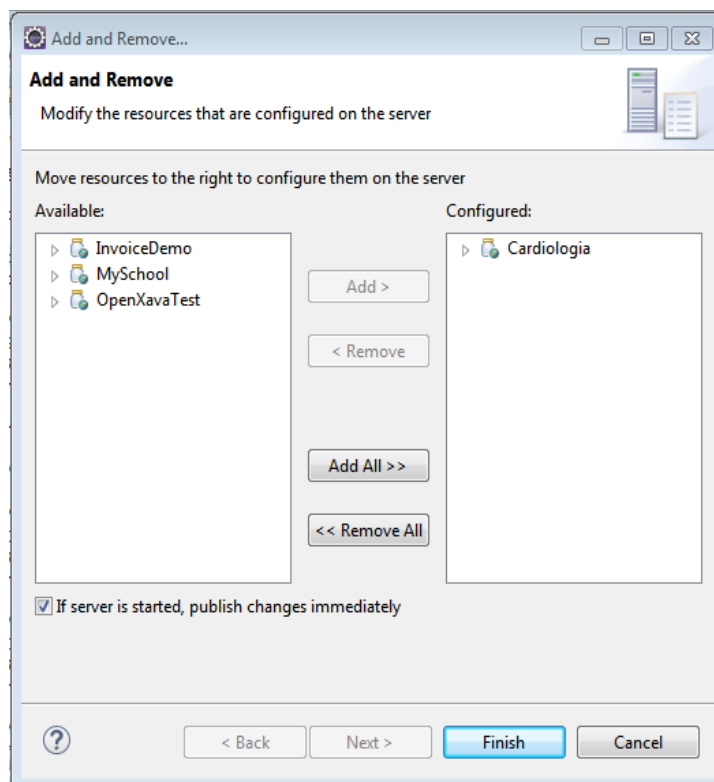
5.1.4 Configuración del *Tomcat*, el servidor para aplicaciones.

Una vez preparada la herramienta IDE para su uso, se configuró el servidor *Tomcat* y el conector para la fuente de datos para conectar la aplicación en desarrollo y la BB.DD. de pruebas de cardiología.

Primero se añadió el servidor de ejecución *Tomcat v.7.0* para ejecutar nuestra aplicación web, y a continuación, la aplicación *Cardiología* a éste. Esto lo hicimos a través del IDE *Eclipse*.

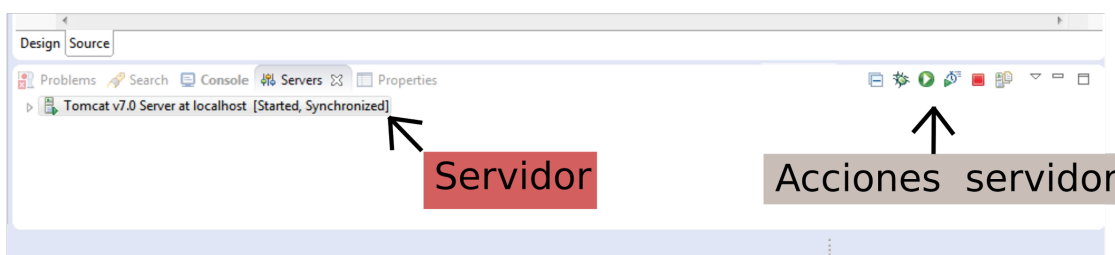


Captura 72: Características del servidor *Tomcat* añadido en *Eclipse* como servidor de ejecución.



Captura 73: Captura de pantalla de las aplicaciones incluidas en el servidor.

En este proyecto, gracias a *Eclipse*, no tendremos que arrancar, parar etc el servidor a través de comandos como los ejemplos presentados en la teoría dedicada al *Tomcat* en el Bloque 4, lo hacemos a través de la interfaz que facilita Eclipse para ello:



Captura 74: Captura de pantalla de la sección servidor del IDE Eclipse.

Cabe señalar que, existieron casos en los que a la hora de conectar el servidor los puertos estaban bloqueados. Cuando esto sucedía hicimos uso de la aplicación llamada *Cports* [99], para “matar” los procesos que bloquean el puerto 8080, ya que la url para acceder a la aplicación en desarrollo es:

<http://localhost:8080/Cardiologia/modules/Paciente>

Process Na...	Proces...	Protocol	Local Port	Local Por...	Local Address	Remote ...	Remote ...
chrome.exe	5188	TCP	50519		127.0.0.1	12344	
chrome.exe	5188	TCP	51691		127.0.0.1	12344	
chrome.exe	5188	TCP	61160		127.0.0.1	12344	
chrome.exe	5188	UDP	5353		0.0.0.0		
chrome.exe	5188	UDP	53635		0.0.0.0		
chrome.exe	5188	UDP	55499		0.0.0.0		
chrome.exe	5188	UDP	57588		0.0.0.0		
chrome.exe	5188	UDP	61067		0.0.0.0		
chrome.exe	5188	UDP	62493		0.0.0.0		
chrome.exe	5188	UDP	5353		::		
javaw.exe	8496	TCP	8005		127.0.0.1		
javaw.exe	8496	TCP	62506		127.0.0.1	5432	
javaw.exe	8496	TCP	8009		0.0.0.0		
javaw.exe	8496	TCP	8080		0.0.0.0		
javaw.exe	8496	TCP	8009		::		
javaw.exe	8496	TCP	8080		::		
LavasoftTcpSe...	1588	TCP	12344		127.0.0.1		

Captura 75: Captura de pantalla de la aplicación que detecta los procesos para “matarlos”.

5.1.5 Configuración del servidor Tomcat para incluir el JDBC para postgresQL.

Al considerar el JDBC una tecnología de servidor, configuramos la fuente de datos contra la BB. DD. de cardiología en el servidor Tomcat desde el fichero de configuración *context.xml*:

```

26
27 <!-- Uncomment this to enable Comet connection tacking (provides events
28      on session expiration as well as webapp lifecycle) -->
29 <!--
30 <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
31 -->
32
33 <!--
34 <Resource name="jdbc/OpenXavaTestDS" auth="Container" type="javax.sql.DataSource"
35     maxActive="20" maxIdle="5" maxWait="10000"
36     username="sa" password="" driverClassName="org.hsqldb.jdbcDriver"
37     url="jdbc:hsqldb:file:../data/openxava-test-db"/>
38 -->
39 <Resource auth="Container" name="jdbc/CardiologiaDS"
40     driverClassName="org.postgresql.Driver"
41     maxActive="20" maxIdle="5" maxWait="10000"
42     username="sa" password="" type="javax.sql.DataSource"
43     url="jdbc:postgresql://127.0.0.1:5432/cardioLogia" />
44
45 <Resource auth="Container" driverClassName="org.hsqldb.jdbcDriver" maxActive="20" maxIdle="5" maxWait="10000" name="jdbc/OpenS
46
47 <Resource auth="Container" driverClassName="org.hsqldb.jdbcDriver" maxActive="20" maxIdle="5" maxWait="10000" name="jdbc/MyScl
48
49 <Resource auth="Container" driverClassName="org.hsqldb.jdbcDriver" maxActive="20" maxIdle="5" maxWait="10000" name="jdbc/Invoi
50
51 </Context>

```

Captura 76: Fragmento de código del fichero de configuración del servidor *context.xml*.

Como vemos, desde la línea 39 a la 43 se añade la fuente de datos **CardiologiaDS**; se especifica el driver de PostgreSQL, el usuario y contraseña para acceder a la BB.DD. y la url.

Como último paso para este apartado, añadimos al servidor el controlador de PostgreSQL *postgresql-8.4-703.jdbc3.jar* [100] (descargado desde la página oficial) en la ruta *Cardiologia/web/WEB-INF/lib*, y en la carpeta de las librerías del Tomcat (en *...\\openxava-5.4\\openxava-5.4\\tomcat\\lib*). Este archivo juega un papel fundamental para que funcione la fuente de datos, ya que sin él, el servidor no puede conectarse a la BB.DD.

5.1.6 Configuración para acceder al sistema de gestión de la BB.DD.

La aplicación Web de cardiología incluye información general de configuración JSP en el fichero *web.xml*, ubicado en la ruta *...\\openxava-5.4\\workspace\\Cardiologia\\web\\WEB-INF\\index.jspweb.xml* que es utilizado por el contenedor JSP.

Este fichero no fue modificado ya que viene predefinido al descargar el *workspace* de *Openxava* para trabajar con él en la herramienta *Eclipse*. Veamos un fragmento de código para ver su configuración:

```
5   If you want to add your own servlets add them in servlets.xml file.
6   servlets.xml must be in WEB-INF and can have several <servlet> and <servlet-mapping>.
7   If you want to add filters add them in filters.xml file in WEB-INF.
8   filters.xml must be in WEB-INF and can have several <filter> and <filter-mapping>.
9   --->
10
11
12  <web-app version="2.4"
13  xmlns="http://java.sun.com/xml/ns/j2ee"
14  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
15  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
16
17
18  <display-name>OpenXava Application</display-name>
19
20  .....
21
22  <!-- Addons filters begin -->
23  <filter>
24  <filter-name>naviox</filter-name>
25  <filter-class>com.openxava.naviox.web.NaviOXFilter</filter-class>
26  </filter>
27  <filter-mapping>
28  <filter-name>naviox</filter-name>
29  <url-pattern>*.jsp</url-pattern>
30  </filter-mapping>
31  <filter-mapping>
32  <filter-name>naviox</filter-name>
33  <url-pattern>/modules/*</url-pattern>
34  <dispatcher>REQUEST</dispatcher>
35  <dispatcher>FORWARD</dispatcher>
36  </filter-mapping>
37  <filter-mapping>
38  <filter-name>naviox</filter-name>
39  <servlet-name>naviox</servlet-name>
40  </filter-mapping>
41  <filter-mapping>
42  <filter-name>naviox</filter-name>
```

Captura 77: Captura de pantalla de fragmento de código del archivo *web.xml*.

En este código vemos cómo se añaden algunos filtros vistos en el apartado de arquitectura JSP, para entre otras cosas, controlar el acceso a la aplicación web por seguridad e integridad de los datos.

Por ejemplo, el código:

```
<filter-mapping>
    <filter-name>naviox</filter-name>
    <servlet-name>phone</servlet-name>
</filter-mapping>
```

asocia el filtro con un *servlet* específico usando la etiqueta **<filter name>**. En este caso, se ejecuta el filtro *naviox* al invocar el *servlet* **phone**.

5.1.7 Configuración para el marco de trabajo *Openxava*.

A continuación se expone un ejemplo de configuración de los diferentes elementos que nos encontraremos en la aplicación del HUGCDN.

5.1.7.1 Componentes de negocio.

Para poder desarrollar la aplicación, todas estas clases se ubican en la ruta del *workspace* de *Openxava* que definen el modelo de los datos ...*\openxava-5.4\openxava-5.4\workspace\Cardiologia\src\org\openxava\cardiologia\model\ClaseJava.java*.

5.1.7.2 Módulos.

Para nuestra aplicación, a la hora de ponerla en producción se debe modificar el código para quitar la navegación entre los módulos que se generan automáticamente con *Openxava*. Ésto es así ya que si se accediera por ejemplo al módulo generador directamente sin tener datos del paciente como el número de historia clínica (que se recogen desde las vistas de ficha de paciente), no tendría sentido grabar un nuevo dispositivo puesto que éste siempre está vinculado a una persona, y se guardaría sin tener la clave foránea que lo relaciona.

Esta configuración la conseguimos modificando entre otros, el *script* que define el menú principal *mainNavigator.jsp* y comentando el código que tiene por defecto para mostrar

los enlaces a los módulos:

```
7 <%%page import="com.openxava.naviox.Modules"%>
8 <%%page import="com.openxava.naviox.util.NaviOXPreferences"%>
9
10 <jsp:useBean id="modules" class="com.openxava.naviox.Modules" scope="session"/>
11
12 <% if (modules.hasModules()) { %>
13 <!-- <a id="show_modules" href=""><xava:message key="all_modules"/></a -->
14 <% } %>
15
16 <%
17 String allModulesClass = modules.hasModules()? "main-navigation-left-with-all-modules": "main-navigation-left-without-all-modules";
18 %>
19
20 &nbsp;
21 <div id="main_navigation_left" class="<%=allModulesClass%>">
22 <noBr>
23 &nbsp;
24 <%
25
26
27
28 for (Iterator it= modules.getTopModules().iterator(); it.hasNext(); ) {
29 MetaModule module = (MetaModule) it.next();
30 if (module.getName().equals("SignIn")) continue;
31
32 String selected = module.getName().equals(request.getParameter("module"))?"selected":"";
33 %>
34 <!-- <a href="<%=modules.getModuleURI(request, module)%>?retainOrder=true" class="<%=selected%>">
35 <%=module.getLabel(request.getLocale())%>
36 -->
37 <%
38 }
39 %>
```

Captura 78: Captura de pantalla de fragmento de código del fichero *mainNavigator.jsp*

Tras modificar código, se observa que en el menú principal de arriba se ha eliminado esta navegación entre módulos, y además se añadió una cabecera con los logos corporativos del HUGCDN.



Captura de pantalla de cambio de cabecera y ocultación de navegación entre módulos.

5.1.7.3 Controladores.

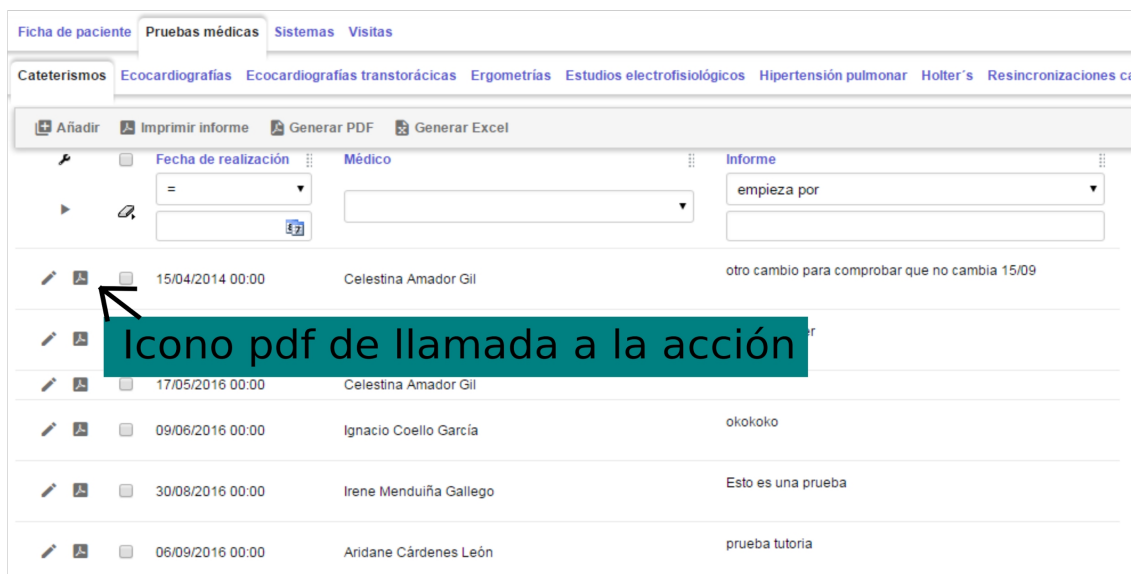
Al añadir nuevas acciones para la aplicación web de cardiología como lo piden los requisitos, podemos hacerlo sobrescribiendo el fichero de configuración *controladores.xml*, donde se encuentra el controlador por defecto del módulo (obligatorio para que tenga las operaciones básicas de manipulación de datos que podemos realizar con SQL llamadas CRUD; crear, leer, actualizar y borrar el elemento), y el controlador de la nueva acción.

Por ejemplo, veamos cómo para poder imprimir el informe de una prueba médica persistente de Cateterismo, añadimos el siguiente código correspondiente a esta nueva acción en el script *controladores.xml*:

```
><!-- controlador con acción nueva para Cateterismo, para imprimir el informe de la prueba.
Esta acción llama a la clase java que contiene la acción de impresión: clase="org.openxava.cardiologia.actions.InformePr
> <controlador nombre="controlpdf">
>   <accion nombre="ImprimirInforme" icono="file-pdf-box" modo="detail/List"
>     clase="org.openxava.cardiologia.actions.InformePrueba"
>     en-cada-fila="true" >
>   </accion>
> </controlador>
```

Captura 79: Captura de pantalla de un fragmento de código del script *controladores.xml*.

Este controlador declara la acción mencionada, y éste llama a la clase que describe la acción de impresión *org.openxava.cardiologia.actions.InformePrueba*. Además permite añadir iconos al botón, y se repite esta acción para todos los elementos de una colección (el paciente puede tener 0, 1 o muchos) como se muestra en la siguiente foto:



Captura 80: Captura de pantalla de la sección Pruebas médicas del paciente de la aplicación web.

5.1.7.4 Editores.

En nuestra aplicación necesitamos que determinados valores que se introducen muestre al lado su unidad de medida, ya que sino el especialista médico no sabe si lo está insertando de manera correcta o no, y puede dar lugar a errores en los resultados, informes de la prueba del paciente, etc.

Se muestra un ejemplo de la configuración realizada para que el campo peso muestre a su derecha la medida *Kg*. Debemos añadir tres scripts; crear el fichero que contiene la descripción de los diferentes estereotipos que hay en la aplicación, escribir su clase formateadora, y escribir el archivo JSP que describe cómo se representan los datos.

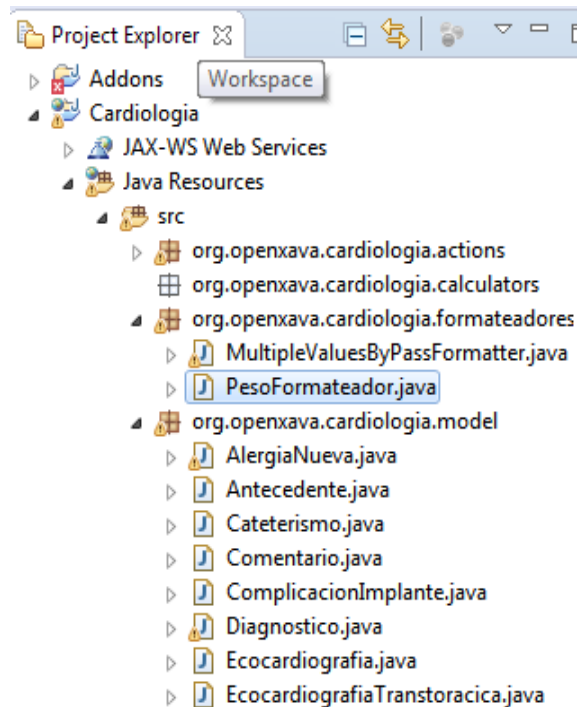
Primer paso. Creamos el fichero *editores.xml* y lo ubicamos en la ruta *..openxava-5.4\openxava-5.4\workspace\Cardiologia\xava\editores.xml*.

```
1 <?xml version = "1.0" encoding = "ISO-8859-1"?>
2
3 <!DOCTYPE editores SYSTEM "dtds/editores.dtd">
4
5 <editores>
6
7   <editor nombre="peso" url="pesoEditor.jsp">
8     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
9     <para-estereotipo estereotipo="PESO"/>
10  </editor>
11
```

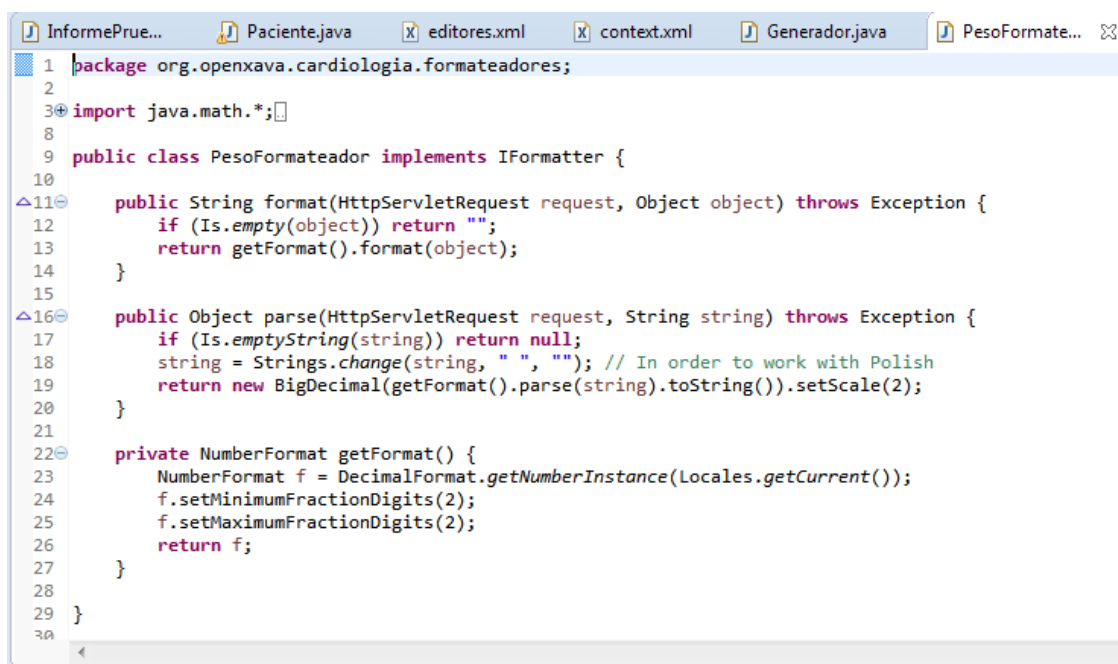
Captura 81: Captura de pantalla donde se muestra el fragmento de código del archivo *editores.xml*.

En este trozo de código estamos declarando el editor peso, el cual se describe en el fichero *pesoEditor.jsp* y contiene la clase formateadora *org.openxava.cardiologia.formateadores.PesoFormateador*.

Segundo paso. Escribimos su clase formateadora *PesoFormateador.java*, ubicada dentro del paquete *org.openxava.cardiologia.formateadores* del proyecto *Cardiologia*:



Captura 82: Captura de pantalla del explorador de paquetes de Eclipse.



Captura 83: Captura de pantalla del contenido de la clase *PesoFormateador.java*.

Tercer paso. Creamos el fichero *pesoEditor.jsp*, ubicado en el directorio que contiene todos los editores de la aplicación `..\openxava-5.4\openxava-5.4\workspace\Cardiologia\web\xava\editors`:

```

1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7 symbol = "Kg"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 symbol = "?";
11 }
12 %>
13
14 <%=symbol%>

```

Captura 84: Contenido del script *pesoEditor.jsp*.

En él, se indica en las líneas 5 hasta la 11 mediante un *scriptlet* de JSP que existe un símbolo *Kg*, que es el que usaremos para representar la unidad de medida de cualquier atributo que declaremos con un estereotipo *peso*, como se puede ver en la clase *paciente*:

```

113 private int telefono2;
114
115 @Column(name="DIRECCION") // La longitud de columna se usa a nivel UI y a nivel DB
116 @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
117 private String direccion;
118
119 @Stereotype("PESO")
120 @Required
121 @Column(name="PESO", length=7)
122 private float peso;
123
124 @Stereotype("METROS")
125 @Required
126 @Column(name="ESTATURA", length=5)
127 private float estatura;
128
129 @Column(name="BSA", length=5)
130 @org.hibernate.annotations.Formula("(peso*estatura)/3600")
131 public float bsa;
132
133 @Column(name="IMC", length=5)
134 @org.hibernate.annotations.Formula("peso/(estatura*estatura)")
135 public float imc;
136
137 @Column(name="SOBREPESO", length=30)

```

Captura 85: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

El efecto que tiene estos *scripts* en las vistas de la ficha paciente al indicar en las líneas resaltadas que el atributo *peso* se representa como un estereotipo *PESO* es el siguiente:

▼ Estado nutricional

Peso	<input checked="" type="checkbox"/>	<input type="text" value="60,00"/>	Kg	
Estatura	<input checked="" type="checkbox"/>	<input type="text" value="1,71"/>	metros	
BSA		<input type="text" value="0,03"/>	IMC	<input type="text" value="20,52"/>
Sobrepeso	<input type="text" value="Peso Normal"/>			

Captura 86: Captura de pantalla del estado nutricional de la ficha de paciente.

Efectivamente, comprobamos que se representa la unidad de medida del peso.

5.1.7.5 Validadores.

Como ya sabemos, corresponde a la lógica de validación que puede usarse en cualquier componente de negocio.

Se expone un ejemplo para entender cómo se validan los datos; los médicos que instalan un generador cardíaco no pueden ser la misma persona. Para ello se debe lanzar una excepción para evaluar la lógica de validación de la variable deseada, y ésta realizarla antes de grabar el dato, tanto al crearlo como actualizarlo.

Para hacer la validación, se escogió la manera más sencilla; dentro de la entidad, añadiendo dos métodos que lanzan el mensaje de error si se cumple la condición.

En nuestro caso, al implantar un generador, el médico y médico ayudante no pueden ser el mismo, y la fecha de explante no puede ser menor que la fecha de implante. Por ello se añade este código dentro de la entidad Generador:


```

369
370 public MedicoAyudante getMedicoAyudante() {
371     return medicoAyudante;
372 }
373
374 //método para lanzar excepción cuando medicoAyudante=medico y/o fechaExplante<fechaImplante al crear objeto
375 @PrePersist //Ejecutado justo antes de grabar el objeto por primera vez
376 private void onPersist() throws Exception{
377     if (medico != null && medicoAyudante != null && medicoAyudante.toString().equals(medico.toString())) { //lógica
378         throw new javax.validation.ValidationException("los_medicos_deben_ser_distintos");
379     }
380     if (fechaExplante != null && fechaExplante.before(fechaImplante) ){
381         throw new javax.validation.ValidationException("fecha_explante_deben_ser_mayor_que_fecha_implante");
382     }
383 }
384
385 //método para lanzar excepción cuando medicoAyudante=medico y/o fechaExplante<fechaImplante al actualizar objet
386 @PreUpdate //Ejecutado al momento de actualizar posteriormente la entidad
387 private void onUpdate() throws Exception{
388     if (medico != null && medicoAyudante != null && medicoAyudante.toString().equals(medico.toString())) { //lógica
389         throw new javax.validation.ValidationException("los_medicos_deben_ser_distintos");
390     }
391     if (fechaExplante != null && fechaExplante.before(fechaImplante) ){
392         throw new javax.validation.ValidationException("fecha_explante_deben_ser_mayor_que_fecha_implante");
393     }
394 }

```

Captura 87: Captura de pantalla de fragmento de código de la clase Generador.java.

Se encarga de que no se cometan estos errores (médicos iguales o fechaExplante<fechaImplante), y se mantiene en el objeto para grabar adecuadamente los campos tras el aviso:

Editar - Generador

Los Médicos no pueden ser iguales

▼ Generador

Modelo	wsdqw	Numero serie	qwsq2	Fabricante	Biotronic
Fecha implante	11/01/2016 00:00	Región implante	Femoral	Acceso	Cefálica Izquierda
Tipo generador	Desfibrilador	Resincro	No	Lugar	Quirófano CCV
Médico	Antonio J. Delgado Espinosa	Médico Ayudante	Antonio J. Delgado Espinosa	Enfermero	Magdalena Rolez Santana
Fecha explante		Causa explante	Agotamiento de Batería		

► Problema (1)

► Complicación implante (1)

▼ Comentario generador

fgdrf

Grabar Quitar Cerrar

Captura 88: Captura de pantalla de la excepción lanzada al comprobar que los médicos son la misma persona.

Además podemos modificar el mensaje para adaptarlo a la sintaxis castellana sin que se muestre la cadena de texto sin tildes, mayúsculas, símbolos, etc, gracias a la Internacionalización (i18n). Esto lo hacemos modificando el fichero *MensajesCardiologia_es.properties*, ubicado en la ruta `..openxava-5.4\openxava-5.4\workspace\Cardiologia\i18n\MensajesCardiologia_es.properties`:

```

1 # Mensajes para la aplicación;
2 welcome_point1=Introduce tu usuario y contraseña.
3 welcome_point2=Accede a los datos de paciente, pruebas médicas cardiológicas, visitas médicas y demás información.
4 welcome_point3=Genera informes médicos de manera rápida y sencilla.
5 los_medicos_deben_ser_distintos=Los médicos no pueden ser iguales
6 fecha_explante_deben_ser_mayor_que_fecha_implante=La fecha de explante debe ser mayor que la fecha de implante
7 fecha_resolucion_deben_ser_mayor_que_fecha_comienzo=La fecha de resolución debe ser mayor que la fecha de comienzo

```

Captura 89: Captura de pantalla del código del archivo *MensajesCardiologia_es.properties*.

5.1.7.6 Calculadores.

Como ya se comentó, un calculador muy recurrido para este proyecto fue el calculador de la fecha actual, por ejemplo, para poner como valor por defecto en las pruebas médicas, o en la instalación del equipo generador cardiológico, el día en curso:

```

46
47 private Fabricante (String nombreFabricante){
48     this.nombreFabricante = nombreFabricante;
49 }
50
51 public String getNombreFabricante() {
52     return nombreFabricante;
53 }
54 }
55
56 @DefaultValueCalculator(CurrentDateCalculator.class)
57 @Column(name="FECHAIMPLANTE")
58 @Stereotype("FECHAHORA")
59 @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
60 private Date fechaImplante;
61
62 @Column(name="FECHAEXPLANTE")
63 @Stereotype("FECHAHORA")
64 @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
65 private Date fechaExplante;
66
67 @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
68 @Column(name="REGIONIMPLANTE")
69 private RegionImplante regionImplante;
70 public enum RegionImplante {
71     ABDOMINAL ("Abdominal"), FEMORAL("Femoral"), PREPECTORAL("Prepectoral"), SUBPECTORAL("Subpectoral");

```

Captura 90: Captura de pantalla de fragmento de código de la clase *Generador.java*.

Se observa que en la línea 56, el atributo *fechaImplante* tiene por defecto, el valor de la fecha actual, gracias a la clase *CurrentDateCalculator.class*.


Captura 91: Captura de pantalla de un fragmento de la ficha Generador del paciente.

5.1.8 Uso de las anotaciones JPA, Openxava y API JPA.

5.1.8.1 Anotaciones JPA.

5.1.8.1.1 Anotación para definir las clases del modelo de datos como Entidad.

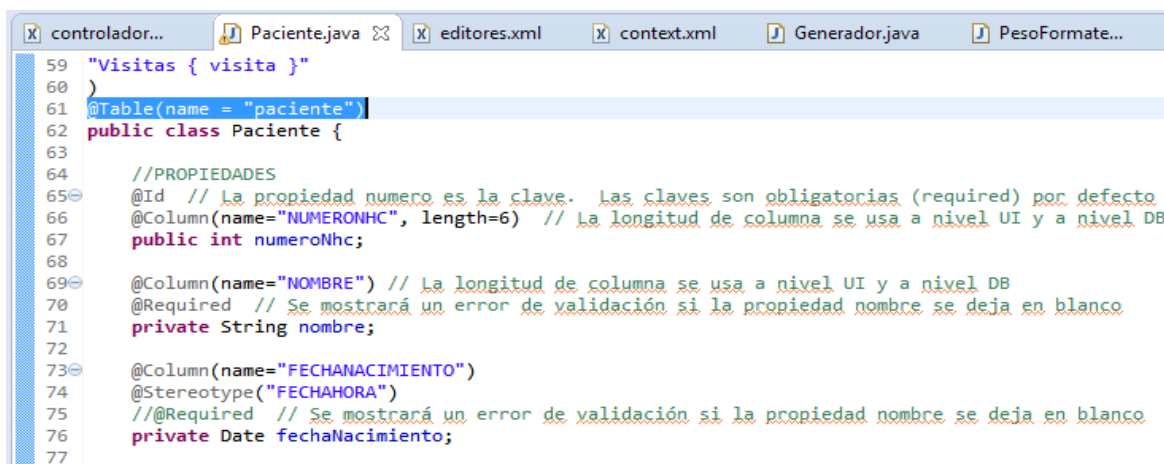
Para definir una clase persistente le agregamos la anotación que la identifica como entidad; una clase cuyas instancias se graban en la base de datos.



```
1 package org.openxava.cardiologia.model;
2
3 import javax.persistence.*;
4 import org.openxava.annotations.*;
5 import org.openxava.jpa.*;
6 import java.util.*;
7
8 @Entity
9 @View(members =
10 "DatosDemograficos [" +
11 " numeroNhc, nombre;" +
12 " fechaNacimiento, sexo;" +
13 " telefono1, telefono2;" +
14 " direccion;" +
15 "]" +
16 "EstadoNutricional [" +
17 " peso;" +
18 " estatura;" +
19 " bsa, imc;" +
20 " sobrepeso;" +
21 "]" +
22 "FichaDePaciente {" +
23 "Antecedentes { antecedente }" +
24 "Alergias { alergiaNueva }" +
25 "Indicaciones { indicacion }" +
26 "Procedimientos { procedimiento }" +
```

Captura 92: Fragmento de código de la clase *Paciente.java*.

Se observa cómo en la línea 8, a la clase se la identifica como una entidad gracias a la anotación **@Entity**. Todas las clases del modelo de datos tendrán esta etiqueta, puesto que todas tienen su tabla correspondiente en la BB.DD. de cardiología.



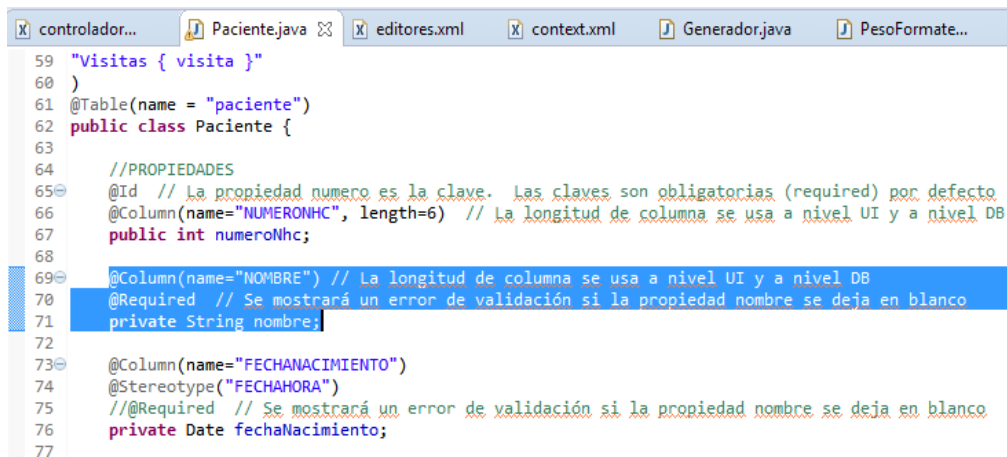
```
59 "Visitas { visita }"
60 )
61 @Table(name = "paciente")
62 public class Paciente {
63
64     //PROPIEDADES
65     @Id // La propiedad numero es la clave. Las claves son obligatorias (required) por defecto
66     @Column(name="NUMERONHC", length=6) // La longitud de columna se usa a nivel UI y a nivel DB
67     public int numeroNhc;
68
69     @Column(name="NOMBRE") // La longitud de columna se usa a nivel UI y a nivel DB
70     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
71     private String nombre;
72
73     @Column(name="FECHANACIMIENTO")
74     @Stereotype("FECHAHORA")
75     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
76     private Date fechaNacimiento;
77
```

Captura 93: Fragmento de código de la clase *Paciente.java*.

Haciendo uso de la anotación **@Table** indicamos el nombre de la tabla de la BB.DD. donde estarán los datos de paciente. También usamos esta anotación en cada una de las clases del modelo de datos.

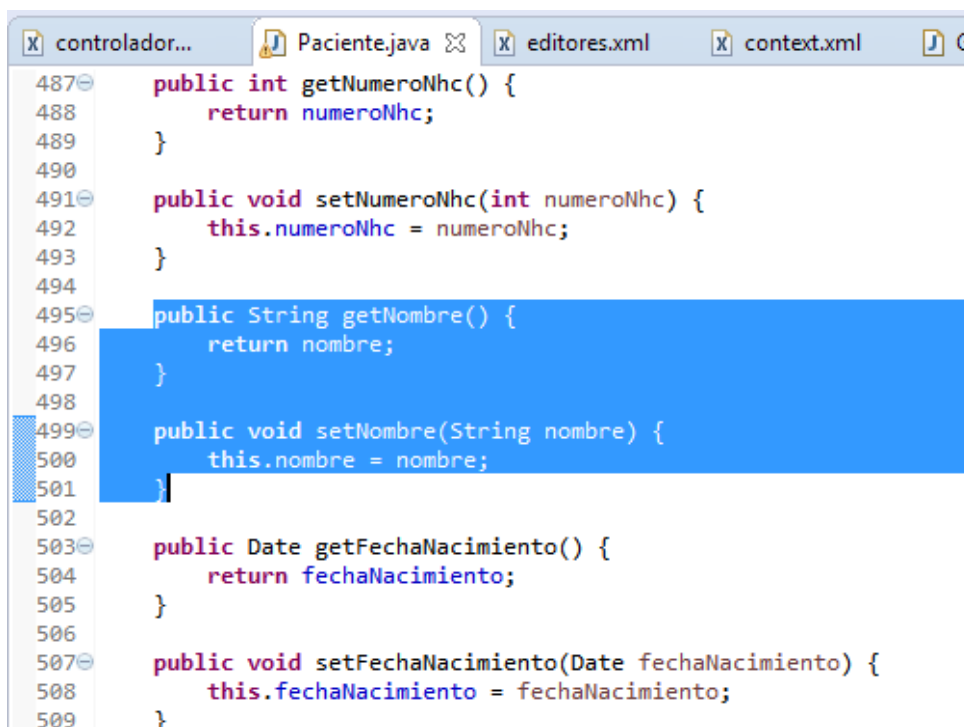
5.1.8.1.2 Propiedades de las clases de la aplicación web.

Las características de una entidad se representa mediante propiedades, y se escriben como propiedades Java convencionales, con **getters** y **setters**.



```
59 "Visitas { visita }"
60 )
61 @Table(name = "paciente")
62 public class Paciente {
63
64     //PROPIEDADES
65     @Id // La propiedad numero es la clave. Las claves son obligatorias (required) por defecto
66     @Column(name="NUMERONHC", length=6) // La longitud de columna se usa a nivel UI y a nivel DB
67     public int numeroNhc;
68
69     @Column(name="NOMBRE") // La longitud de columna se usa a nivel UI y a nivel DB
70     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
71     private String nombre;
72
73     @Column(name="FECHANACIMIENTO")
74     @Stereotype("FECHAHORA")
75     //@Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
76     private Date fechaNacimiento;
77
```

Captura 94: Captura de pantalla de una propiedad requerida de la clase *Paciente.java*.



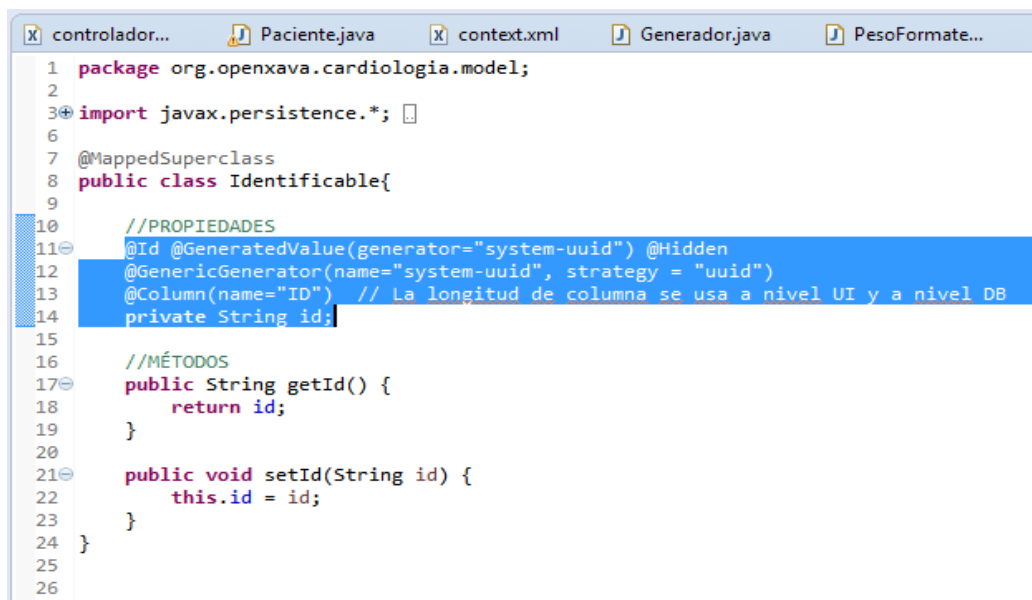
```
487 public int getNumeroNhc() {
488     return numeroNhc;
489 }
490
491 public void setNumeroNhc(int numeroNhc) {
492     this.numeroNhc = numeroNhc;
493 }
494
495 public String getNombre() {
496     return nombre;
497 }
498
499 public void setNombre(String nombre) {
500     this.nombre = nombre;
501 }
502
503 public Date getFechaNacimiento() {
504     return fechaNacimiento;
505 }
506
507 public void setFechaNacimiento(Date fechaNacimiento) {
508     this.fechaNacimiento = fechaNacimiento;
509 }
```

Captura 95: Captura de pantalla de los métodos de propiedad *nombre* requerida de la clase *Paciente.java*.

5.1.8.1.3 Clave primaria de la clase.

Para indicar cuál propiedad es la clave primaria de la entidad usamos la anotación **@Id**. Ésta es siempre obligatoria.

Para ello, en este proyecto se describe una clase *Identificable.java*, la cual contiene un atributo *Id* oculto con las anotaciones **@Hidden** y **@Id** y sus respectivos métodos. Esta clase se usó para que todas las demás clases (excepto Paciente, ya que su *Id* coincide con el número de historia que hay en la BB.DD. Drago) extiendan de ella y hereden sus propiedades y métodos.



```
1 package org.openxava.cardiologia.model;
2
3 import javax.persistence.*;
4
5
6
7 @MappedSuperclass
8 public class Identificable{
9
10 //PROPIEDADES
11 @Id @GeneratedValue(generator="system-uuid") @Hidden
12 @GenericGenerator(name="system-uuid", strategy = "uuid")
13 @Column(name="ID") // La longitud de columna se usa a nivel UI y a nivel DB
14 private String id;
15
16 //MÉTODOS
17 public String getID() {
18     return id;
19 }
20
21 public void setID(String id) {
22     this.id = id;
23 }
24 }
25
26
```

Captura 96: Captura de pantalla del código de la clase *Identificable.java*.

Además, podemos ver en la línea 11 y 12 que gracias a las anotaciones **@GeneratedValue** y **@GenericGenerator** indicamos que el *Id* se genera de forma automática y que la secuencia que los genera les da un valor único.

5.1.8.1.4 Especificación de la columna de la tabla.

Para indicar el nombre y la longitud de la columna donde persiste el atributo de la clase usamos la anotación **@Column**.

```

129 @Column(name="BSA", length=5)
130 @org.hibernate.annotations.Formula("(peso*estatura)/3600")
131 public float bsa;
132
133 @Column(name="IMC", length=5)
134 @org.hibernate.annotations.Formula("peso/(estatura*estatura)")
135 public float imc;
136
137 @Column(name="SOBREPESO", length=30)
138 @org.hibernate.annotations.Formula("imc")
139 private String sobrepeso;
140

```

Captura 97: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

En la línea 133 de código vemos cómo se mapea el atributo *imc* con la columna *IMC* de la BB.DD. Además se indica que la longitud máxima de éste es 5, valor que se usa a nivel interfaz de usuario para representar el máximo del input de *imc*, y el tamaño máximo con que se guardará en la BB.DD.

▼ Estado nutricional

Peso Kg

Estatura metros

BSA **IMC**

Sobrepeso

Captura 98: Captura de pantalla del valor IMC.

Se aprecia que es un valor que no alcanza un tamaño mayor de 5 cifras.

5.1.8.1.5 Referencias.

Una entidad puede hacer referencia a otra entidad, usando la anotación JPA **@ManyToOne**.

```

240 @stereotype("MEMO") // Esto es para un texto grande, se usara un area de texto o equivalente
241 @Column(name="COMENTARIOGENERADOR")
242 private String comentarioGenerador;
243
244
245 //pag. 34 Openxava con Ejemplos
246 //acceder al nombre de Usuario para almacenar por Seguridad
247 //@ReadOnly
248 //private String userId;
249
250 //REFERENCIAS
251 @ManyToOne(fetch=FetchType.LAZY) // La referencia tiene que tener valor siempre. No permite Generadores sin Paciente
252 @JoinColumn(
253     name="numeroNhc",
254     nullable = true,
255     foreignKey = @ForeignKey(name = "fk_generador_numeroNhc") // numeroNhc es la columna para la clave foránea
256     //@DescriptionList //Muestra la referencia usando un combo, página OpenXava con ejemplos página 32
257 private Paciente generadoresLocal; // Una referencia Java convencional
258
259 //CLASE INCRUSTADA
260 @OneToOne
261 @JoinColumn(
262     name = "idtc",
263     nullable = true,
264     foreignKey = @ForeignKey(name = "fk_generador_idtc"))
265 @AsEmbedded

```

Captura 99: Captura de pantalla de fragmento de código de la clase *Generador.java*.

Entre las líneas 251 y 257 podemos ver que se relaciona la clase generador con la clase paciente, con una relación muchos a uno, ya que paciente puede tener 0, 1 o muchos generadores. En la línea 253 se especifica la columna clave foránea con la cual se relacionan dichas tablas.

A su vez, hay que definir la misma relación pero a la inversa en la clase Paciente con la anotación **@OneToMany**:

```

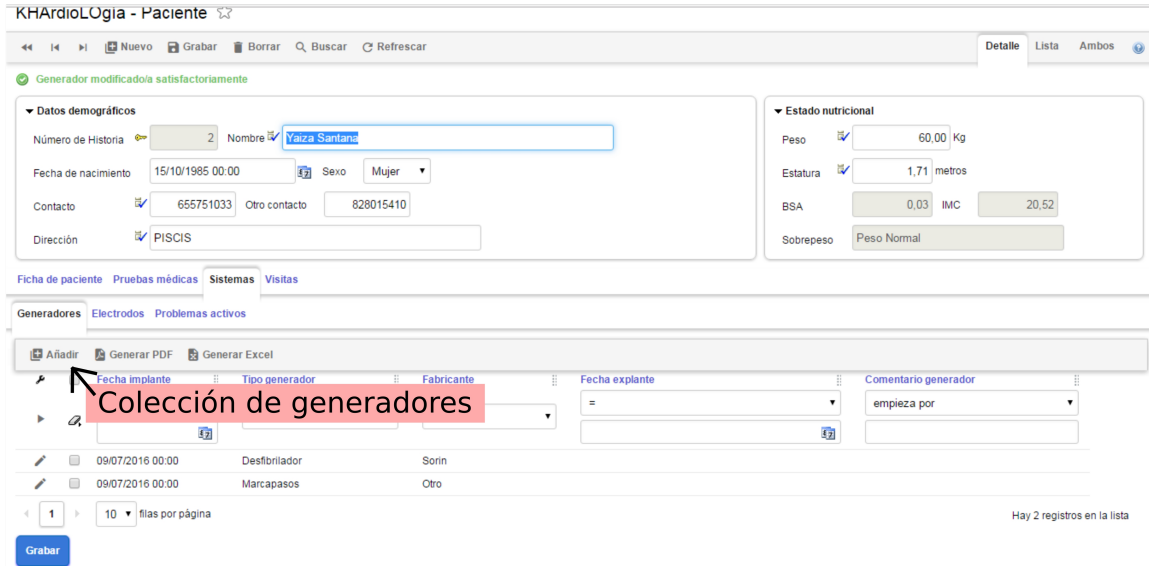
425 @RemoveAction("")
426 @RemoveSelectedAction("")
427 @OneToMany (mappedBy="holtersLocal", cascade=CascadeType.ALL)
428 private Collection<Holter> holter = new ArrayList<Holter>();
429
430 @RemoveAction("")
431 @RemoveSelectedAction("")
432 @OneToMany (mappedBy="resincrosLocal", cascade=CascadeType.ALL)
433 private Collection<ResincronizacionCardiaca> resincronizacionCardiaca = new ArrayList<ResincronizacionCardiaca>();
434
435 @RemoveAction("")
436 @RemoveSelectedAction("")
437 @OneToMany (mappedBy="generadoresLocal", cascade=CascadeType.ALL)
438 private Collection<Generador> generador = new ArrayList<Generador>();
439
440 @RemoveAction("")
441 @RemoveSelectedAction("")
442 @OneToMany (mappedBy="electrodosLocal", cascade=CascadeType.ALL)
443 private Collection<Electrodo> electrodo = new ArrayList<Electrodo>();
444
445 @RemoveAction("")
446 @RemoveSelectedAction("")
447 @OneToMany (mappedBy="visitasLocal", cascade=CascadeType.ALL)
448 private Collection<Visita> visita = new ArrayList<Visita>();
449
450 //COLECCIONES CALCULADAS

```

Captura 100: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

En la línea señalada 437 se aprecia como mapea *generadoresLocal*, para indicar que esta referencia se almacena en la BB.DD como una relación uno a muchos entre la tabla

Paciente y la tabla para *Generador*. Además, se indica que si borramos un elemento *Paciente*, se borran sus elementos *Generadores*, con el código *cascade=CascadeType.ALL*.



Captura 101: Captura de pantalla de la aplicación web, de la sección Sistemas, ficha generadores del paciente.

En la fotografía anterior vemos el botón añadir que permite insertar más registros a la tabla *Generador* vinculados al paciente, y cómo se muestra la colección de elementos como una lista.

5.1.8.1.6 Clases embebidas.

Además de las entidades usamos las clases incrustables. Éstas se definen con la anotación **@OneToOne** y no pueden existir si no existe la clase principal; la clase donde es embebida.


```

351     }
352
353     //COMENTARIO OUTCOMES
354     @Column(name="COMENTARIOOUTCOMES", nullable = true)
355     private String comentarioOutcomes;
356
357     //INCRUSTABLES
358     @OneToOne
359     @JoinColumn(
360         name = "id",
361         nullable = true,
362         foreignKey = @ForeignKey(name = "fk_paciente_id"))
363     @AsEmbedded
364     private Comentario comentario;
365
366     //COLECCIONES
367     // @EditAction("Paciente.editarAlergia")
368     @RemoveAction("")
369     @RemoveSelectedAction("")
370     @SaveAction("lopd.Guardar")
371     @OneToMany(mappedBy="alergiasNuevasLocal", cascade=CascadeType.ALL)
372     private Collection<AlergiaNueva> alergiasNueva = new ArrayList<AlergiaNueva>();
373
374     @RemoveAction("")
375     @RemoveSelectedAction("")
376     @OneToMany(mappedBy="anteredentesLocal", cascade=CascadeType.ALL)

```

Captura 102: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

Entre la línea 358 y la línea 364 vemos cómo se embebe la clase *Comentario* dentro de la clase *Paciente* con la anotación **@AsEmbedded**. Éste es un campo de texto en el que se anotan observaciones generales del paciente.

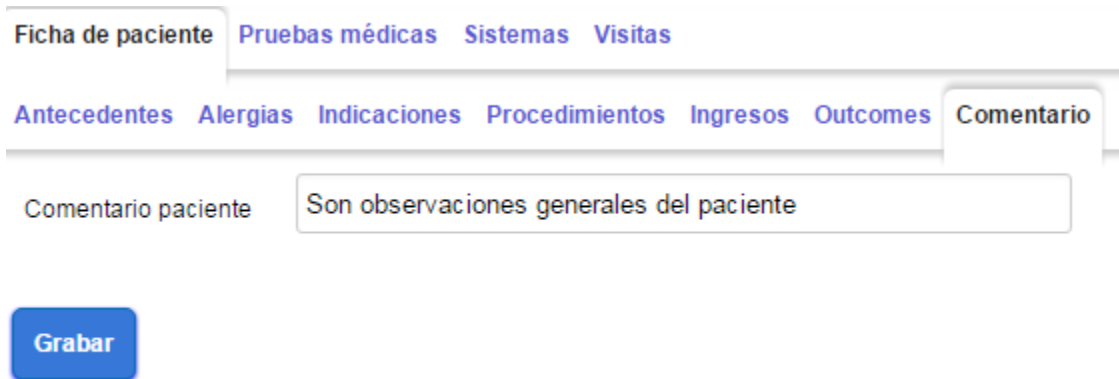
En la clase *Comentario.java*, en cambio, no fue necesario añadir otra anotación como ocurre con las colecciones anteriores.

```

1 package org.openxava.cardiologia.model;
2 import javax.persistence.*;
3
4 @Entity
5 @Table(name = "comentario")
6 public class Comentario extends Identificable{
7
8     //PROPIEDADES
9     @Column(name="COMENTARIOPACIENTE", nullable = true)
10    private String comentarioPaciente;
11
12    //MÉTODOS
13    public String getComentarioPaciente() {
14        return comentarioPaciente;
15    }
16
17    public void setComentarioPaciente(String comentarioPaciente) {
18        this.comentarioPaciente = comentarioPaciente;
19    }
20
21 }

```

Captura 103: Captura de pantalla del código de la clase *Comentario.java*.



Captura 104: Captura de pantalla de la aplicación web, de la sección Ficha de paciente, observaciones generales del paciente.

5.1.8.1.7 Clases transitorias.

Estas clases no se almacenan en la BB.DD, y pueden usarse para acceder a información de sólo lectura. Un ejemplo de esta anotación **@Transient** la vemos en la clase *Paciente*, para acceder a los problemas activos de los generadores cardiológicos que pueda tener un paciente de esta especialidad médica.

```

1 //COLECCIONES CALCULADAS
2 @Transient
3 @ReadOnly
4 public Collection<Problema> getProblemilla() throws Exception {
5     int numero = (int) getNumeroNhc();
6     Query query = Xpersistence.getManager().createQuery("from Problema p where p.nhc = :numero and p.fechaResolucion = null");
7     query.setParameter("numero", numero);
8     if (query.getResultList().size()==0) return Collections.EMPTY_LIST;
9     return query.getResultList();
10 }

```

Captura 105: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

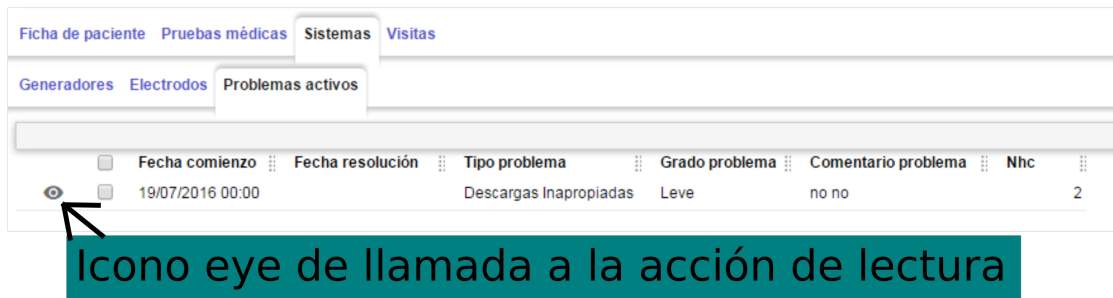
En esta declaración los métodos `get` y `set` van dentro de la colección. Así mismo, se especifica la condición:

"from Problema p where p.nhc = :numero and p.fechaResolucion = null"

que crea la consulta JPA con el código:

Query query = Xpersistence.getManager().createQuery(...

Finalmente vemos como tenemos la lista de problemas activos que son sólo lectura, indicado anteriormente con la anotación **@ReadOnly**, e indicado visualmente con el icono ojo para cada fila que permite ver los detalles de cada problema.



Captura 106: Captura de pantalla de la aplicación web, de la sección Sistemas, ficha de problemas activos del paciente.



Captura 107: Captura de pantalla de la llamada a la acción de lectura de los detalles del problema activo.

Una vez revisadas las anotaciones JPA usadas para escribir las clases, a continuación explicaremos como se usó el API JPA para leer y escribir de/en la base de datos de desde el código java.

5.1.8.2 API JPA.

La clase de JPA más importante es ***javax.persistence.EntityManager***. El ***EntityManager*** [101] nos permite grabar, modificar y buscar entidades.

En nuestras clases importaremos varias librerías para poder hacer uso de las funciones más importantes para la persistencia de los datos, entre otros:

```
import javax.persistence.*;
import org.openxava.annotations.*;
import org.openxava.jpa.*;
import java.util.*;
```

setProperty

```
void setProperty(java.lang.String propertyName,
                java.lang.Object value)
```

Set an entity manager property or hint. If a vendor-specific property or hint is not recognized, it is silently ignored.

Parameters:

propertyName - name of property or hint
value - value for property or hint

Throws:

IllegalArgumentException - if the second argument is not valid for the implementation

Since:

Java Persistence 2.0

getProperties

```
java.util.Map<java.lang.String,java.lang.Object> getProperties()
```

Get the properties and hints and associated values that are in effect for the entity manager. Changing the contents of the map does not change the configuration in effect.

Returns:

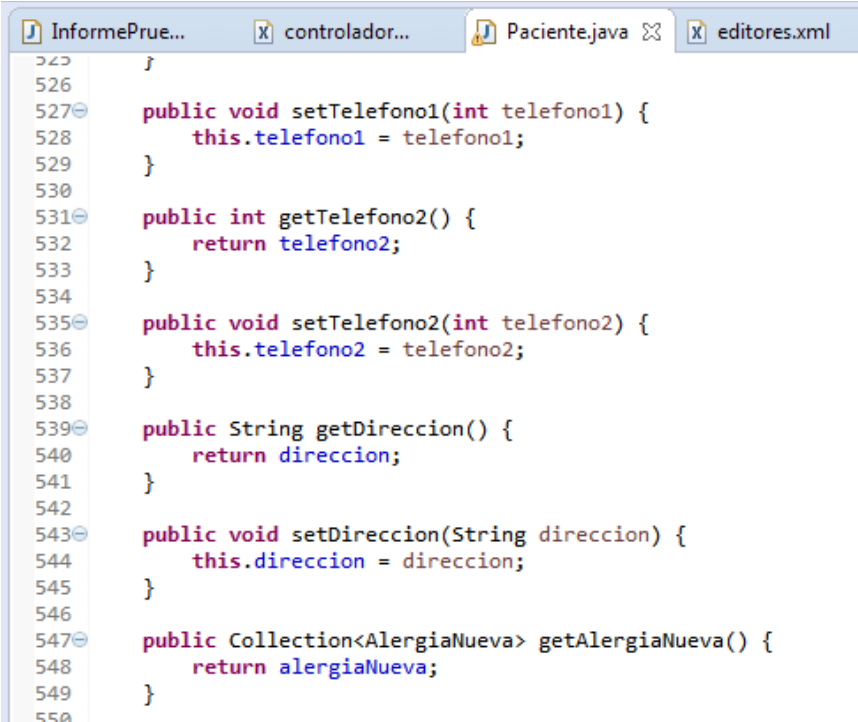
map of properties and hints in effect for entity manager

Since:

Java Persistence 2.0

Figura 54: Especificaciones de los métodos *setProperty*, *getProperty* de la interfaz *EntityManager* [102].

Gracias a la primera importación, podemos hacer uso de los métodos *getter*, *setter* de cada una de las propiedades de la clase persistente como vemos en el siguiente código:



```
InformePrue... controlador... Paciente.java editores.xml
526     }
527     public void setTelefono1(int telefono1) {
528         this.telefono1 = telefono1;
529     }
530
531     public int getTelefono2() {
532         return telefono2;
533     }
534
535     public void setTelefono2(int telefono2) {
536         this.telefono2 = telefono2;
537     }
538
539     public String getDireccion() {
540         return direccion;
541     }
542
543     public void setDireccion(String direccion) {
544         this.direccion = direccion;
545     }
546
547     public Collection<AlergiaNueva> getAlergiaNueva() {
548         return alergiaNueva;
549     }
550
```

Captura 108: Captura de pantalla de fragmento de código de los métodos de la clase *Paciente.java*.

5.1.8.3 Anotaciones *Openxava* para las vistas en la interfaz de usuario.

Openxava genera una interfaz de usuario por defecto, aunque nos permite modificar la disposición de los elementos para poder navegar dentro de la aplicación web de manera más intuitiva.

Por citar un ejemplo, vemos cómo hemos organizado los atributos de paciente para darle un orden lógico, ordenándolos por datos demográficos, estado nutricional, o por secciones. Esto lo hemos conseguido con las anotaciones de *Openxava* y su nomenclatura para asignar la disposición que tendrán; usando comas para separar los elementos, punto y coma para salto de línea, corchetes para grupos (marcos), o llaves para secciones (pestañas):

The screenshot displays a patient record interface. At the top, there are two main sections: 'Datos demográficos' and 'Estado nutricional'. The 'Datos demográficos' section includes fields for 'Número de Historia' (2), 'Nombre' (Yaiza Santana), 'Fecha de nacimiento' (15/10/1985 00:00), 'Sexo' (Mujer), 'Contacto' (655751033), 'Otro contacto' (828015410), and 'Dirección' (PISCIS). The 'Estado nutricional' section includes 'Peso' (60,00 Kg), 'Estatura' (1,71 metros), 'BSA' (0,03), 'IMC' (20,52), and 'Sobrepeso' (Peso Normal). Below these sections is a navigation bar with tabs for 'Ficha de paciente', 'Pruebas médicas', 'Sistemas', and 'Visitas'. Under 'Ficha de paciente', there are sub-tabs for 'Antecedentes', 'Alergias', 'Indicaciones', 'Procedimientos', 'Ingresos', 'Outcomes', and 'Comentario'. The 'Antecedentes' tab is active, showing a table of medical history items. The table has columns for 'Fecha antecedente', 'Tipo antecedente', and 'Comentario antecedente'. The table contains six rows of data, including 'Asma Bronquial', 'AIT', 'Otro', 'DM1', 'AIT', and 'Bebedor Activo'. At the bottom of the table, there is a pagination control showing '1' of '10' items per page, and a 'Grabar' button. A note at the bottom right says 'Hay 6 registros en la lista'.

Captura 109: Captura de pantalla de la interfaz de usuario de la aplicación web.

Si analizamos por partes, para definir el grupo de los datos demográficos tenemos el siguiente código, donde cada línea de código corresponde con la fila, y cada elemento de la línea, una columna:

```

7
8 @Entity
9 @View{members =
10 "DatosDemograficos [" +
11 " numeroNhc, nombre;" +
12 " fechaNacimiento, sexo;" +
13 " telefonol, telefono2;" +
14 "direccion;" +
15 "]" +

```

Captura 110: Captura de pantalla de un fragmento de código de la clase *Paciente.java*.

Para las secciones, tenemos por ejemplo las líneas de código desde la 44 hasta la 53 donde se dispone la sección Pruebas médicas en concreto:

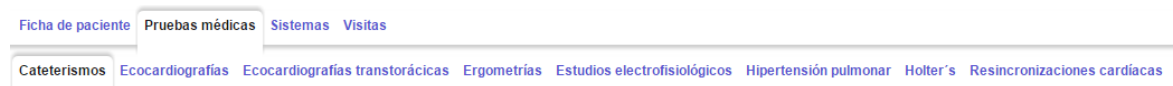
```

43 "]" +
44 "PruebasMedicas {" +
45 "Cateterismos { cateterismo }" +
46 "Ecocardiografías { ecocardiografia}" +
47 "EcocardiografiasTranstoracicas {ecocardiografiaTranstoracica}" +
48 "Ergometrias { ergometria}" +
49 "EstudiosElectrofisiologicos { estudioElectrofisiologico }" +
50 "HipertensionPulmonar { hipertensionPulmonar }" +
51 "Holter's { holter }" +
52 "ResincronizacionesCardiacas { resincronizacionCardiaca}" +
53 "}" +
54 "Sistemas {" +
55 "Generadores { generador }" +
56 "Electrodos { electrodo }" +
57 "ProblemasActivos { problemilla }" +
58 "}" +
59 "Visitas { visita }"
60 )
61 @Table(name = "paciente")
62 public class Paciente {
63

```

Captura 111: Captura de pantalla de un fragmento de código de la clase *Paciente.java*.

El efecto que tiene este código es el siguiente:



Captura 112: Captura de pantalla de la sección Pruebas médicas.

5.1.9 Configuración de AJAX.

Existen muchas funciones ya predefinidas para AJAX, aunque en este proyecto su

programación será transparente para el desarrollador, ya que no se modificó esta parte del marco del trabajo *Openxava*.

5.1.10 Modificación del nivel de presentación de los datos de *Openxava*.

Para este proyecto fin de carrera, se modificó la cabecera de la aplicación.

Para ello debimos, por un lado, crear un *script* que contuviera la descripción de la cabecera, y por otro lado, incluir este contenido en la portada de la aplicación. El contenido del *script* de cabecera es el siguiente:

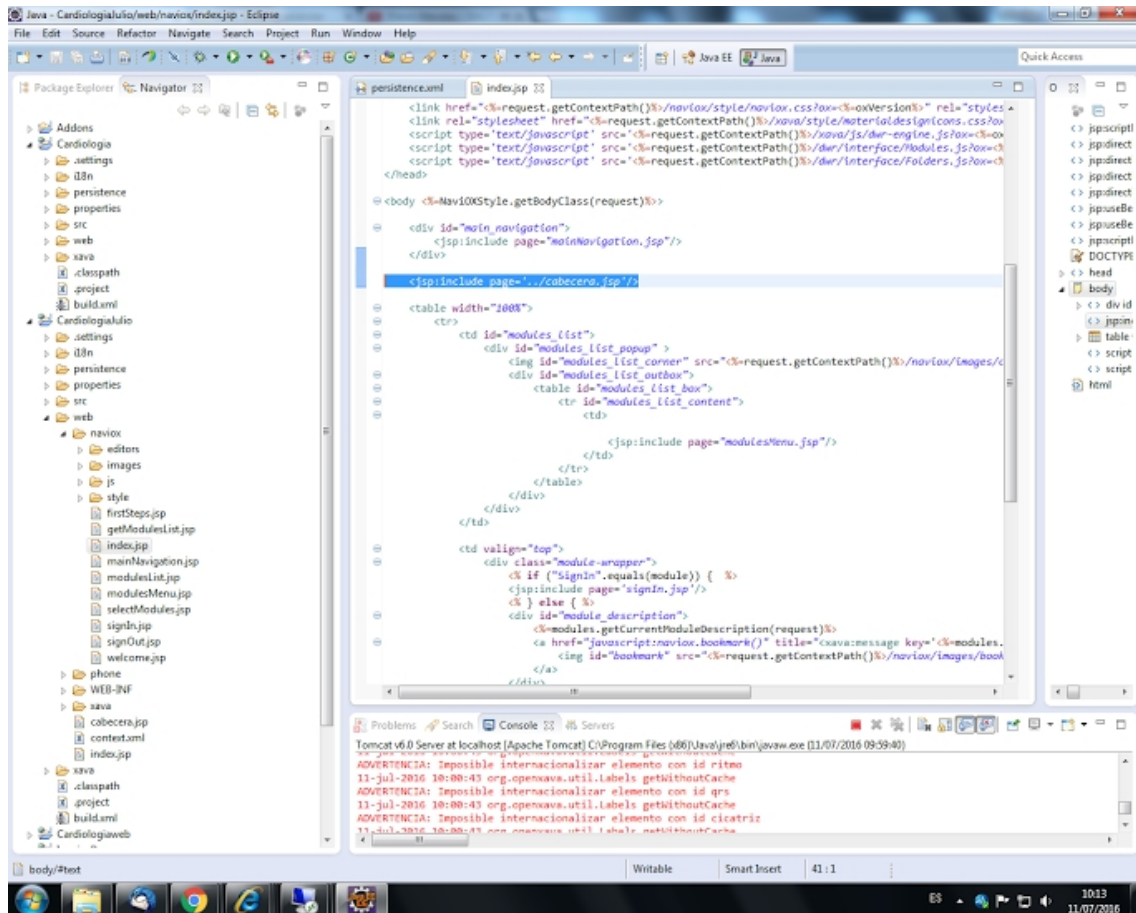
```
1 <style type="text/css">
2
3 .Estilol {
4   color: #000066;
5   font-weight: bold;
6   font-size: 25px;
7 }
8 </style>
9
10 <table width="100%" height="100" border="0" align="center"><!--etiqueta de inicio de la tabla con una fila y tres columnas -->
11 <tr><!--etiqueta de inicio de la fila de la tabla-->
12 <td width="15%" height="100"><div align="left"></div></td><!--etiqueta de inicio, contenido y etiqueta de fin de la 1ª columna de la tabla-->
14 <td width="85%"><div align="center"><span class="Estilol"> CARDIO WEB-ESTIM <BR />Gestión de Pruebas Especiales e Implantes
15 Cardiológicos </span></div></td><!--etiqueta de inicio, contenido y etiqueta de fin de la 2ª columna de la tabla-->
16 <td width="25%"><div align="right"></div></td>
17 </tr><!--etiqueta de fin de la fila de la tabla-->
18 </table><!--etiqueta de fin de la tabla con una fila y tres columnas -->
```

Captura 113: Captura de pantalla de fragmento de código de *cabecera.jsp*.

Para entender cada línea anterior hemos comentado en color verde qué funcionalidad tiene cada trozo de código. En definitiva, la cabecera será una tabla con una fila, y tres columnas. Para la primera columna, se indica la ruta del servidor donde se encuentra el primer logo que se debe visualizar y sus dimensiones, para la segunda columna, el nombre de la aplicación web, y para la tercera columna, se indica la ruta del servidor donde se encuentra el segundo logo que se debe visualizar y sus dimensiones.

Cabecera.jsp lo incluimos en el mismo directorio donde está la portada, *index.jsp*, en la ruta *openxava-5.4\openxava-5.4\workspace\Cardiologia\web\naviox\cabecera.jsp*. Ahora debemos incluir este código en *index.jsp* como vimos en el capítulo anterior dedicado a JSP:

```
<jsp:include page='../cabecera.jsp'/>
```



Captura de pantalla de Eclipse donde se modificó el *script index.jsp*.

5.1.11 Configuración para el uso de la interfaz gráfica *Ireport*.

Para poder usar esta interfaz con nuestro marco de trabajo, además de instalar el software *Ireport* en el equipo de desarrollo, tuvimos que obtener algunos recursos de los proveedores de *Java* e *IText*.

- Instalación del *Java SE Development Kit* (JDK) correcto [103], y especificar la ruta de ubicación de este JDK en el archivo de configuración *ireport.config* de esta interfaz:

default location of JDK/JRE, can be overridden by using --jdkhome <dir> switch

jdkhome="C:\Program Files\Java\jdk1.7.0_79"

- Añadir la librería correcta de *iText*, ya que las que existen por defecto en el marco de trabajo no eran válidas; *iText-2.1.7.js2.jar*.

- Inclusión del archivo *joda-time-2.4.jar* en la ruta del del proyecto `..\openxava-5.4\openxava-5.4\workspace\Cardiologia\web\WEB-INF\lib`, el cual incluye código marco desarrollado para poder usar *JasperReport*.

5.1.11.1 Clase informe.

Una vez hecho estos primeros pasos de instalación, escribimos la clase en la cual obtenemos los datos necesarios para la realización del informe médico y además incluimos la plantilla *.jxml* que contiene el diseño para este reporte. Para ello, nos enfrentamos a varios problemas; recoger los datos de la vista, y traer los datos desde la base de datos del objeto seleccionado.

5.1.11.2 Obtener datos de las vistas.

Para exportar al informe los datos demográficos del paciente para añadirlos como identificación, hicimos uso de las librerías *Openxava* que permiten interactuar con los datos de las vistas, llamados de ahora en adelante en este capítulo, **parámetros**. En concreto, usamos la clase *ViewBaseAction.java*, que tiene una propiedad *view* que permite manejar la interfaz de usuario. En este caso estamos obteniendo los atributos *nombre*, *numeroNhc*, *telefono1* y *direccion* en las líneas 45, 46, 47 y 48 con la función del marco *Openxava* ***getView().getValue(parametro)***.

```

1 package org.openxava.cardiologia.actions;
2
3 import java.util.*;
4
5 import javax.ejb.*;
6 import javax.inject.*;
7
8 import net.sf.jasperreports.engine.*;
9 import net.sf.jasperreports.engine.data.*;
10
11 import org.apache.commons.logging.Log;
12 import org.apache.commons.logging.LogFactory;
13 import org.openxava.actions.*;
14 import org.openxava.model.*;
15 import org.openxava.tab.*;
16 import org.openxava.cardiologia.model.*;
17 import org.openxava.util.*;
18 import org.openxava.validators.*;
19 import org.openxava.view.*;
20 import org.openxava.hibernate.*;
21 import net.sf.jasperreports.engine.JRDataSource;
22 import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
23
24 public class InformePrueba extends JasperReportBaseAction{
25
26     //clases que necesito para el informe
27     private Cateterismo cateterismo;
28
29     //código necesario para que no salte error de viewObject
30     @Inject
31     private Tab tab;
32     private int row = -1;
33     private String viewObject;
34

```

Captura 114: Captura de pantalla de la clase InformePrueba.java donde se incluyen las librerías *org.openxava.view.**

```

44     Map parametros = new HashMap();
45     parametros.put("nombre", getView().getValue("nombre"));
46     parametros.put("numeronhc", getView().getValue("numeroNhc"));
47     parametros.put("telefonol", getView().getValue("telefonol"));
48     parametros.put("direccion", getView().getValue("direccion"));
49     //ahora recojo la clave del elemento de la colección (datos tabulares) selecciona y lo guardo en key
50     View view =(View) getContext().get(getRequest(), viewObject);
51     Map key = (Map) view.getCollectionTab().getTableModel().getObjectAt(getRow());
52     //guardo el informe seleccionado en cateterismo
53     cateterismo = (Cateterismo) MapFacade.findEntity("Cateterismo",key);
54     parametros.put("id", cateterismo.getId());
55     parametros.put("fecharealizacion", cateterismo.getFechaRealizacion());
56     //parametros.put("informe", getCateterismo().getInforme());
57     //parametros.put("medico", getCateterismo().getMedico());
58     return parametros;
59 }

```

Captura 115: Captura de pantalla de la clase InformePrueba.java donde se recogen los valores de las vistas.

```

1  package org.openxava.actions;
2
3  import java.util.*;
4
5  import javax.inject.*;
6
7  import org.apache.commons.logging.*;
8  import org.openxava.util.*;
9  import org.openxava.view.*;
10
11  /**
12   *
13   * @author Javier Paniza
14   */
15
16  abstract public class ViewBaseAction extends BaseAction {
17
18      private static Log log = LoggerFactory.getLog(ViewBaseAction.class);
19
20      @Inject
21      private View view;
22      @Inject
23      private Stack previousViews;
24      private boolean dialogShown = false;
25      private boolean hasNextControllers = false;
26
27      /**
28       * Creates a new view and shows it. <p>
29       *
30       * After it if you call to getView() it will return this new view.<br>
31       *
32       * @since 4m1

```

Captura 116: Captura de pantalla del atributo *view* de la clase *ViewBaseAction.java*.

También fue necesario saber qué elemento de la colección estamos seleccionando, para acceder a los datos de este registro de la BB.DD de cardiología una vez pasado este valor a la plantilla *.jxml*. Esto lo conseguimos con las funciones *Openxava*;

```

/**
 * Extract from the viewObject the name of the collection. <p>
 *
 * Useful for using Tab actions for collections. <br>
 */
public void setViewObject(String viewObject) {
    this.viewObject = viewObject;
}

/**
 * This property has value when the action has been clicked from the row. <p>
 *
 * If not its value is -1.
 */
public int getRow() {
    return row;
}

public void setRow(int row) {
    this.row = row;
}

```

Captura 117: Captura de pantalla de código de clase *InformePrueba.java*.

Por último, en el siguiente código de la clase, indicamos dónde se encuentra la plantilla que contiene el diseño del informe médico:

```
79     protected String getJRXML() { // 5
80         //return "informe_cateterismo.jrxml";
81         return "C:\\openxava-5.4\\openxava-5.4\\informes\\ejemploIreport2.jrxml";
82         // Para leer del sistema de ficheros
83     }
84
85     /*
```

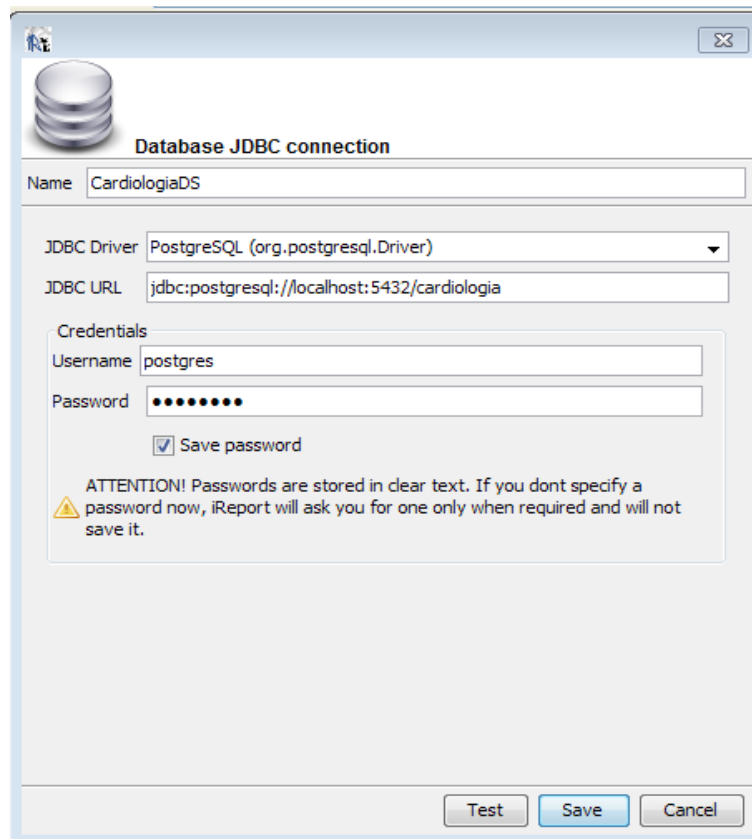
Captura 118: Captura de pantalla de código de clase *InformePrueba.java* donde se obtiene la plantilla generada con *Ireport*.

5.1.11.3 Obtener datos de la BB.DD.

Para exportar al informe los datos del registro que el usuario ha seleccionado, a la hora de crear la plantilla debemos:

5.1.11.3.1 Crear la conexión con la BB.DD a través del JDBC.

Debemos indicar el tipo de Driver, y autenticación del usuario para conectar con los datos.

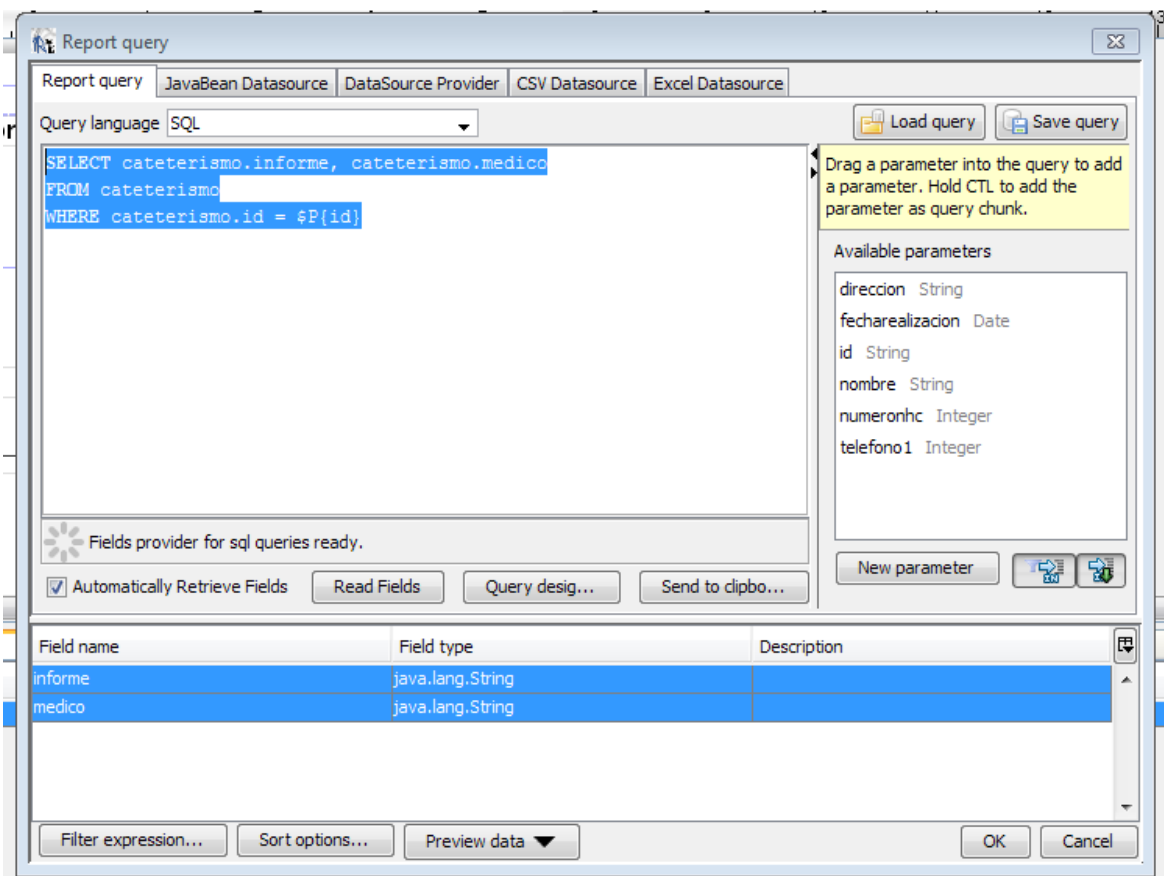


Captura 119: Captura de pantalla de herramienta *Ireport* para crear conexión con la BB.DD.

5.1.11.3.2 Crear los campos.

En Ireport, los campos se refieren a los tipos de datos traídos desde la BB.DD.

Como ya tenemos el id de cateterismo recogido anteriormente desde la fila de la colección Cateterismo, al crear una conexión a la BB.DD desde *Ireport*, con el siguiente SQL seleccionamos la tabla y los elementos del informe indicado:





Captura 120: Código SQL incluido con *Ireport* para acceder a las columnas de la tabla deseada de la BB.DD.

Finalmente vemos como tenemos nuestra plantilla la cual recoge los parámetros y campos explicados, y de manera gráfica podemos situarlos donde queramos:

$\$P\{\}$ parámetros

$\$F\{\}$ campos (fields)

 Servicio Canario de la Salud HOSPITAL UNIVERSITARIO GRAN CANARIA DR. NEGRÍN ESPAÑA	Informe de Cateterismo Servicio de Cardiología	 Gobierno de Canarias
Nombre: \${P{nombre}} NHC: \${P{numerohnc}} Teléfono: \${P{telefono1}}	Dirección: \${P{direccion}} Id. Informe: \${P{id}} Fecha Prueba: \${P{fecharealizacion}}	
Valoración: \${F{informe}}		

Captura 121: Captura de pantalla de la plantilla *ejemploReport2.jrxml*.

Si en nuestra aplicación accedemos al usuario de prueba, y obtenemos el primer informe de prueba cateterismo, tendremos el siguiente documento PDF, incluso disponible para descargar:



 Servicio Canario de la Salud HOSPITAL UNIVERSITARIO GRAN CANARIA DR. NEGRÍN ESPAÑA	Informe de Cateterismo Servicio de Cardiología	 Gobierno de Canarias
Nombre: Yaiza Santana NHC: 2 Teléfono: [Redacted]	Dirección: [Redacted] Id. Informe: 4028804353f0a6ea0153f0ac6c6e00 Fecha Prueba: 15/04/2014 00.00.00	
Valoración: otro cambio para comprobar que no cambia 15/09		
Firma: Celestina Amador Gil		

Captura 122: Captura de pantalla del informe médico visualizado a través del navegador web Internet Explorer.

5.2 Persistencia de los datos de paciente de cardiología.

Para poder implementar la persistencia de los datos de paciente cardiología, además de realizar las configuraciones de *Hibernate* en el bloque anterior, necesitamos crear las siguientes tablas en la BB. DD. con el gestor *PgAdmin III*. Las tablas y sus atributos se describen a continuación:

<i>alergianueva</i>			
Campo	Tipo (Data type)	Características	Referencias
nuevaalergia	character varying	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
id	character varying	Identificador UUID	
		Primary key	
		Not NULL	

Tabla 13: Tabla *alergianueva* e información relevante de sus atributos.

<i>antecedente</i>			
Campo	Tipo (Data type)	Características	Referencias
tipoantecedente	character varying	Puede ser nulo	
comentarioantecedente	character varying	Puede ser nulo	
fechaantecedente	date	Not NULL	
numeronhc	integer	Foreign key	paciente(numeronhc)
id	character varying	Not NULL	
		Primary key	
		Identificador UUID	

Tabla 14: Tabla *antecedente* e información relevante de sus atributos.

<i>cateterismo</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Primary key	
		Identificador UUID	
fecharealizacion	date	Not NULL	
informe	character varying	Foreign key	paciente(numeronhc)
medico	character varying	Puede ser nulo	
medico2	character varying	Puede ser nulo	

Tabla 15: Tabla *cateterismo* e información relevante de sus atributos.

<i>comentario</i>		
Campo	Tipo (Data type)	Características
comentariopaciente	character varying	Puede ser nulo
id	character varying	Not NULL
		Primary key
		Identificador UUID

Tabla 16: Tabla *comentario* e información relevante de sus atributos.

<i>complicacionimplante</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
tipocomplicacion	character varying	Puede ser nulo	
idgenerador	character varying	Foreign key	generador(id)

Tabla 17: Tabla *complicacionimplante* e información relevante de sus atributos.

<i>diagnostico</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Primary key	
		Identificador UUID	
modotaqui	character varying	Puede ser nulo	
situacionbateria	character varying	Puede ser nulo	
voltajebateria	numeric	Puede ser nulo	
indicadorbateria	integer	Puede ser nulo	
presencial	character varying	Puede ser nulo	
amplitudmvdv	integer	Puede ser nulo	
umbralvd	integer	Puede ser nulo	
impedancia	integer	Puede ser nulo	
estimulacionvd	integer	Puede ser nulo	
estadoelectrodovd	character varying	Puede ser nulo	
comentario	character varying	Puede ser nulo	
tratamientoyrecomendaciones	character varying	Puede ser nulo	
tipoagenda	character varying	Puede ser nulo	
idevento	character varying	Foreign key	eventomedico(id)
idpb	character varying	Foreign key	programacionbradicardia(id)
idpt	character varying	Foreign key	programaciontaquicardia(id)
firma	character varying	Puede ser nulo	
noacudio	boolean	Puede ser nulo	

Tabla 18: Tabla *diagnostico* e información relevante de sus atributos.

ecocardiografia			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fecharealizacion	date	Not NULL	
informe	character varying	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
medico	character varying	Puede ser nulo	

Tabla 19: Tabla *ecocardiografia* e información relevante de sus atributos.

ecocardiografiatranstoracica			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fecharealizacion	date	not NULL	
dtdvi	integer	Puede ser nulo	
teicholtz	integer	Puede ser nulo	
pp	integer	Puede ser nulo	
gmmitral	integer	Puede ser nulo	
ai	integer	Puede ser nulo	
gmaortico	integer	Puede ser nulo	
etiv	integer	Puede ser nulo	
onda	integer	Puede ser nulo	
dtsvi	integer	Puede ser nulo	
tiv	integer	Puede ser nulo	
dtdvd	integer	Puede ser nulo	
raizao	integer	Puede ser nulo	
gpaortico	integer	Puede ser nulo	
paps	integer	Puede ser nulo	
onda	integer	Puede ser nulo	
fe	integer	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
medico	character varying	Puede ser nulo	

Tabla 20: Tabla *ecocardiografiatranstoracica* e información relevante de sus atributos.

<i>electrodo</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
modelo	character varying	Puede ser nulo	
numeroserie	character varying	Puede ser nulo	
fabricante	character varying	Puede ser nulo	
fechaimplante	date	not NULL	
fechaexplante	date	Puede ser nulo	
camara	character varying	not NULL	
causaexplante	character varying	Puede ser nulo	
deflexionintrinsicode	integer	Puede ser nulo	
amplitud	integer	Puede ser nulo	
impedancia	integer	Puede ser nulo	
numeronhc	integer	foreign key	paciente(numeronhc)

Tabla 21: Tabla *electrodo* e información relevante de sus atributos.

<i>ergometria</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fecharealización	date	not NULL	
protocolo	character varying	Puede ser nulo	
ritmobasal	character varying	Puede ser nulo	
fcbasal	integer	Puede ser nulo	
fcmáxima	integer	Puede ser nulo	
porcentaje	integer	Puede ser nulo	
tabasal	integer	Puede ser nulo	
tamáxima	integer	Puede ser nulo	
tem	integer	Puede ser nulo	
tes	integer	Puede ser nulo	
mets	integer	Puede ser nulo	
resultado	character varying	Puede ser nulo	
notas	character varying	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
medico	character varying	Puede ser nulo	

Tabla 22: Tabla *ergometria* e información relevante de sus atributos.

<i>estudioelectrofisiologico</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fecharealización	date	not NULL	
informe	character varying	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
medico	character varying	Puede ser nulo	

Tabla 23: Tabla *estudioelectrofisiologico* e información relevante de sus atributos.

<i>eventomedico</i>		
Campo	Tipo (Data type)	Características
id	character varying	Not NULL
		Identificador UUID
		Primary key
fibrilacionauricular	character varying	Puede ser nulo
taquicardiaventricular	character varying	Puede ser nulo
otrasarritmias	character varying	Puede ser nulo
descargas	character varying	Puede ser nulo
comentariodescargas	character varying	Puede ser nulo
atp	character varying	Puede ser nulo
comentarioatp	character varying	Puede ser nulo
reiniciocontadores	character varying	Puede ser nulo
episodiosfv	character varying	Puede ser nulo
episodiosv	character varying	Puede ser nulo
episodiosv1	character varying	Puede ser nulo
episodiosnvs	character varying	Puede ser nulo
cambiosrta	character varying	Puede ser nulo
episodiosfa	character varying	Puede ser nulo
episodiossv	character varying	Puede ser nulo

Tabla 24: Tabla *eventomedico* e información relevante de sus atributos.

<i>generador</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
modelo	character varying	Puede ser nulo	
numeroserie	character varying	Puede ser nulo	
fabricante	character varying	Puede ser nulo	
fechaimplante	date	not NULL	
fechaexplante	date	Puede ser nulo	
regionimplante	character varying	Puede ser nulo	
lugar	character varying	Puede ser nulo	
acceso	character varying	Puede ser nulo	
tipogenerador	character varying	Puede ser nulo	
resincro	character varying	Puede ser nulo	
medico	character varying	Puede ser nulo	
medicoayudante	character varying	Puede ser nulo	
enfermero	character varying	Puede ser nulo	
causaexplante	character varying	Puede ser nulo	
comentariogenerador	character varying	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
idtc	character varying	Foreign key	tipocomplicacion(id)

Tabla 25: Tabla *generador* e información relevante de sus atributos.

<i>hipertensionpulmonar</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fecharealizacion	date	not NULL	
ai	integer	Puede ser nulo	
dtdvi	integer	Puede ser nulo	
tiv	integer	Puede ser nulo	
etiv	integer	Puede ser nulo	
teicholtz	integer	Puede ser nulo	
frecuenciacardiaca	integer	Puede ser nulo	
t1	integer	Puede ser nulo	
areaad	integer	Puede ser nulo	
areavd	integer	Puede ser nulo	
tapse	integer	Puede ser nulo	
gp	integer	Puede ser nulo	
dpgt	integer	Puede ser nulo	
g1	integer	Puede ser nulo	
g2	integer	Puede ser nulo	
raiza0	integer	Puede ser nulo	
dtsvi	integer	Puede ser nulo	
pp	integer	Puede ser nulo	
dtdvd	integer	Puede ser nulo	
arteriapulmonar	integer	Puede ser nulo	
tensionarterial	integer	Puede ser nulo	
t2	integer	Puede ser nulo	
d1	integer	Puede ser nulo	
d2	integer	Puede ser nulo	
indiceexcentrvi	integer	Puede ser nulo	
vele	integer	Puede ser nulo	
vela	integer	Puede ser nulo	
gm	integer	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
medico	character varying	Puede ser nulo	
gp2	integer	Puede ser nulo	

Tabla 26: Tabla *hipertensionpulmonar* e información relevante de sus atributos.

<i>holter</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fecharealizacion	date	not NULL	
informe	character varying	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
medico	character varying	Puede ser nulo	

Tabla 27: Tabla *holter* e información relevante de sus atributos.

<i>indicacion</i>			
Campo	Tipo (Data type)	Características	Referencias
fechaindicacion	date	not NULL	
tipindicacion	character varying	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
comentarioindicacion	character varying	Puede ser nulo	

Tabla 28: Tabla *indicacion* e información relevante de sus atributos.

<i>ingreso</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fechaingreso	date	not NULL	
servicio	character varying	Puede ser nulo	
comentario	character varying	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)

Tabla 29: Tabla *ingreso* e información relevante de sus atributos.

<i>lopd</i>			
Campo	Tipo (Data type)	Características	
id	character varying	Not NULL	Primary key
accion	character varying	Not NULL	
usuario	character varying	Not NULL	
paciente	integer	Not NULL	

Tabla 30: Tabla *lopd* e información relevante de sus atributos.

<i>paciente</i>			
Campo	Tipo (Data type)	Características	Referencias
numeronhc	integer	Not NULL	
		Identificador único	
		Primary key	
nombre	character varying	Not NULL	
telefono1	integer	not NULL	
telefono2	integer	not NULL	
direccion	character varying	not NULL	
id	character varying	Foreign key	comentario(id)
peso	numeric	not NULL	
estatura	numeric	not NULL	
bsa	numeric	not NULL	
imc	numeric	not NULL	
sobrepeso	character varying	not NULL	
mcppevio	character varying	Puede ser nulo	
mcpdependiente	character varying	Puede ser nulo	
diaprevio	character varying	Puede ser nulo	
terapiasprevias	character varying	Puede ser nulo	
causaexituscv	character varying	Puede ser nulo	
causaexitus	character varying	Puede ser nulo	
respondedor	character varying	Puede ser nulo	
remodelador	character varying	Puede ser nulo	
fechadeexitus	date	Puede ser nulo	
vms	character varying	Puede ser nulo	
ritmo	character varying	Puede ser nulo	
cicatriz	character varying	Puede ser nulo	
localizacion	character varying	Puede ser nulo	
qrs	character varying	Puede ser nulo	
comentariooutcomes	character varying	Puede ser nulo	
fechanacimiento	date	not NULL	
sexo	character varying	not NULL	
idpa	character varying	Foreign key	problemasactivos(id)

Tabla 31: Tabla *paciente* e información relevante de sus atributos.

<i>problema</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fechacomienzo	date	not NULL	
fecharesolucion	date	Puede ser nulo	
tipoproblema	character varying	Puede ser nulo	
gradoproblema	character varying	Puede ser nulo	
comentarioproblema	character varying	Puede ser nulo	
idgenerador	character varying	Foreign key	generador(id)
nhc	integer	Puede ser nulo	

Tabla 32: Tabla *problema* e información relevante de sus atributos.

La siguiente tabla no contendrá datos, ya que corresponde con una colección calculada transitoria (consultas de registros de la BB.DD. De cardiología) de problemas activos de generadores de la Sección sistemas, problemas activos. La tuvimos que crear ya que Openxava la requiere para poder visualizar estos datos.

<i>problemasactivos</i>		
Campo	Tipo (Data type)	Características
id	character varying	Not NULL
		Identificador UUID
		Primary key
fechacomienzo	date	not NULL
fecharesolucion	date	null
tipoproblema	character varying	Puede ser nulo
gradoproblema	character varying	Puede ser nulo
comentarioproblema	character varying	Puede ser nulo
nhc	integer	paciente(numeronhc)
idgenerador	character varying	generador(id)

Tabla 33: Tabla *problemasactivos* e información relevante de sus atributos.

<i>procedimiento</i>			
Campo	Tipo (Data type)	Características	Referencias
fechaprocedimiento	date	not NULL	
tipoprocedimiento	character varying	Puede ser null	
diagnosticoprocedimiento	character varying	Puede ser null	
comentarioprocedimiento	character varying	Puede ser null	
numeronhc	integer	Foreign key	paciente(numeronhc)
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	

Tabla 34: Tabla *procedimiento* e información relevante de sus atributos.

<i>programacionbradicardia</i>		
Campo	Tipo (Data type)	Características
id	character varying	Not NULL
		Identificador UUID
		Primary key
modoestimulacion	character varying	Puede ser null
lift	integer	Puede ser null
lms	integer	Puede ser null
lsf	character varying	Puede ser null
tipoestimulacionventricular	character varying	Puede ser null
retrasoventriculos	integer	Puede ser null
intervaloavestimulacion	integer	Puede ser null
intervaloavdeteccion	integer	Puede ser null
ciclosbusquedaav	integer	Puede ser null
histeresisav	integer	Puede ser null
sensibilidadvd	integer	Puede ser null
salidavd	integer	Puede ser null
configuracionestideteccionvd	integer	Puede ser null

Tabla 35: Tabla *programacionbradicardia* e información relevante de sus atributos.

<i>programaciontaquicardia</i>		
Campo	Tipo (Data type)	Características
id	character varying	Not NULL
		Identificador UUID
		Primary key
zonatv1	integer	Puede ser nulo
atptv1	character varying	Puede ser nulo
numrafagasatp1tv1	integer	Puede ser nulo
atp2tv1	integer	Puede ser nulo
numrafagasatp2tv1	integer	Puede ser nulo
descargatv1	integer	Puede ser nulo
descarga2tv1	integer	Puede ser nulo
maxenergiatv1	integer	Puede ser nulo
prorrogatv1	integer	Puede ser nulo
zonatv	integer	Puede ser nulo
atptv	character varying	Puede ser nulo
numrafagasatp1tv	integer	Puede ser nulo
atp2tv	integer	Puede ser nulo
numrafagasatp2tv	integer	Puede ser nulo
descargatv	integer	Puede ser nulo
descarga2tv	integer	Puede ser nulo
maxenergiatv	integer	Puede ser nulo
prorrogatv	integer	Puede ser nulo
zonafv	integer	Puede ser nulo
descargafv	integer	Puede ser nulo
maxenergiafv	integer	Puede ser nulo
prorrogafv	integer	Puede ser nulo
modotaqui	character varying	Puede ser nulo
comentarioprogramaciontaquicardia	character varying	Puede ser nulo

Tabla 36: Tabla *programaciontaquicardia* e información relevante de sus atributos.

<i>resincronizacioncardiaca</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fecharealizacion	date	not NULL	
rimm	character varying	Puede ser nulo	
ritpp	character varying	Puede ser nulo	
fatcva	character varying	Puede ser nulo	
asmas	character varying	Puede ser nulo	
imp	character varying	Puede ser nulo	
img2	character varying	Puede ser nulo	
mmsdc	character varying	Puede ser nulo	
dptd4c	character varying	Puede ser nulo	
numeronhc	integer	Foreign key	paciente(numeronhc)
tpa	integer	Puede ser nulo	
rsp	integer	Puede ser nulo	
tdrrr	integer	Puede ser nulo	
medico	character varying	Puede ser nulo	

Tabla 37: Tabla *resincronizacioncardiaca* e información relevante de sus atributos.

<i>tipocomplicacion</i>		
Campo	Tipo (Data type)	Características
id	character varying	Not NULL
		Identificador UUID
		Primary key
derramepericardico	boolean	Puede ser nulo
disecioncoronaria	boolean	Puede ser nulo
estimulacionfrenica	boolean	Puede ser nulo
hematoma	boolean	Puede ser nulo
hemotorax	boolean	Puede ser nulo
infeccion	boolean	Puede ser nulo
insuficienciarenal	boolean	Puede ser nulo
insuficienciacardiaca	boolean	Puede ser nulo
muerte	boolean	Puede ser nulo
neumotorax	boolean	Puede ser nulo
perforacioncoronaria	boolean	Puede ser nulo
perforacionsenocoronario	boolean	Puede ser nulo
sepsis	boolean	Puede ser nulo
shock	boolean	Puede ser nulo
taponamiento	boolean	Puede ser nulo
trombosissubclavia	boolean	Puede ser nulo

Tabla 38: Tabla *tipocomplicacion* e información relevante de sus atributos.

<i>visita</i>			
Campo	Tipo (Data type)	Características	Referencias
id	character varying	Not NULL	
		Identificador UUID	
		Primary key	
fechavisita	date	not NULL	
tipovisita	character varying	Puede ser nulo	
iddiagnostico	character varying	Foreign key	diagnostico(id)
numeronhc	integer	Foreign key	paciente(numeronhc)

Tabla 39: Tabla *visita* e información relevante de sus atributos.

5.3 Cabecera de la página de información de paciente.

Para atender a los requerimientos de la aplicación de disponer siempre de los datos demográficos y el estado nutricional del paciente visualizado, se generó la siguiente cabecera de página, donde el usuario puede leer los datos demográficos recogidos de la BB.DD de Drago, y leer, modificar las variables peso y estatura que se guardan en la BB.DD de cardiología.

Para el desarrollo de esta memoria se verán los ejemplos accediendo a los datos de paciente de la BB.DD de pruebas para proteger la información de acuerdo a la LEY ORGÁNICA 15/1999 de 13 de Diciembre de protección de datos de Carácter Personal [16]. Al final de este apartado, veremos el código que se generó para agregar la fuente de datos para la BB.DD de Drago y ver los datos de paciente reales.

Captura 123: Captura de pantalla de la cabecera de la página de paciente.

Para definir la disposición de los atributos de paciente en la vista, lo conseguimos gracias a la anotación `@View` en la clase `paciente.java`:

```
8  @Entity
9  @View(members =
10 "DatosDemograficos [" +
11 " numeroNhc, nombre;" +
12 " fechaNacimiento, sexo;" +
13 " telefono1, telefono2;" +
14 " direccion;" +
15 "]" +
16 "EstadoNutricional [" +
17 " peso;" +
18 " estatura;" +
19 " bsa, imc;" +
20 " sobrepeso;" +
21 "]" +
```

Captura 124: Captura de pantalla de fragmento de código de la clase `Paciente.java`.

donde los elementos que están separados por comas van en la misma línea, y los que van separados por punto y coma, van en una nueva línea. Los corchetes se traducen visualmente a un cuadro.

Los declaración de los atributos que necesitamos para esta cabecera se añaden a continuación, donde se explica con comentarios en color verde, qué hace el código:

```

61 @Table(name = "paciente") //se referencia a la tabla de la BB.DD. llamada paciente
62 public class Paciente {
63
64     //PROPIEDADES
65     @Id // La propiedad numero es la clave. Las claves son obligatorias (required) por defecto
66     @Column(name="NUMERONHC", length=6) // La longitud de columna se usa a nivel UI y a nivel DB
67     public int numeroNhc;
68
69     @Column(name="NOMBRE")
70     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
71     private String nombre;
72
73     @Column(name="FECHANACIMIENTO")
74     @Stereotype("FECHAHORA")
75     private Date fechaNacimiento;
76
77     @Column (name="SEXO")
78     private Sexo sexo;
79     public enum Sexo { //se describe un tipo enumerado que se visualiza como un SELECT
80         HOMBRE("Hombre"), MUJER("Mujer");
81
82         private String nombreSexo;
83
84         private Sexo (String nombreSexo){
85             this.nombreSexo = nombreSexo;
86         }
87
88         public String getNombreSexo() {
89             return nombreSexo;
90         }
91     }
92
93     //añadir edad; desde Base de Datos del Negrín
94     //añadir sexo; desde Base de Datos del Negrín
95     //añadir fecha de nacimiento; desde Base de Datos del Negrín
96
97     @Column(name="TELEFONO1")
98     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
99     private int telefono1;
100
101     @Column(name="TELEFONO2")
102     private int telefono2;
103
104     @Column(name="DIRECCION")
105     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
106     private String direccion;
107
108     @Stereotype("PESO") //Anotación Openxava para añadir unidad de medida Kg.
109     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
110     @Column(name="PESO", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
111     private float peso;
112
113     @Stereotype("METROS") //Anotación Openxava para añadir unidad de medida metros.
114     @Required // Se mostrará un error de validación si la propiedad nombre se deja en blanco
115     @Column(name="ESTATURA", length=5) // La longitud de columna se usa a nivel UI y a nivel DB
116     private float estatura;
117
118     @Column(name="BSA", length=5) // La longitud de columna se usa a nivel UI y a nivel DB
119     @org.hibernate.annotations.Formula("(peso*estatura)/3600") //operación que calcula BSA
120     public float bsa;
121
122     public float bsa;
123
124     @Column(name="IMC", length=5) // La longitud de columna se usa a nivel UI y a nivel DB
125     @org.hibernate.annotations.Formula("peso/(estatura*estatura)") //operación que calcula BSA
126     public float imc;
127
128     @Column(name="SOBREPESO", length=30)
129     @org.hibernate.annotations.Formula("imc") //se pasa la variable imc para calcular el sobrepeso
130     //del paciente según la OMS
131     private String sobrepeso;

```

Capturas 125: Capturas de pantalla de fragmento de código de la clase *Paciente.java*.

Los métodos de estos atributos se describen así:

```
481 //MÉTODOS
482 public int getNumeroNhc() {
483     return numeroNhc;
484 }
485
486 public void setNumeroNhc(int numeroNhc) {
487     this.numeroNhc = numeroNhc;
488 }
489
490 public String getNombre() {
491     return nombre;
492 }
493
494 public void setNombre(String nombre) {
495     this.nombre = nombre;
496 }
497
498 public Date getFechaNacimiento() {
499     return fechaNacimiento;
500 }
501
502 public void setFechaNacimiento(Date fechaNacimiento) {
503     this.fechaNacimiento = fechaNacimiento;
504 }
505
506 public Sexo getSexo() {
507     return sexo;
508 }
509
510 public void setSexo(Sexo sexo) {
511     this.sexo = sexo;
512 }
513
514 public int getTelefono1() {
515     return telefono1;
516 }
517
518 public void setTelefono1(int telefono1) {
519     this.telefono1 = telefono1;
520 }
521
522 public int getTelefono2() {
523     return telefono2;
524 }
525
526 public void setTelefono2(int telefono2) {
527     this.telefono2 = telefono2;
528 }
529
530 public String getDireccion() {
531     return direccion;
532 }
533
534 public void setDireccion(String direccion) {
535     this.direccion = direccion;
536 }
```

Capturas 126: Capturas de pantalla de fragmento de código de la clase *Paciente.java*.

```

666 public float getPeso() {
667     return peso;
668 }
669
670 public void setPeso(float peso) {
671     this.peso = peso;
672 }
673
674 public float getEstatura() {
675     return estatura;
676 }
677
678 public void setEstatura(float estatura) {
679     this.estatura = estatura;
680 }
681
682 //cálculo del área de superficie corporal
683 public float getBsa() {
684     return bsa;
685 }
686
687
688 //cálculo del Índice de Masa Corporal
689 public float getImc() {
690     return imc;
691 }
692

```

Captura 127: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

Vemos que no existen los métodos *set* para BSA, IMC y Sobrepeso ya que son sólo lectura.

```

693
694 //cálculo de sobrepeso según la OMS
695 public String getSobrepeso() {
696     if (imc<16.00) {
697         sobrepeso="Delgadez Severa";
698     } else if (imc>=16.00 && imc<=16.99){
699         sobrepeso="Delgadez Moderada";
700     } else if (imc>=17.00 && imc<=18.49){
701         sobrepeso="Delgadez Aceptable";
702     } else if (imc>=18.50 && imc<=24.99){
703         sobrepeso="Peso Normal";
704     }else if (imc>=25.00 && imc<=29.99){
705         sobrepeso="Sobrepeso";
706     }else if (imc>=30.00 && imc<=34.99){
707         sobrepeso="Obeso: Tipo I";
708     }else if (imc>=35.00 && imc<=40.00){
709         sobrepeso="Obeso: Tipo II";
710     }else {
711         sobrepeso="Obeso: Tipo III";
712     }
713     return sobrepeso;
714 }
715

```

Captura 128: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

Como ya se explicó con un ejemplo cómo dar formato a los datos en las vistas a través de los estereotipos, se añade el código para este apartado. En este caso, damos formato al campo peso y estatura:

- Fichero *editores.xml*:

```
1 <?xml version = "1.0" encoding = "ISO-8859-1"?>
2
3 <!DOCTYPE editores SYSTEM "dtds/editores.dtd">
4
5 <editores>
6
7   <editor nombre="peso" url="pesoEditor.jsp">
8     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
9     <para-estereotipo estereotipo="PESO"/>
10  </editor>
11
12  <editor nombre="metros" url="metrosEditor.jsp">
13    <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
14    <para-estereotipo estereotipo="METROS"/>
15  </editor>
16
```

Captura 129: Captura de pantalla de fragmento de código del fichero *editores.xml*.

- Clase formateadora (sólo se ha de crear una sólo vez):

```
1 package org.openxava.cardiologia.formateadores;
2
3 import java.math.*;
4
5 public class PesoFormateador implements IFormatter {
6
7     public String format(HttpServletRequest request, Object object) throws Exception {
8         if (Is.empty(object)) return "";
9         return getFormat().format(object);
10    }
11
12    public Object parse(HttpServletRequest request, String string) throws Exception {
13        if (Is.emptyString(string)) return null;
14        string = Strings.change(string, " ", ""); // In order to work with Polish
15        return new BigDecimal(getFormat().parse(string).toString()).setScale(2);
16    }
17
18    private NumberFormat getFormat() {
19        NumberFormat f = DecimalFormat.getNumberInstance(Locales.getCurrent());
20        f.setMinimumFractionDigits(2);
21        f.setMaximumFractionDigits(2);
22        return f;
23    }
24 }
25
26
27
28
29
30
```

Captura 130: Captura de pantalla del código de la clase *PesoFormateador.java*.

- Scripts de los editores:

```

1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7 symbol = "metros"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 symbol = "?";
11 }
12 %>
13
14 <%=symbol%>

```

Captura 131: Captura de pantalla del script *metrosEditor.jsp*.

```

1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7 symbol = "Kg"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 symbol = "?";
11 }
12 %>
13
14 <%=symbol%>

```

Captura 132: Captura de pantalla del script *pesoEditor.jsp*.

Como vemos a continuación, por defecto *Openxava* crea las acciones CRUD para la clase paciente, por lo tanto permite al usuario crear, leer, actualizar, borrar o incluso buscar otro paciente a través de los botones que se indican en la siguiente captura. También permite la paginación entre pacientes en el segundo y tercer botón de los pacientes existentes en la base de datos.

Captura 133: Menú principal con acciones CRUD y paginado de pacientes.

Si se desea eliminar la acción borrar paciente, se podrá modificar como veremos más adelante, modificando código.

5.3.1 Visualización de los datos de paciente desde la BB.DD de Drago.

Para poder visualizar los datos de paciente que figuran en la BB.DD. de la sanidad pública, debemos añadir el fichero con extensión jar *ojdbc14.jar* [104] en los directorios:

1. *..openxava-5.4/openxava-5.4/tomcat/lib/*
2. *..openxava-5.4/openxava-5.4/workspace/Cardiologia/web/WEB-INF/lib*

A continuación se ofrece sólo un fragmento de código por confidencialidad de la clase *XXX.java* la cual accede a la BB.DD de Drago, y trae los datos requeridos:

```
Connection con = null;
Context ctx = new InitialContext();
String recurso="drago/";
javax.sql.DataSource dataSource = (javax.sql.DataSource) ctx.lookup(con);
```

Una vez hecho esto, se debe añadir la fuente de datos de la conexión a Drago en *context.xml* la cual no se añade por protección de la información.

Al pasar al entorno de producción donde las variables serán sólo de lectura, sólo se debe añadir la anotación **@ReadOnly** a los anteriores atributos que aparecían en modo escritura (nombre, fecha de nacimiento, sexo, contacto, otro contacto y dirección).

5.4 Navegación de la aplicación.

La estructura de navegación de la aplicación web de cardiología requiere que cada usuario autenticado, una vez dentro de la aplicación, pueda ver a todos los pacientes de la BB.DD, y pueda realizar búsquedas de pacientes filtrando por los siguientes atributos de paciente: por número de historia clínica, por nombre, por teléfonos de contacto, por dirección postal, por peso, por estatura, por BSA, por IMC, o por sobrepeso. Una vez encontrado el paciente deseado, se permite acceder a toda la información del paciente elegido.

Openxava por defecto, permite que el usuario añada o elimine columnas como filtros para las búsquedas de las colecciones. Debemos modificar esto para que los filtros de

búsquedas de paciente sean siempre los requeridos. Una vez ajustados los filtros como solicitan los requisitos, conseguimos quitar la opción de editar columnas por el usuario dentro del proyecto *Cardiología*, en la ruta `openxava-5.4\openxava-5.4\workspace\Cardiología\properties\xava.properties`, añadiendo:

customizeList=false

Por lo tanto, al acceder a la aplicación el usuario verá la siguiente página:

KHArдиоLOgía - Paciente ☆

Nuevo Generar PDF Generar Excel Mis informes

Detalle Lista Ambos

Número de Historia	Nombre	Contacto	Otro contacto	Dirección	Peso	Estatura	BSA	IMC	Sobrepeso
4	Maria Jesús Santana Santar	928000000	0	calle la nube	94,00	1,59	0,04	37,18	
5	Prueba	928000000	0	calle fulanito	89,00	1,90	0,05	24,65	
1	Mónica Santana	928000000	0	calle pascal	100,00	1,70	0,05	34,6	
15	prueba	65	655	calle	20,00	2,00	0,01	5	
2	Yaiza Santana	928000000	887799	LEO	62,00	1,71	0,03	21,2	
3	Oscar Vega	928000000	620262427	calle Piscis	90,00	2,00	0,05	22,5	
Σ		Σ	Σ		Σ	Σ	Σ	Σ	

Captura 134: Captura de pantalla de la búsqueda de pacientes de la aplicación web de cardiología.

Como ya anteriormente ocultamos la navegación entre módulos, veremos esta información haciendo que la página de acceso a la aplicación sea el enlace del módulo paciente:

http://URL_HUGCDN/Cardiología/m/Paciente

La aplicación de cardiología además, requiere que la información relacionada se agrupe para mejorar la usabilidad de la aplicación, y la navegación en las diferentes funcionalidades de la aplicación. Es por esto que se agrupó de manera lógica y alfabéticamente todos los datos de paciente en una misma página, representados en color verde en el siguiente gráfico:

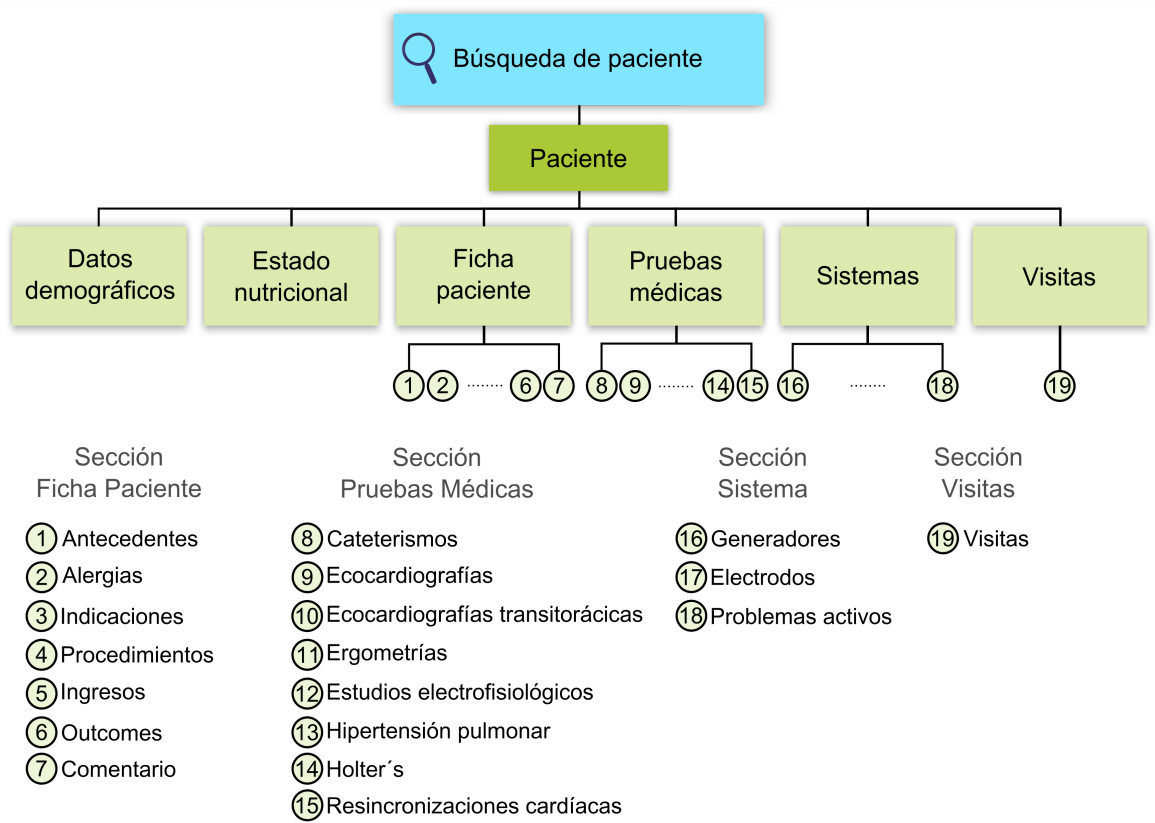


Figura 55: Estructura de navegación de la aplicación web.

A continuación, vemos como existen 4 secciones, cada una de ellas con sus apartados:

Datos demográficos

Número de Historia: 1 Nombre: Mónica Santana
 Fecha de nacimiento: 15/10/1985 00:00 Sexo: Mujer
 Contacto: 928000000 Otro contacto: 0
 Dirección: calle pascal

Estado nutricional

Peso: 100,00 Kg
 Estatura: 1,70 metros
 BSA: 0,05 IMC: 34,6
 Sobrepeso: Obeso: Tipo I

Ficha de paciente | Pruebas médicas | Sistemas | Visitas

Antecedentes | Alergias | Indicaciones | Procedimientos | Ingresos | Outcomes | Comentario

Antecedentes

Fecha antecedente	Tipo antecedente	Comentario antecedente
02/12/2013 00:00	Exfumador	otro
03/03/2014 00:00	AIT	jdawudhwu
01/04/2016 00:00	Exbebedor	hoy deajo

Hay 3 registros en la lista

Esto lo conseguimos gracias a la anotación `@View` comentada antes, para distribuir la información de paciente como convenga:

```

1 package org.openxava.cardiologia.model;
2
3 import javax.persistence.*;
4 import org.openxava.annotations.*;
5 import org.openxava.jpa.*;
6 import java.util.*;
7
8 @Entity
9 @View(members =
10 "DatosDemograficos [" +
11 " numeroNhc, nombre;" +
12 " fechaNacimiento, sexo;" +
13 " telefono1, telefono2;" +
14 "direccion;" +
15 "]" +
16 "EstadoNutricional [" +
17 " peso;" +
18 " estatura;" +
19 " bsa, imc;" +
20 " sobrepeso;" +
21 "]" +
22 "FichaDePaciente {" +
23 "Antecedentes { antecedente }" +
24 "Alergias { alergiaNueva }" +
25 "Indicaciones { indicacion }" +
26 "Procedimientos { procedimiento }" +
27 "Ingresos { ingreso }" +
28 "Outcomes {" +
29 "ElectrocardiogramaBasal [" +
30 "ritmo, qrs, cicatriz, localizacion;" +
31 "];" +
32 "DispositivosPrevios [" +
33 "mcpPrevio, mcpDependiente, daiPrevio, terapiasPrevias;" +
34 "];" +
35 "Outcomes [" +
36 "fechaDeExitus, causaExitusCV, causaExitus, respondedor, remodelador, vvms;" +
37 "];" +
38 "ComentarioOutcomes [" +
39 "comentarioOutcomes;" +
40 "];" +
41 "}" +
42 "Comentario { comentario }" +
43 "]" +
44 "PruebasMedicas {" +
45 "Cateterismos { cateterismo }" +
46 "Ecocardiografías { ecocardiografia}" +
47 "EcocardiografiasTranstoracicas {ecocardiografiaTranstoracica}" +
48 "Ergometrías { ergometria}" +
49 "EstudiosElectrofisiologicos { estudioElectrofisiologico }" +
50 "HipertensionPulmonar { hipertensionPulmonar }" +
51 "Holter's { holter }" +
52 "ResincronizacionesCardiacas { resincronizacionCardiaca}" +
53 "]" +
54 "Sistemas {" +
55 "Generadores { generador }" +
56 "Electrodos { electrodo }" +
57 "ProblemasActivos { problemilla }" +
58 "]" +
59 "Visitas { visita }"
60 )

```

Capturas 135: Capturas de pantalla de fragmento de código de la clase *Paciente.java*.

Sólo nos queda mejorar las vistas con la internacionalización de las etiquetas, para que se muestre “Pruebas médicas” en lugar de “PruebasMedicas como vemos en la línea 44, por ejemplo. Seguidamente se muestran las etiquetas añadidas en el fichero *EtiquetasCardiologias_es.properties* para la clase *paciente.java*:

```
1 # Etiquetas para la aplicaci:;%n Cardiologia
2
3 #página de Inicio
4 Cardiologia=KHArдиоLOGía
5
6 # Componentes de Paciente
7 DatosDemograficos=Datos demográficos
8 EstadoNutricional=Estado nutricional
9 numeroNhc=Número de Historia
10 nombre=Nombre
11 HOMBRE=Hombre
12 MUJER=Mujer
13 telefono1=Contacto
14 telefono2=Otro contacto
15 direccion=Dirección
16 bsa=BSA
17 imc=IMC
18 FichaDePaciente=Ficha de paciente
19 ElectrocardiogramaBasal=Electrocardiograma basal
20 DispositivosPrevios=Dispositivos previos
21 ComentarioOutcomes=Comentario outcomes
22 PruebasMedicas=Pruebas médicas
23 EcocardiografiasTranstoracicas=Ecocardiografías transtorácicas
24 EstudiosElectrofisiologicos=Estudios electrofisiológicos
25 HipertensionPulmonar=Hipertensión pulmonar
26 ResincronizacionesCardiacas=Resincronizaciones cardíacas
27
```

Captura 136: Captura de pantalla de fragmento de código del fichero *EtiquetasCardiologias_es.properties*

5.5 Sección de ficha de paciente.

La aplicación web muestra información de interés del paciente para la especialidad que puede ayudar para el diagnóstico de cardiopatías, para la realización de pruebas médicas o algún protocolo médico. Esta información son los antecedentes, las alergias, las indicaciones, los procedimientos, los ingresos, outcomes y los comentarios de paciente definidos en el capítulo de requisitos.

Explicaremos cómo hemos logrado que estos datos figuren en la información de paciente.

5.5.1 Clase *Identificable.java*.

Para evitar repetir código en las clases que necesitan un id único, se creó la clase *Identificable.java*, para que éstas hereden la propiedad y método del identificador UUID.

El código de la clase es el siguiente:

The image shows a screenshot of an IDE with several tabs open. The active tab is 'identificable.java'. The code is as follows:

```
1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5 import org.hibernate.annotations.*;
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8
9 @MappedSuperclass
10 public class Identificable{
11
12     //PROPIEDADES
13     @Id @GeneratedValue(generator="system-uuid") @Hidden //se genera un id UUID
14     @GenericGenerator(name="system-uuid", strategy = "uuid")
15     @Column(name="ID") //el nombre de la columna que almacena el id en la BB.DD
16     private String id;
17
18     //MÉTODOS
19     public String getID() {
20         return id;
21     }
22
23     public void setID(String id) {
24         this.id = id;
25     }
26 }
27
```

Captura 137: Captura de pantalla del código de la clase *Identificable.java*.

5.5.2 Antecedentes.

El usuario, tal como se pide para la implementación de esta aplicación, puede visualizar el historial de antecedentes del paciente, los datos de cada uno de ellos, y también

modificarlos. La opción de eliminar el registro, veremos como deshabilitarla, ya que los datos se borrarán a través del gestor de la BB.DD de cardiología, y bajo solicitud de mantenimiento al departamento de informática del HUGCDN.

The screenshot shows a web application interface for patient records. At the top, there are tabs for 'Ficha de paciente', 'Pruebas médicas', 'Sistemas', and 'Visitas'. Below these, there are more tabs for 'Antecedentes', 'Alergias', 'Indicaciones', 'Procedimientos', 'Ingresos', 'Outcomes', and 'Comentario'. A toolbar contains buttons for '+ Añadir', 'Generar PDF', and 'Generar Excel'. Below the toolbar is a form for adding a new antecedent, with fields for 'Fecha antecedente', 'Tipo antecedente', and 'Comentario antecedente'. The 'Comentario antecedente' field has a dropdown menu with 'empieza por' selected. Below the form is a table of existing antecedents. The table has columns for 'Fecha antecedente', 'Tipo antecedente', and 'Comentario antecedente'. The rows are:

Fecha antecedente	Tipo antecedente	Comentario antecedente
11/05/2016 00:00	Asma Bronquial	
04/03/2013 00:00	AIT	comprobar cambios
09/03/2015 00:00	Otro	otro cambio
07/09/2016 00:00	AIT	comprobar uno nuevo
19/08/2016 00:00	Bebedor Activo	prueba

Annotations in the image include:

- 'Botón añadir' pointing to the '+ Añadir' button.
- 'Botón editar' pointing to the edit icon (pencil) in the first row of the table.
- 'Historial antecedente' pointing to the table of antecedents.

Captura 138: Captura de pantalla de la sección Ficha de paciente, apartado Antecedentes.

Para registrar un antecedente en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de un sencillo formulario se registran y modifican los datos.

Crea una nueva entidad - Antecedente

Fecha antecedente

Tipo antecedente

Comentario antecedente

Grabar Grabar y continuar Cerrar

Captura 139: Captura de pantalla del formulario crear antecedente.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón cerrar no guarda el objeto y cierra el formulario.

Editar - Antecedente

Fecha antecedente

Tipo antecedente

Comentario antecedente

Grabar Cerrar

Captura 140: Captura de pantalla del formulario editar antecedente.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD. El botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *Antecedente.java* que se ha escrito para este apartado, es el siguiente:

```

1 package org.openxava.cardiologia.model;
2
3 import javax.persistence.*;
4 import org.openxava.annotations.*;
5 import java.util.Date;
6
7 @Entity
8 @Table(name = "antecedente") //se referencia a la tabla de la BB.DD. llamada antecedente
9 public class Antecedente extends Identificable{ //hereda de la clase Identificable
10
11     //PROPIEDADES
12     @Stereotype("FECHAORA") //Anotación Openxava para añadir un calendario.
13     @Required // Se mostrará un error de validación si la propiedad se deja en blanco
14     private Date fechaAntecedente;
15
16     private TipoAntecedente tipoAntecedente;
17     public enum TipoAntecedente { //se describe un tipo enumerado que se visualiza como un SELECT
18         ACV("ACV"), AIT("AIT"), ASMABRONQUIAL("Asma Bronquial"), BEBEDORACTIVO ("Bebedor Activo"),
19         CIRROSISHEPATICA("Cirrosis Hepática"), CISQUEMICA("C. Isquémica"),
20         DM1 ("DM1"), DM2("DM2"), DLP("DLP"), EPOC("EPOC"), EXBEBEDOR ("Exbebedor"),
21
22         EXFUMADOR("Exfumador"), FIBRILACIONAURICULAR ("Fibrilación Auricular"),
23         FLUTTERAURICULAR("Flutter Auricular"), HBP("HBP"), HIPERURICEMIA("Hiperuricemia"),
24         HTA("HTA"), IRC("IRC"), NEOPLASIAS("NEOPLASIAS"), OTRASARRITMIAS("Otras Arritmias"),
25         OTRO("Otro"), PROTESISVALVULAR("Prótesis Valvular"), TABAQUISMOACTIVO("Tabaquismo Activo"),
26         TRASPLANTERENAL("Trasplante Renal");
27
28     private String nombreTipoAntecedente;
29     private TipoAntecedente (String nombreTipoAntecedente){
30         this.nombreTipoAntecedente = nombreTipoAntecedente;
31     }
32
33     public String getNombreTipoAntecedente() {
34         return nombreTipoAntecedente;
35     }
36
37 }
38
39 @Stereotype("MEMO") // Área de texto o equivalente
40 private String comentarioAntecedente;
41
42
43 //REFERENCIAS
44 @ManyToOne//Referencia que no permite Antecedente sin Paciente
45 @JoinColumn( //se define la clave foránea de la tabla
46     name="numeroNhc", //columna para la clave foránea
47     nullable = true,
48     foreignKey = @ForeignKey(name = "fk_antecedente_numeronhc"))
49     //fk_antecedente_numeronhc es la la clave foránea en la BB.DD de cardiologia
50     private Paciente antecedentesLocal; // Una referencia Java convencional
51
52 //MÉTODOS
53 public Date getFechaAntecedente() {
54     return fechaAntecedente;
55 }
56
57 public void setFechaAntecedente(Date fechaAntecedente) {
58     this.fechaAntecedente = fechaAntecedente;
59 }
60
61 public String getComentarioAntecedente() {
62     return comentarioAntecedente;
63 }
64
65 public void setComentarioAntecedente(String comentarioAntecedente) {
66     this.comentarioAntecedente = comentarioAntecedente;
67 }
68
69 public TipoAntecedente getTipoAntecedente() {
70     return tipoAntecedente;
71 }
72
73 public void setTipoAntecedente(TipoAntecedente tipoAntecedente) {
74     this.tipoAntecedente = tipoAntecedente;
75 }
76
77 //Métodos para el historial de antecedente
78 public Paciente getAntecedentesLocal() {
79     return antecedentesLocal;
80 }

```

```

81
82     public void setAntecedentesLocal(Paciente antecedentesLocal) {
83         this.antecedentesLocal = antecedentesLocal;
84     }
85     }
86 }
87

```

Capturas 141: Capturas de pantalla de la clase *Antecedente.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *Antecedente.java*. Lo indicamos con el siguiente código:

```

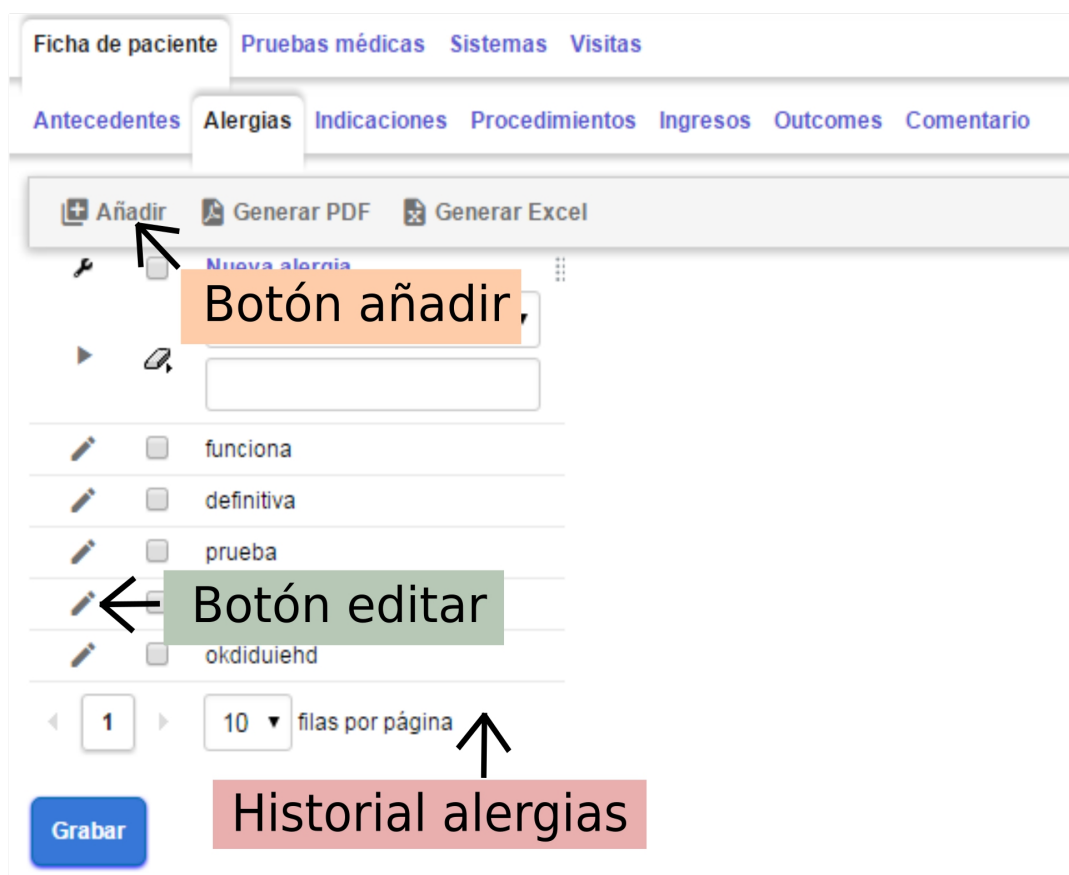
358     //se deshabilita la opción borrar desde el formulario del antecedente
359     @RemoveAction("")
360     //se deshabilita la opción borrar desde la lista de antecedentes
361     @RemoveSelectedAction("")
362     //si se borra paciente, se
363     //borrarán sus antecedentes
364     @OneToMany (mappedBy="antecedentesLocal", cascade=CascadeType.ALL)
365     //para poder visualizar
366     //la lista de antecedentes
367     private Collection<Antecedente> antecedente = new ArrayList<Antecedente>();
368

```

Captura 142: Captura de pantalla de la clase *Paciente.java*.

5.5.3 Alergias.

El usuario puede escribir la alergia de manera detallada, visualizar el historial de alergias del paciente, los datos de cada una, y también modificarlas. De ahora en adelante, como se hizo con los antecedentes, deshabilitaremos el borrado de los datos en modo lista y en modo detalle para las demás clases.

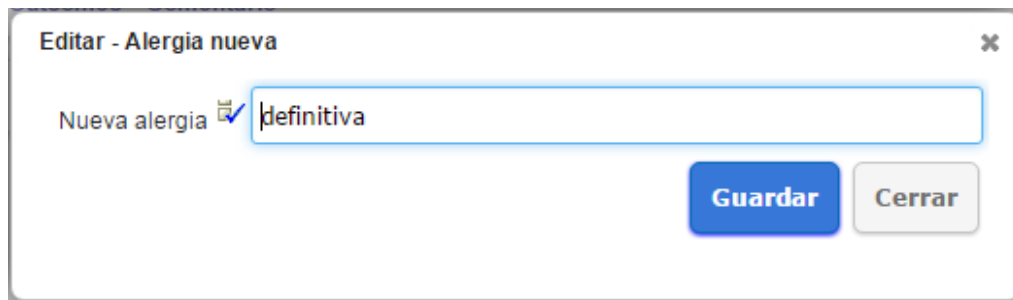


Captura 143: Captura de pantalla de la sección Ficha de paciente, apartado Alergias.

Para registrar una alergia en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran o modifican los datos.

Captura 144: Captura de pantalla del formulario crear alergia.

El botón Guardar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.



Captura 145: Captura de pantalla del formulario editar alergia.

El botón Guardar cierra el formulario al modificar el objeto en la BB.DD. El botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *AlergiaNueva.java* que se ha escrito para este apartado, es el siguiente:

```

alergianueva.java x
1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5 //librerías para usar las funciones Openxava
6 import org.openxava.annotations.*;
7 import org.openxava.util.*;
8
9 @Entity
10 @Table(name = "alergianueva") //se referencia a la tabla de la BB.DD. llamada alergianueva
11 public class AlergiaNueva extends Identificable { //hereda de la clase Identificable, para
12 //tener un identificador universal único, que se autogenera con funciones.
13
14 //PROPIEDADES
15 @Required // Se mostrará un error de validación si la propiedad se deja en blanco
16 @Column(name="NUEVAALERGIA")
17 private String nuevaAlergia;
18
19
20 //REFERENCIAS
21 @ManyToOne //Referencia que no permite Alergia sin Paciente
22 @JoinColumn( //se define la clave foránea de la tabla
23             name="numeroNhc", //columna para la clave foránea
24             nullable = true,
25             foreignKey = @ForeignKey(name = "fk_alergianueva_numeroNhc"))
26 //fk_alergianueva_numeroNhc es la la clave foránea en la BB.DD de cardiología
27 private Paciente alergiasNuevasLocal; // Una referencia Java convencional
28
29 //Métodos para el historial de alergias
30 public Paciente getAlergiasNuevasLocal() {
31     return alergiasNuevasLocal;
32 }
33
34 public void setAlergiasNuevasLocal(Paciente alergiasNuevasLocal) {
35     this.alergiasNuevasLocal = alergiasNuevasLocal;
36 }
37
38 //MÉTODOS
39 public void setNuevaAlergia(String nuevaAlergia) {
40     this.nuevaAlergia = nuevaAlergia;
41 }
42
43 public String getNuevaAlergia() {
44     return nuevaAlergia;
45 }
46
47 }
48

```

Capturas 146: Capturas de pantalla de la clase *AlergiaNueva.java*.

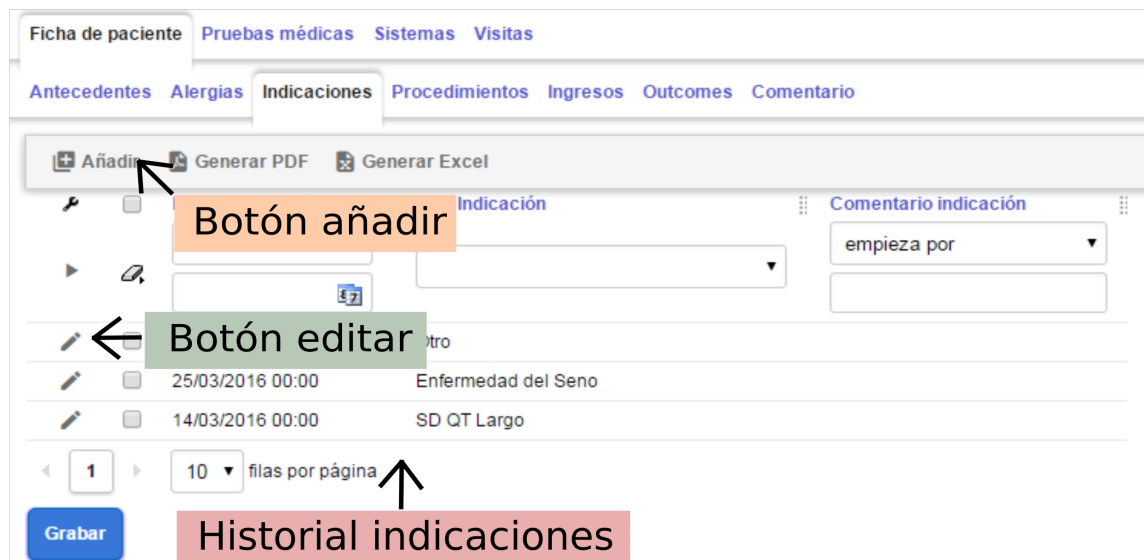
En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *AlergiaNueva.java*. Lo indicamos con el siguiente código:

```
347 //COLECCIONES
348 //cada colección, indicado con la anotación @OneToMany
349 //se referirá a una relación 1 a 0, 1 o muchos.
350
351 //se deshabilita la opción borrar desde el formulario de alergia
352 @RemoveAction("")
353 //se deshabilita la opción borrar desde la lista de alergias
354 @RemoveSelectedAction("")
355 //ejemplo de cómo auditar los datos en la BB.DD.
356 @SaveAction("lopd.Guardar")
357 //si se borra paciente, se
358 //borrarán sus alergias
359 @OneToMany (mappedBy="alergiasNuevasLocal", cascade=CascadeType.ALL)
360 //para poder visualizar
361 //la lista de alergias
362 private Collection<AlergiaNueva> alergiaNueva = new ArrayList<AlergiaNueva>();
```

Captura 147: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.5.4 Indicaciones.

El usuario puede escribir indicaciones a realizar por el paciente, visualizar el historial de indicaciones del paciente, los datos de cada una, y también modificarlas.



Captura 148: Captura de pantalla de la sección Ficha de paciente, apartado Indicaciones.

Para registrar una indicación en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran o modifican los datos.

Captura 149: Captura de pantalla del formulario crear indicación.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.

Captura 150: Captura de pantalla del formulario editar indicación.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *Indicacion.java* que se ha escrito para este apartado, es el siguiente:

```

1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8
9 //librería para trabajar con fechas
10 import java.util.Date;
11
12 @Entity
13 @Table(name = "indicacion") //se referencia a la tabla de la BB.DD. llamada indicacion
14 public class Indicacion extends Identificable{//hereda de la clase Identificable, para
15 //tener un identificador universal único, que se autogenera con funciones.
16
17 //PROPIEDADES
18 @Column(name="FECHAINDICACION")
19 @StoredProcedure("FECHAHORA") //Anotación Openxava para añadir un calendario.
20 @Required // Se mostrará un error de validación si la propiedad se deja en blanco

```

```

21     private Date fechaIndicacion;
22
23     @Column(name="TIPOINDICACION")
24     private TipoIndicacion tipoIndicacion;
25     public enum TipoIndicacion { //se describe un tipo enumerado que se visualiza como un SELECT
26         BRIHH("BRIHH"), CARDIOPATIAISQUEMICA ("Cardiopatía Isquémica"),
27         DISFUNCIONVISEVERA ("Disfunción VI Severa"), ENFERMEDADELSENO ("Enfermedad del Seno"),
28         FACONRESPUESTAVENTRICULARLENTA("Fa con Respuesta Ventricular Lenta"),
29         HISTORIAFAMILIARMUERTESUBITA("Historia Familiar Muerte Súbita"),
30         INDUCCIONTVNEEF("Inducción TV en EEF"),
31         MARCAPASOSDEPENDIENTE("Marcapasos Dependiente"), MIOCARDIOPATIADILATADA("Miocardiopatía Dilatada"),
32         OTRO ("Otro"), SDQTLARGO("SD QT Largo"), TVSDOCUMENTADA("TVS Documentada");
33
34     private String nombreTipoIndicacion;
35
36     private TipoIndicacion (String nombreTipoIndicacion){
37         this.nombreTipoIndicacion = nombreTipoIndicacion;
38     }
39
40     public String getNombreTipoIndicacion() {
41
42         return nombreTipoIndicacion;
43     }
44 }
45
46 @Column(name="COMENTARIOINDICACION", nullable = true)
47 private String comentarioIndicacion;
48
49 //REFERENCIAS
50 @ManyToOne //Referencia que no permite Indicación sin Paciente
51 @JoinColumn( //se define la clave foránea de la tabla
52     name="numeroNhc", //columna para la clave foránea
53     nullable = true,
54     foreignKey = @ForeignKey(name = "fk_indicacion_numeronhc"))
55     //fk_indicacion_numeronhc es la la clave foránea en la BB.DD de cardiología
56     private Paciente indicacionesLocal; // Una referencia Java convencional
57
58 //MÉTODOS
59
60 public Date getFechaIndicacion() {
61
62     return fechaIndicacion;
63 }
64
65 public void setFechaIndicacion(Date fechaIndicacion) {
66     this.fechaIndicacion = fechaIndicacion;
67 }
68
69 public TipoIndicacion getTipoIndicacion() {
70     return tipoIndicacion;
71 }
72
73 public void setTipoIndicacion(TipoIndicacion tipoIndicacion) {
74     this.tipoIndicacion = tipoIndicacion;
75 }
76
77 public String getComentarioIndicacion() {
78     return comentarioIndicacion;
79 }
80
81 public void setComentarioIndicacion(String comentarioIndicacion) {

```

Capturas 151: Capturas de pantalla del código de la clase *Indicacion.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *Indicacion.java*. Lo indicamos con el siguiente código:

```

375     //se deshabilita la opción borrar desde el formulario de indicación
376     @RemoveAction("")
377     //se deshabilita la opción borrar desde la lista de indicaciones
378     @RemoveSelectedAction("")
379     //si se borra paciente, se
380     //borrarán sus indicaciones
381     @OneToMany (mappedBy="indicacionesLocal", cascade=CascadeType.ALL)
382     //para poder visualizar
383     //la lista de indicaciones
384     private Collection<Indicacion> indicacion = new ArrayList<Indicacion>();

```

Captura 152: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.5.5 Procedimientos.

El usuario puede grabar procedimientos realizados al paciente en el pasado, visualizar el historial de procedimientos del paciente, los datos de cada uno, y también modificarlos.

Fecha procedimiento	Tipo procedimiento	Diagnóstico procedimiento	Comentario procedimiento
09/03/2015 00:00	Cateterismo	nbuybt	cgdethdcr
	Otro	jijiji	cvgecerct

Captura 153: Captura de pantalla de la sección Ficha de paciente, apartado Procedimientos.

Para registrar un procedimiento en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Crea una nueva entidad - Procedimiento ✕

Fecha procedimiento 📅

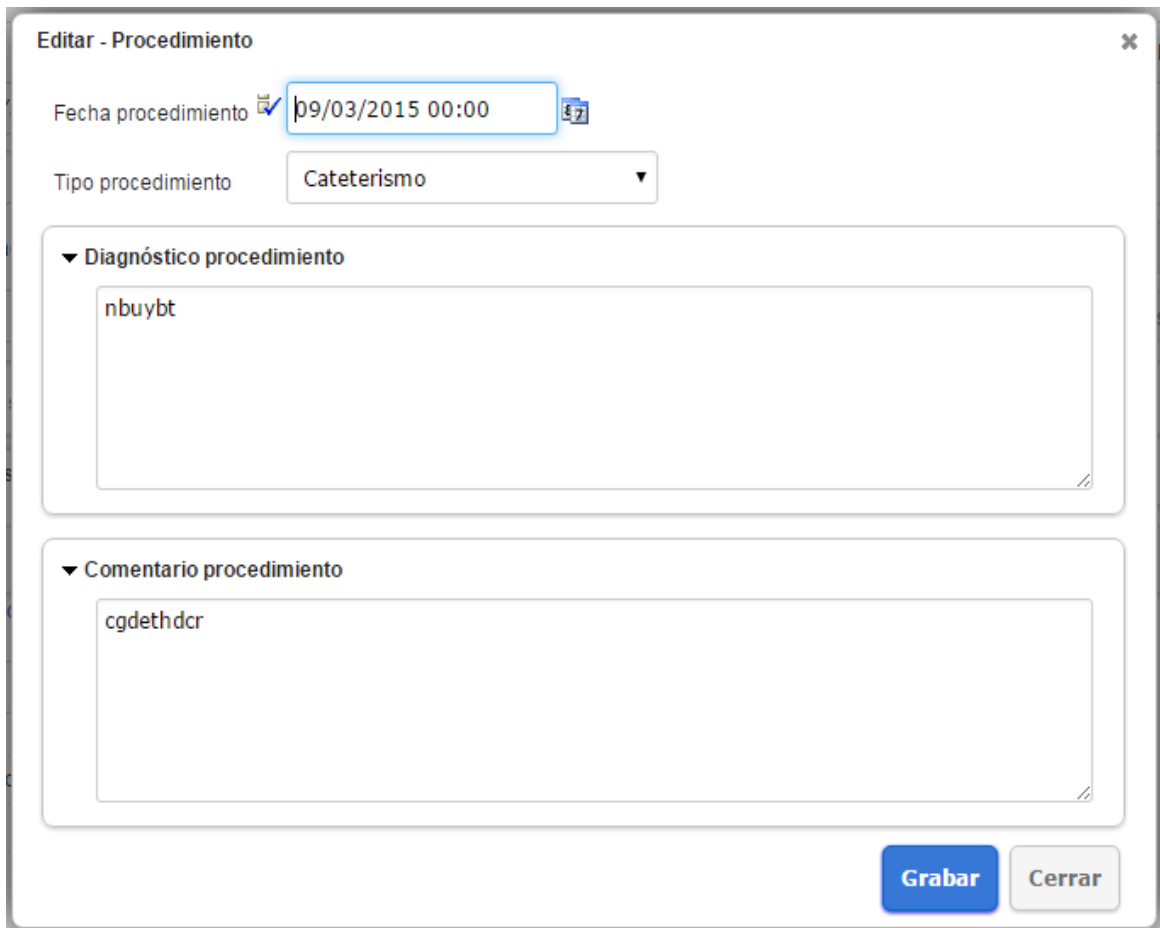
Tipo procedimiento

▼ Diagnóstico procedimiento

▼ Comentario procedimiento

Captura 154: Captura de pantalla del formulario crear procedimiento.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.



Captura 155: Captura de pantalla del formulario editar procedimiento.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *Procedimiento.java* que se ha escrito para este apartado, es el siguiente:

```

1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8
9 //librería para trabajar con fechas
10 import java.util.Date;
11
12 @Entity
13 public class Procedimiento extends Identificable{ //hereda de la clase Identificable, para
14 //tener un identificador universal único, que se autogenera con funciones
15
16     @Column(name="FECHAPROCEDIMIENTO")
17     @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
18     @Required // Se mostrará un error de validación si la propiedad se deja en blanco
19     private Date fechaProcedimiento;
20

```

```

21     @Column(name="TIPOPROCEDIMIENTO")
22     private TipoProcedimiento tipoProcedimiento;
23     public enum TipoProcedimiento { //se describe un tipo enumerado que se visualiza como un SELECT
24         CATETERISMO("Cateterismo"), ECOCARDIOGRAMA("Ecocardiograma"),
25         EEF("Estudio Electrofisiológico"), ETE("Ecocardiograma Transesofágico"),
26         OTRO("Otro"), RMN("Resonancia Magnética Nuclear"), SPECT("Espectograma"),
27         TAC("Tomografía Axial Computarizada");
28
29         private String nombreTipoProcedimiento;
30
31         private TipoProcedimiento (String nombreTipoProcedimiento){
32             this.nombreTipoProcedimiento = nombreTipoProcedimiento;
33         }
34
35         public String getNombreTipoProcedimiento() {
36             return nombreTipoProcedimiento;
37         }
38
39     }
40

```

```

41     @Column(name="DIAGNOSTICOPROCEDIMIENTO")
42     @Stereotype("MEMO") // Área de texto o equivalente
43     private String diagnosticoProcedimiento;
44
45     @Column(name="COMENTARIOPROCEDIMIENTO")
46     @Stereotype("MEMO") //Área de texto o equivalente
47     private String comentarioProcedimiento;
48
49
50     //REFERENCIAS
51     @ManyToOne //Referencia que no permite Procedimiento sin Paciente
52     @JoinColumn( //se define la clave foránea de la tabla
53         name="numeroNhc", //columna para la clave foránea
54         nullable = true,
55         foreignKey = @ForeignKey(name = "fk_procedimiento_numeronhc"))
56         //fk_procedimiento_numeronhc es la la clave foránea en la BB.DD de cardiología
57     private Paciente procedimientosLocal; // Una referencia Java convencional
58
59     //MÉTODOS
60     public Date getFechaProcedimiento() {

```

```

61         return fechaProcedimiento;
62     }
63
64     public void setFechaProcedimiento(Date fechaProcedimiento) {
65         this.fechaProcedimiento = fechaProcedimiento;
66     }
67
68     public String getDiagnosticoProcedimiento() {
69         return diagnosticoProcedimiento;
70     }
71
72     public void setDiagnosticoProcedimiento(String diagnosticoProcedimiento) {
73         this.diagnosticoProcedimiento = diagnosticoProcedimiento;
74     }
75
76     public String getComentarioProcedimiento() {
77         return comentarioProcedimiento;
78     }
79
80     public void setComentarioProcedimiento(String comentarioProcedimiento) {

```

```

81         this.comentarioProcedimiento = comentarioProcedimiento;
82     }
83
84     public TipoProcedimiento getTipoProcedimiento() {
85         return tipoProcedimiento;
86     }
87
88     public void setTipoProcedimiento(TipoProcedimiento tipoProcedimiento) {
89         this.tipoProcedimiento = tipoProcedimiento;
90     }
91
92     //Métodos para el historial de procedimientos
93     public Paciente getProcedimientosLocal() {
94         return procedimientosLocal;
95     }
96
97     public void setProcedimientosLocal(Paciente procedimientosLocal) {
98         this.procedimientosLocal = procedimientosLocal;
99     }
100
101 }

```

Capturas 156: Capturas de pantalla del código de la clase *Procedimiento.java*.

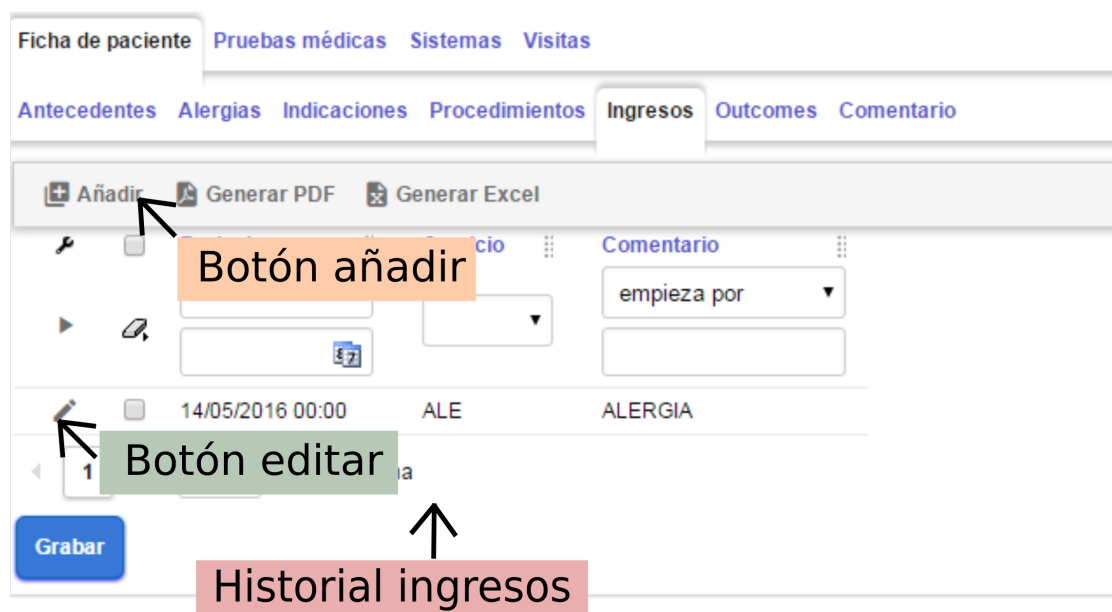
En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *Procedimiento.java*. Lo indicamos con el siguiente código:

```
386 //se deshabilita la opción borrar desde el formulario del procedimiento
387 @RemoveAction("")
388 //se deshabilita la opción borrar desde la lista de procedimientos
389 @RemoveSelectedAction("")
390 //si se borra paciente, se
391 //borrarán sus procedimientos
392 @OneToMany (mappedBy="procedimientosLocal", cascade=CascadeType.ALL)
393 //para poder visualizar
394 //la lista de procedimientos
395 private Collection<Procedimiento> procedimiento = new ArrayList<Procedimiento>();
```

Captura 157: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.5.6 Ingresos.

El usuario puede grabar ingresos del paciente en los diferentes servicios de la sanidad pública, visualizar el historial de éstos, los datos de cada uno, y también modificarlos.



Captura 158: Captura de pantalla de la sección Ficha de paciente, apartado Ingresos.

Para registrar un ingreso en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Crea una nueva entidad - Ingreso

Fecha ingreso 06/05/2017 22:58

Servicio

Comentario

ACV
ALE
ANR
CAR
CCV
CECG
CETR
CG1
CG2
CGD
CMF
CPL
CTX
DER
DIG
END
ENFS
ENRX
GER

Captura 159: Captura de pantalla del formulario crear ingreso.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.

Editar - Ingreso

Fecha ingreso 14/05/2016 00:00

Servicio ALE

Comentario

ALERGIA

Captura 160: Captura de pantalla del formulario editar ingreso.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *Ingreso.java* que se ha escrito para este apartado, es el siguiente:

```
ingreso.java x
1 package org.openxava.cardiologia.model;
2
3 //librería para trabajar con fechas
4 import java.util.*;
5
6 //librerías para persistencia de los datos
7 import javax.persistence.*;
8
9 //librerías para usar las funciones Openxava
10 import org.openxava.annotations.*;
11 import org.openxava.calculators.*;
12
13 @Entity
14 @Table(name = "ingreso") //se referencia a la tabla de la BB.DD. llamada ingreso
15 public class Ingreso extends Identificable { //hereda de la clase Identificable, para
16 //tener un identificador universal único, que se autogenera con funciones
17
18 //PROPIEDADES
19 @DefaultValueCalculator(CurrentDateCalculator.class) //claculador día actual
20 @Column(name="FECHAINGRESO")
21
22 @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
23 @Required // Se mostrará un error de validación si la propiedad se deja en blanco
24 private Date fechaIngreso;
25
26 @Column (name="SERVICIO")
27 private Servicio servicio;
28 public enum Servicio { //se describe un tipo enumerado que se visualiza como un SELECT
29 ACV("ACV"), ALE("ALE"), ANR("ANR"), CAR("CAR"), CCV("CCV"), CECG("CECG"),
30 CETR("CETR"), CG1("CG1"), CG2("CG2"), CGD("CGD"), CMF("CMF"), CPL("CPL"),
31 CTX("CTX"), DER("DER"), DIG("DIG"), END("END"), ENFS("ENFS"), ENRX("ENRX"),
32 GER("GER"), GIN("GIN"), HADO("HADO"), HEM("HEM"), HDMI("HDMI"), INFC("INFC"),
33 INM("INM"), MIN("MIN"), MPR("MPR"), NEF("NEF"), NEU("NEU"), NRC("NRC"),
34 NRL("NRL"), OFT("OFT"), ONC("ONC"), ORL("ORL"), PSQ("PSQ"), RDI("RDI"),
35 REU("REU"), RHE("RHE"), RMIN("RMIN"), RXI("RXI"), TRA("TRA"), TMO("TMO"),
36 TOC("TOC"), UCI("UCI"), UCP("UCP"), UDC("UDC"), UDH("UDH"), UMI("UMI"), URG("URG"),
37 URA("URA"), URO("URO"), URQ("URQ"), USMB("USMB"), USMC("USMC"),
38 USMP("USMP"), UVEN("UVEN");
39
40 private String nombreServicio;
41
42 private Servicio (String nombreServicio){
43     this.nombreServicio = nombreServicio;
44 }
45
46 public String getNombreServicio() {
47     return nombreServicio;
48 }
49
50 @Column(name="COMENTARIO")
51 @Stereotype("MEMO") // Área de texto o equivalente
52 private String comentario;
53
54 //REFERENCIA
55 @ManyToOne //Referencia que no permite Ingreso sin Paciente
56 @JoinColumn( //se define la clave foránea de la tabla
57     name="numeroNhc", //columna para la clave foránea
58     nullable = true,
59     foreignKey = @ForeignKey(name = "fk_ingreso_numeroNhc"))
60
```

```

61 //fk_ingreso_numeronhc es la la clave foránea en la BB.DD de cardiología
62 private Paciente ingresosLocal; // Una referencia Java convencional
63
64 //MÉTODOS
65 public Date getFechaIngreso() {
66     return fechaIngreso;
67 }
68
69 public void setFechaIngreso(Date fechaIngreso) {
70     this.fechaIngreso = fechaIngreso;
71 }
72
73 public Servicio getServicio() {
74     return servicio;
75 }
76
77 public void setServicio(Servicio servicio) {
78     this.servicio = servicio;
79 }
80
81 public String getComentario() {
82     return comentario;
83 }
84
85 public void setComentario(String comentario) {
86     this.comentario = comentario;
87 }
88
89
90 //Métodos para el historial de ingresos
91 public Paciente getIngresosLocal() {
92     return ingresosLocal;
93 }
94
95 public void setIngresosLocal(Paciente ingresosLocal) {
96     this.ingresosLocal = ingresosLocal;
97 }
98
99 }
100

```

Capturas 161: Capturas de pantalla del código de la clase *Ingresos.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *Ingresos.java*. Lo indicamos con el siguiente código:

```

397 //se deshabilita la opción borrar desde el formulario del ingreso
398 @RemoveAction("")
399 //se deshabilita la opción borrar desde la lista de ingresos
400 @RemoveSelectedAction("")
401 //si se borra paciente, se
402 //borrarán sus ingresos
403 @OneToMany (mappedBy="ingresosLocal", cascade=CascadeType.ALL)
404 //para poder visualizar
405 //la lista de ingresos
406 private Collection<Ingreso> ingreso = new ArrayList<Ingreso>();

```

Captura 162: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.5.7 Outcomes.

El usuario puede grabar información del paciente en caso de fallecimiento, visualizar esta información, y también modificarla.

Ficha de paciente Pruebas médicas Sistemas Visitas

Antecedentes Alergias Indicaciones Procedimientos Ingresos **Outcomes** Comentario

▼ Electrocardiograma basal

Ritmo Qrs Cicatriz Localizacion

▼ Dispositivos previos

MCP previo MCP dependiente DAI previo Terapias previas

▼ Outcomes

Fecha de exitus Causa exitus CV Causa exitus

▼ Comentario outcomes

Comentario outcomes

Grabar ← **Botón añadir/modificar información**

Captura 163: Captura de pantalla de la sección Ficha de paciente, apartado *Outcomes*.

El botón Grabar modifica la información del objeto paciente, en la BB.DD.

El código para la clase *Paciente.java* que se ha escrito para este apartado, es el siguiente:

```

131 //DISPOSITIVOS PREVIOS
132 @Column(name="MCPPREVIO")
133 private McpPrevio mcpPrevio;
134 //se describe un tipo enumerado que se visualiza como un SELECT
135 public enum McpPrevio {
136     SI ("Sí"), NO ("No");
137
138     private String nombreMcpPrevio;
139
140     private McpPrevio (String nombreMcpPrevio){
141         this.nombreMcpPrevio = nombreMcpPrevio;
142     }
143
144     public String getMcpPrevio() {
145         return nombreMcpPrevio;
146     }
147
148 }
149
150 @Column(name="MCPDEPENDIENTE")

```

```

151 private McpDependiente mcpDependiente;
152 //se describe un tipo enumerado que se visualiza como un SELECT
153 public enum McpDependiente {
154     SI ("Sí"), NO ("No");
155
156     private String nombreMcpDependiente;
157
158     private McpDependiente (String nombreMcpDependiente){
159         this.nombreMcpDependiente = nombreMcpDependiente;
160     }
161
162     public String getMcpDependiente() {
163         return nombreMcpDependiente;
164     }
165
166 }
167
168 @Column(name="DAIPREVIO")
169 private DaiPrevio daiPrevio;
170 //se describe un tipo enumerado que se visualiza como un SELECT
171 public enum DaiPrevio {

```

```

172     SI ("Sí"), NO ("No");
173
174     private String nombreDaiPrevio;
175
176     private DaiPrevio (String nombreDaiPrevio){
177         this.nombreDaiPrevio = nombreDaiPrevio;
178     }
179
180     public String getDaiPrevio() {
181         return nombreDaiPrevio;
182     }
183
184 }
185
186 @Column(name="TERAPIASPREVIAS")
187 private TerapiasPrevias terapiasPrevias;
188 //se describe un tipo enumerado que se visualiza como un SELECT
189 public enum TerapiasPrevias {
190     SI ("Sí"), NO ("No");
191
192     private String nombreTerapiasPrevias;

```

```

193
194     private TerapiasPrevias (String nombreTerapiasPrevias){
195         this.nombreTerapiasPrevias = nombreTerapiasPrevias;
196     }
197
198     public String getTerapiasPrevias() {
199         return nombreTerapiasPrevias;
200     }
201 }

```

```

202
203 //OUTCOMES
204 // La longitud de columna se usa a nivel UI y a nivel DB
205 @Column(name="VVMS", length=7)
206 private String vvms;
207
208 @Column(name="FECHADEEXITUS")
209 //Anotación Openxava para añadir un calendario.
210 @Stereotype("FECHAHORA")
211 private Date fechaDeExitus;
212
213 @Column(name="CAUSAEXITUSCV")
214 private CausaExitusCV causaExitusCV;
215 //se describe un tipo enumerado que se visualiza como un SELECT
216 public enum CausaExitusCV {
217     SI ("Sí"), NO ("No");
218
219     private String nombreCausaExitusCV;
220
221     private CausaExitusCV (String nombreCausaExitusCV){
222         this.nombreCausaExitusCV = nombreCausaExitusCV;
223     }
224
225     public String getCausaExitusCV() {
226         return nombreCausaExitusCV;
227     }
228
229 }

```

```

230
231 @Column(name="CAUSAEXITUS")
232 private CausaExitus causaExitus;
233 //se describe un tipo enumerado que se visualiza como un SELECT
234 public enum CausaExitus {
235     ACCIDENTE ("Accidente"), ICC ("ICC"), INFECCION ("Infección"),
236     MUERTESUBITA ("Muerte Súbita"), NEOPLASIA ("Neoplasia"),
237     OTRO ("Otro"), SEPSIS ("Sepsis"), VRTVR ("vrtvr");
238
239     private String nombreCausaExitus;
240
241     private CausaExitus (String nombreCausaExitus){
242         this.nombreCausaExitus = nombreCausaExitus;
243     }
244
245     public String getCausaExitus() {
246         return nombreCausaExitus;
247     }
248
249 }
250
251

```

```

252 @Column(name="RESPONDEDOR")
253 private Respondedor respondedor;
254 //se describe un tipo enumerado que se visualiza como un SELECT
255 public enum Respondedor {
256     SI ("Sí"), NO ("No");
257

```

```

258     private String nombreRespondedor;
259
260     private Respondedor (String nombreRespondedor){
261         this.nombreRespondedor = nombreRespondedor;
262     }
263
264     public String getRespondedor() {
265         return nombreRespondedor;
266     }
267
268 }
269
270 @Column(name="REMODELADOR")
271 private Remodelador remodelador;
272 //se describe un tipo enumerado que se visualiza como un SELECT
273 public enum Remodelador {
274     SI ("Sí"), NO ("No");
275
276     private String nombreRemodelador;
277
278     private Remodelador (String nombreRemodelador){
279         this.nombreRemodelador = nombreRemodelador;
280     }
281
282     public String getRemodelador() {
283         return nombreRemodelador;
284     }
285 }

```

```

287 //ELECTROCARDIOGRAMABASAL
288
289 @Column(name="RITMO")
290 private Ritmo ritmo;
291 //se describe un tipo enumerado que se visualiza como un SELECT
292 public enum Ritmo {
293     VRTVR ("vrtvr"), SINUSAL ("Sinusal"), FA("FA"), OTRO ("Otro");
294
295     private String nombreRitmo;
296
297     private Ritmo (String nombreRitmo){
298         this.nombreRitmo = nombreRitmo;
299     }
300
301     public String getRitmo() {
302         return nombreRitmo;
303     }
304
305 }

```

```

306
307 // La longitud de columna se usa a nivel UI y a nivel DB
308 @Column(name="QRS", length=7)
309 private String qrs;
310
311 @Column(name="CICATRIZ")
312 private Cicatriz cicatriz;
313 //se describe un tipo enumerado que se visualiza como un SELECT
314 public enum Cicatriz {
315     SI ("Sí"), NO ("No");

```

```

316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343

    private String nombreCicatriz;

    private Cicatriz (String nombreCicatriz){
        this.nombreCicatriz = nombreCicatriz;
    }

    public String getCicatriz() {
        return nombreCicatriz;
    }
}

@Column(name="LOCALIZACION")
private Localizacion localizacion;
//se describe un tipo enumerado que se visualiza como un SELECT
public enum Localizacion {
    ANT ("Ant"), ANTEROLATERAL ("Anterolateral"),
    ANTEROSEPTAL ("Anteroseptal"), INF ("Inf"),
    INFEROPOSTERIOR ("Inferoposterior"), LATERAL ("Lateral"),
    POST ("Post"), POSTEROLATERAL ("Posterolateral");

    private String nombreLocalizacion;

    private Localizacion (String nombreLocalizacion){
        this.nombreLocalizacion = nombreLocalizacion;
    }

    public String getLocalizacion() {
        return nombreLocalizacion;
    }
}

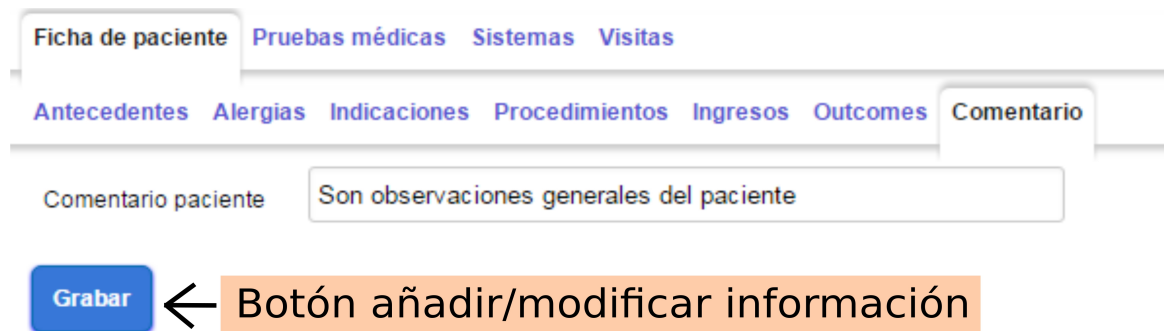
//COMENTARIO OUTCOMES
@Column(name="COMENTARIOOUTCOMES", nullable = true)
private String comentarioOutcomes;
353

```

Capturas 164: Capturas de pantalla de fragmentos de código de la clase *Paciente.java*.

5.5.8 Comentario.

El usuario puede grabar información adicional del paciente, visualizarla, y también modificarla.



Captura 165: Captura de pantalla de sección Paciente, apartado Comentario.

El botón Grabar modifica la información comentario del objeto paciente en la BB.DD.

Para añadir este requisito, implementamos la clase comentario como una clase incrustada en la clase paciente.

El código de la clase comentario es el siguiente:

```
comentario.java x
1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 @Entity
7 @Table(name = "comentario") //se referencia a la tabla de la BB.DD. llamada comentario
8 public class Comentario extends Identificable{ //hereda de la clase Identificable, para
9 //tener un identificador universal único, que se autogenera con funciones
10
11 //PROPIEDADES
12 @Column(name="COMENTARIOPACIENTE", nullable = true)
13 private String comentarioPaciente;
14
15 //MÉTODOS
16 public String getComentarioPaciente() {
17     return comentarioPaciente;
18 }
19
20 public void setComentarioPaciente(String comentarioPaciente) {
21     this.comentarioPaciente = comentarioPaciente;
22 }
23
24 }
```

Captura 166: Captura de pantalla del código de la clase *Comentario.java*.

En la clase Paciente, se debe indicar que esta clase estará incrustada:


```

354 //INCRUSTABLES
355 @OneToOne
356 @JoinColumn( //se define la clave foránea de la tabla paciente
357             name = "id", //columna que corresponde con el id de comentario
358             nullable = true,
359             foreignKey = @ForeignKey(name = "fk_paciente_id")) //nombre de
360             //la clave foránea en la tabla paciente de la BB.DD.
361 @AsEmbedded //Anotación Openxava para incrustar una clase en otra.
362 private Comentario comentario;

```

Captura 167: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.6 Generación automática de informes de historiales.

Openxava, por defecto, permite la impresión en *pdf* y exportación a *excel* de cualquier colección de la aplicación web de cardiología a través de dos botones en el menú principal del apartado, por lo tanto, como extra en la aplicación de cardiología, podremos imprimir todos los historiales que se requieran. Como ejemplo, vemos cómo podemos imprimir en *pdf* el historial de antecedentes del paciente, y cómo podemos exportarlos a *excel*. Esto último puede ser útil, si necesitáramos trabajar con estadísticas, calcular algún valor a partir de parámetros de pruebas médicas, etc.

The screenshot shows a web interface for a patient's medical history. At the top, there are tabs for 'Ficha de paciente', 'Pruebas médicas', 'Sistemas', and 'Visitas'. Below these, there are sub-tabs for 'Antecedentes', 'Alergias', 'Indicaciones', 'Procedimientos', 'Ingresos', 'Outcomes', and 'Comentario'. A toolbar contains buttons for '+ Añadir', 'Generar PDF', and 'Generar Excel'. The 'Generar Excel' button is highlighted with a green box and labeled 'Botón Excel' with an arrow. The 'Generar PDF' button is highlighted with an orange box and labeled 'Botón PDF' with an arrow. Below the toolbar is a table with columns for 'Fecha antecedente', 'Tipo antecedente', and 'Comentario antecedente'. The table contains several rows of medical history entries. At the bottom, there is a pagination control showing '1' and '10' files per page.

	Fecha antecedente	Tipo antecedente	Comentario antecedente
	=		empieza por
	11/05/2016 00:00	Asma Bronquial	
	04/03/2013 00:00	AIT	comprobar cambios
	09/03/2015 00:00	Otro	otro cambio
	15/07/2014 00:00	DM1	comprobar uno nuevo
	07/09/2016 00:00	AIT	prueba
	19/08/2016 00:00	Bebedor Activo	

Captura 168: Botones para exportación de historiales.

Antecedente de Paciente: Yaiza Santana

Fecha antecedente	Tipo antecedente	Comentario antecedente
11/05/2016 00:00	Asma Bronquial	
04/03/2013 00:00	AIT	comprobar cambios
09/03/2015 00:00	Otro	otro cambio
15/07/2014 00:00	DM1	comprobar uno nuevo
07/09/2016 00:00	AIT	prueba
19/08/2016 00:00	Bebedor Activo	

Captura 169: Captura de pantalla del archivo *pdf* que contiene del historial de antecedentes exportado.

The screenshot shows the OpenOffice Calc interface with a CSV file named 'Antecedente de Paciente- Yaiza Santana 20170507_1144.csv'. The spreadsheet contains the following data:

	A	B	C
1	Fecha antecedente	Tipo antecedente	
2	11/05/16 0:00	Asma Bronquial	
3	4/03/13 0:00	AIT	comprobar cambios
4	9/03/15 0:00	Otro	otro cambio
5	15/07/14 0:00	DM1	comprobar uno nuevo
6	7/09/16 0:00	AIT	prueba
7	19/08/16 0:00	Bebedor Activo	

Captura 170: Captura de pantalla del archivo *.csv* que contiene del historial de antecedentes exportado.

5.7 Sección Pruebas médicas.

En la especificación de requisitos se pide que en cada apartado de prueba médica, se pueda visualizar el historial de pruebas de ese tipo, visualizar los datos de cada prueba, y también modificarlas.

En este capítulo se explicará cómo se implementó cada tipo de prueba médica cardiológica. y las referencias en la clase paciente y de cada una de las pruebas médicas, ya que el paciente puede realizarse 0, 1 o muchas pruebas médicas de cualquier tipo.

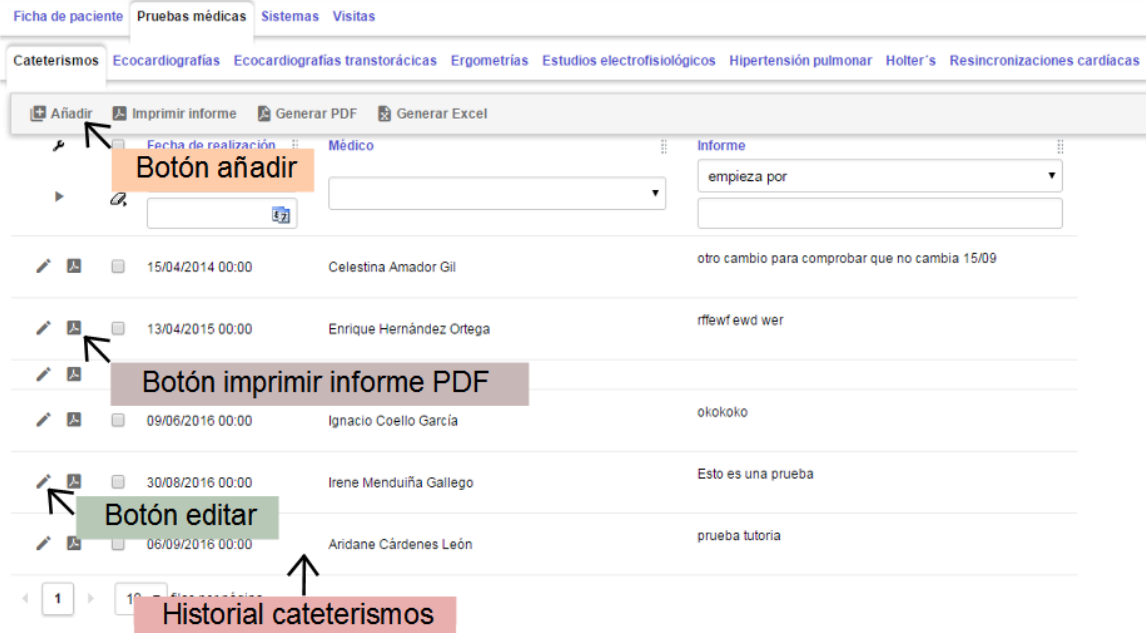
Al explicarse anteriormente cómo implementar los informes PDF personalizados de pruebas médicas, sólo añadiremos el código con sus respectivos comentarios.

5.7.1 Cateterismo.

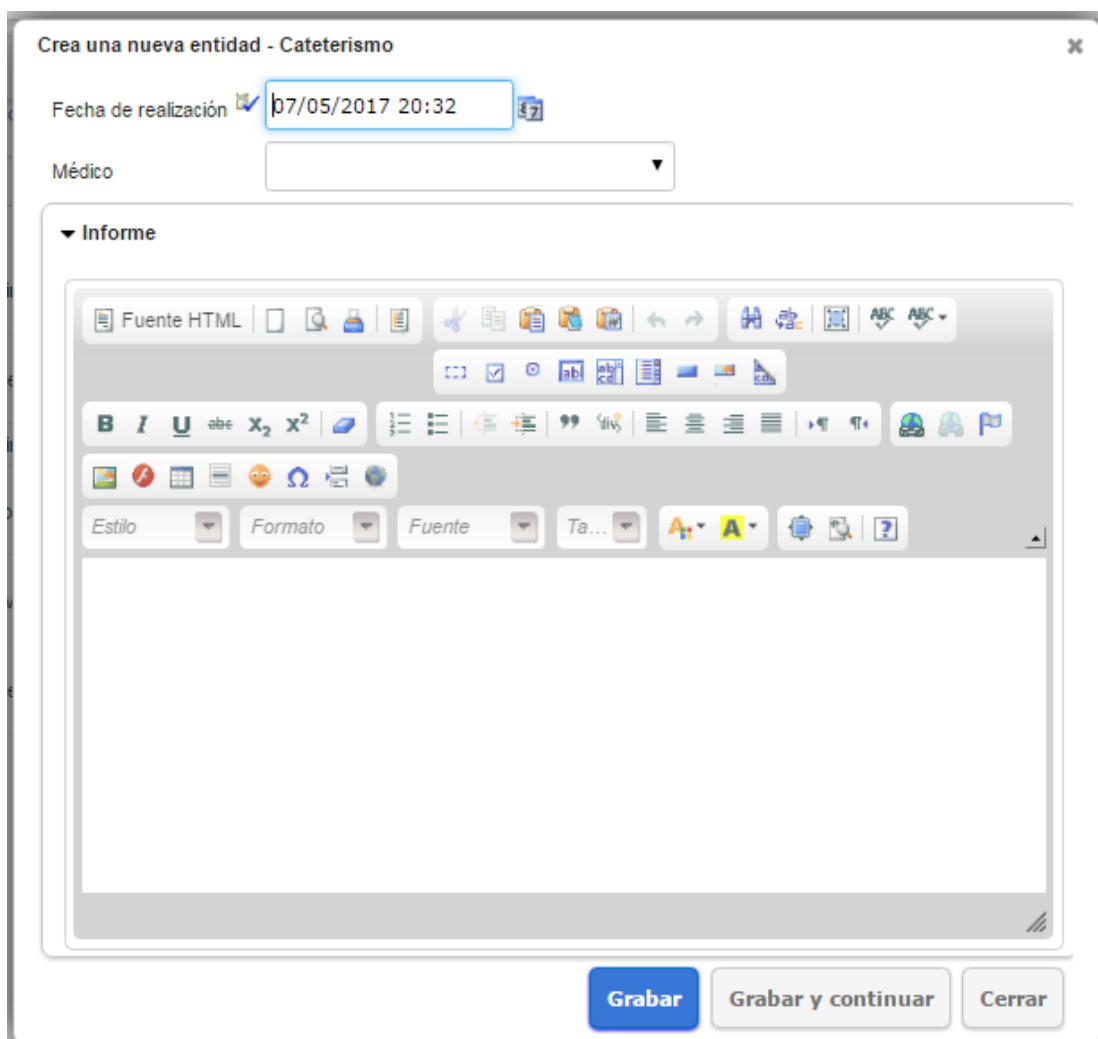
El usuario puede grabar pruebas de cateterismo del paciente, visualizar el historial de éstas, los datos de cada una, y también modificarlas.

Para registrar una prueba de cateterismo en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Si desea imprimir el informe *pdf*, puede hacerlo a través de un botón.

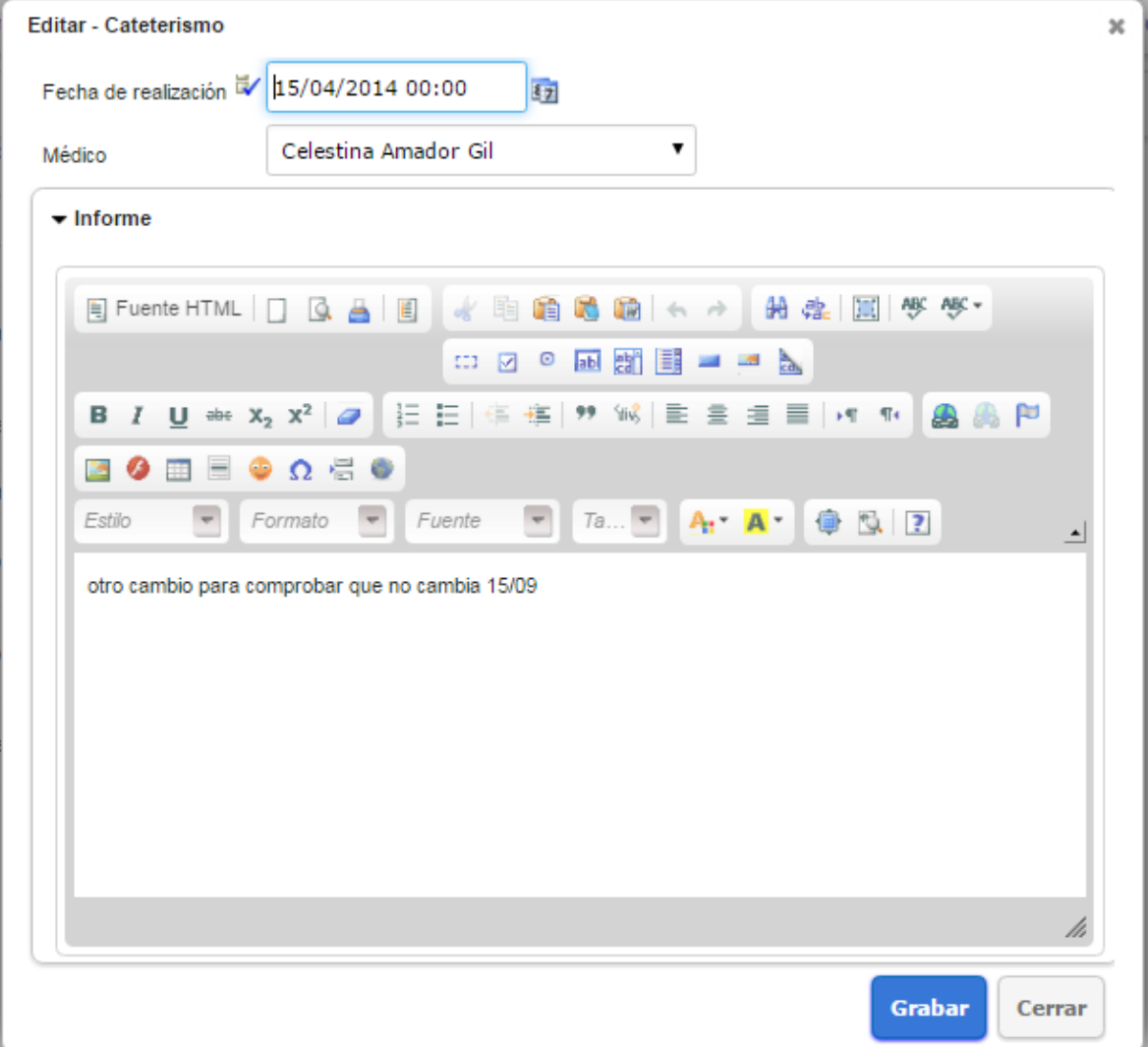


Captura 171: Captura de pantalla de la sección Pruebas médicas, apartado Cateterismos.



Captura 172: Captura de pantalla del formulario crear cateterismo.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.



Editar - Cateterismo

Fecha de realización 15/04/2014 00:00

Médico Celestina Amador Gil

▼ Informe

Fuente HTML

otro cambio para comprobar que no cambia 15/09

Grabar Cerrar

Captura 173: Captura de pantalla del formulario editar cateterismo.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *Cateterismo.java* que se ha escrito para este apartado, es el siguiente:

```
cateterismo.java x
1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8 import org.openxava.calculators.*;
9
10 //librería para trabajar con fechas
11 import java.util.Date;
12
13 @Entity
14 @Table(name = "cateterismo") //se referencia a la tabla de la BB.DD. llamada cateterismo
15 public class Cateterismo extends Identificable{ //hereda de la clase Identificable, para
16 //tener un identificador universal único, que se autogenera con funciones
17
18 //PROPIEDADES
19 @DefaultValueCalculator(CurrentDateCalculator.class) //calculador para darle al campo
20 //por defecto la fecha actual
21
22 @Column(name="FECHAREALIZACION")
23 @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
24 @Required // Se mostrará un error de validación si la propiedad se deja en blanco
25 private Date fechaRealizacion;
26
27 @Column (name="MEDICO")
28 private Medico medico;
29 public enum Medico { //se describe un tipo enumerado que se visualiza como un SELECT
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
```

```

81 //este campo medico2 se usa para visualizar correctamente el nombre del médico
82 //en el informe pdf
83 @Hidden
84 @Column (name="MEDICO2")
85 private String medico2;
86
87
88 public String getMedico2() {
89     return medico2;
90 }
91
92 public void setMedico2(String medico2) {
93     this.medico2 = medico2;
94 }
95
96 @Column(name="INFORME", length = 500, nullable = true)
97 @Stereotype("HTML_TEXT") // Estereotipo que permite visualizar el editor de texto
98 private String informe;
99
100
101 //REFERENCIA
102 @ManyToOne //Referencia que no permite Cateterismo sin Paciente
103 @JoinColumn( //se define la clave foránea de la tabla
104     name="numeroNhc", //columna para la clave foránea
105     nullable = true,
106     foreignKey = @ForeignKey(name = "fk_cateterismo_umeronhc"))
107 //fk_cateterismo_umeronhc es la la clave foránea en la BB.DD de cardiología
108 private Paciente cateterismosLocal; // Una referencia Java convencional
109
110 //MÉTODOS
111 public Date getFechaRealizacion() {
112     return fechaRealizacion;
113 }
114
115 public void setFechaRealizacion(Date fechaRealizacion) {
116     this.fechaRealizacion = fechaRealizacion;
117 }
118
119 public String getInforme() {
120     return informe;
121 }
122
123 public void setInforme(String informe) {
124     this.informe = informe;
125 }
126
127 public Medico getMedico() {
128     return medico;
129 }
130
131 //en este procedimiento lo que hacemos es revisar la variable medico, y asignarle
132 //la cadena de texto del nombre del médico a la variable medico2, ya que sino en
133 //el informe pdf no se verá correctamente el nombre del facultativo.
134 public void setMedico(Medico medico) {
135     this.medico = medico;

```

...se omiten los nombres de los médicos...

```

261 }
262
263 public void setCateterismosLocal(Paciente cateterismosLocal) {
264     this.cateterismosLocal = cateterismosLocal;
265 }
266 }
267

```

Captura 174: Capturas de pantalla del código de la clase *Cateterismo.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *Cateterismos.java*. Lo indicamos con el siguiente código:

```

425 //se deshabilita la opción borrar desde el formulario de cateterismo
426 @RemoveAction("")
427 //se deshabilita la opción borrar desde la lista de cateterismos
428 @RemoveSelectedAction("")
429 //ejemplo de la acción que se repite por cada fila de la lista,
430 //la cual imprime el informe pdf del elemento de la lista elegido a través del botón.
431 @ListAction("controlpdf.ImprimirInforme")
432 //si se borra paciente, se
433 //borrarán sus pruebas médicas de cateterismo
434 @OneToMany(mappedBy="cateterismosLocal", cascade=CascadeType.ALL)
435 //para poder visualizar
436 //la lista de pruebas cateterismo
437 private Collection<Cateterismo> cateterismo = new ArrayList<Cateterismo>();

```

Captura 175: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.7.2 Ecocardiografías.

El usuario puede grabar pruebas de ecocardiografías del paciente, visualizar el historial de éstas, los datos de cada una, y también modificarlas.

Para registrar una prueba de ecocardiografía en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Si desea imprimir el informe *pdf*, puede hacerlo a través de un botón.

The screenshot displays a web application interface for medical tests. At the top, there are navigation tabs: 'Ficha de paciente', 'Pruebas médicas', 'Sistemas', and 'Visitas'. Under 'Pruebas médicas', there are sub-tabs for 'Cateterismos', 'Ecocardiografías', 'Ecocardiografías transtorácicas', 'Ergometrias', 'Estudios electrofisiológicos', 'Hipertensión pulmonar', 'Holter's', and 'Resincronizaciones cardíacas'. The 'Ecocardiografías' tab is active, showing a toolbar with 'Añadir', 'Generar PDF', and 'Generar Excel' buttons. Below the toolbar is a form for adding a new test, with fields for 'Fecha de realización', 'Médico', and 'Informe'. A table below the form lists existing tests with columns for date, doctor, and test name. Annotations with arrows point to the 'Añadir' button, the 'Imprimir PDF' icon, the 'Editar' icon, and the table header 'Historial ecocardiografías'. A 'Grabar' button is visible at the bottom left.

Captura 176: Captura de pantalla de la sección Pruebas médicas, apartado ecocardiografías.

Crea una nueva entidad - Ecocardiografía

▼ Ecocardiografía

Fecha de realización 07/05/2017 21:32 Médico

Plantilla

▼ Informe

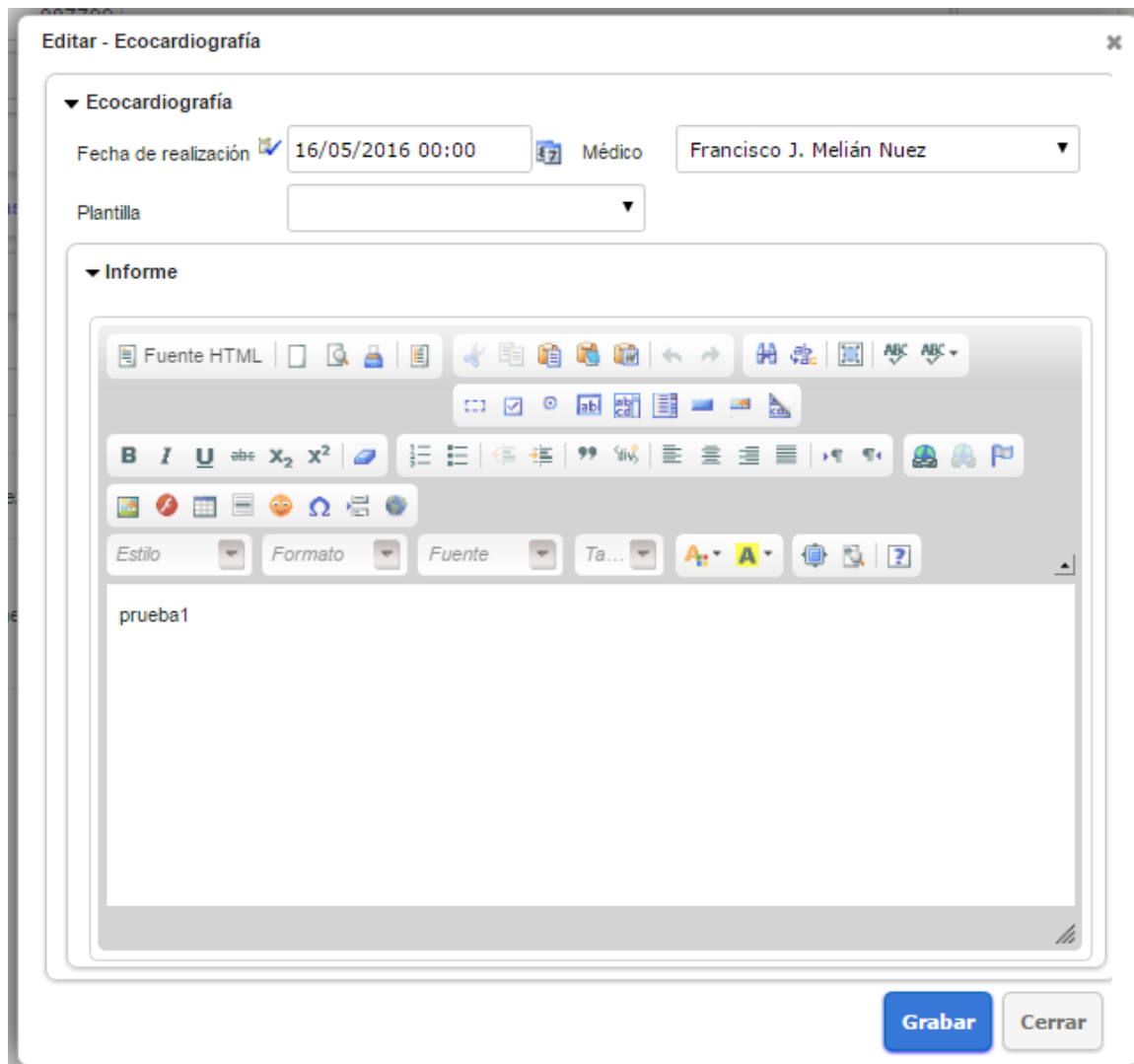
Fuente HTML

Se llega a la Fc máxima(), sin clínica ni cambios en el ECG (basal en)
Respuesta tensional adecuada (TA basal TA max)
Ecocardiográficamente no se aprecian cambios en la contractilidad regional a ningún nivel
CONCLUSIÓN:

Grabar Grabar y continuar Cerrar

Captura 177: Captura de pantalla del formulario crear ecocardiografía.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.



Captura 178: Captura de pantalla del formulario editar ecocardiografía.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *Ecocardiografia.java* que se ha escrito para este apartado, es el siguiente:

```

ecocardiografia.java
1 package org.openxava cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librería para trabajar con fechas
7 import java.util.Date;
8
9 //librerías para usar las funciones Openxava
10 import org.openxava.annotations.*;
11 import org.openxava.calculators.*;
12 import org.openxava cardiologia.actions.*;
13
14 @Entity
15 @View(members = //disponemos los parámetros para las vistas
16 "Ecocardiografia [" +
17 " fechaRealizacion, medico;" +
18 " plantilla;" +
19 " informe;" +
20 "];"
21 )
22 @Table(name = "ecocardiografia") //se referencia a la tabla de la BB.DD. llamada cateterismo
23 public class Ecocardiografia extends Identificable{ //hereda de la clase Identificable, para
24 //tener un identificador universal único, que se autogenera con funciones
25
26 //PROPIEDADES
27 @DefaultValueCalculator(CurrentDateCalculator.class //calculador para darle al campo
28 //por defecto la fecha actual
29 @Column(name="FECHAREALIZACION")
30 @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
31 @Required // Se mostrará un error de validación si la propiedad se deja en blanco
32 private Date fechaRealizacion;
33
34 @Column (name="MEDICO")
35 private Medico medico;
36 public enum Medico { //se describe un tipo enumerado que se visualiza como un SELECT

```

...se omiten los nombres de los médicos...

```

80 private Medico (String nombreMedico){
81     this.nombreMedico = nombreMedico;
82 }
83
84 public String getNombreMedico() {
85     return nombreMedico;
86 }
87
88 }
89
90 @Column(name="INFORME", length = 500, nullable = true)
91 @Stereotype("HTML_TEXT") // Estereotipo que permite visualizar el editor de texto
92 private String informe;
93
94 //PROPIEDAD TRANSITORIA
95 @Transient
96 @OnChange(CargarPlantilla.class) //para cargar plantilla en función de la elección del SELECT
97 private Plantilla plantilla;
98 public enum Plantilla {
99     ECOCARDIOGRAMATRANSTORACICO ("Ecocardiograma transtorácico"),
100     ECOCARDIOGRAMATRANSESOFAGICO ("Ecocardiograma transesofágico"),

```

```

101
102 private String nombrePlantilla;
103
104 private Plantilla (String nombrePlantilla){
105     this.nombrePlantilla = nombrePlantilla;
106 }
107
108 public String getNombrePlantilla() {
109     return nombrePlantilla;
110 }
111
112 }
113
114 //MÉTODOS PROPIEDAD TRANSITORIA
115 @Transient
116 public Plantilla getPlantilla() {
117     return plantilla;
118 }
119
120 @Transient
121 public void setPlantilla(Plantilla plantilla) {

```

```

121         this.plantilla = plantilla;
122     }
123
124
125     //REFERENCIA
126     @ManyToOne //Referencia que no permite Ecocardiografía sin Paciente
127     @JoinColumn //se define la clave foránea de la tabla
128         name="numeroNhc", //columna para la clave foránea
129         nullable = true,
130         foreignKey = @ForeignKey(name = "fk_ecocardiografia_numeronhc"))
131     //fk_ecocardiografia_numeronhc es la la clave foránea en la BB.DD de cardiología
132     private Paciente ecocardiografiasLocal; // Una referencia Java convencional
133
134     //MÉTODOS
135     public Date getFechaRealizacion() {
136         return fechaRealizacion;
137     }
138
139     public void setFechaRealizacion(Date fechaRealizacion) {
140         this.fechaRealizacion = fechaRealizacion;
141     }
142
143     public String getInforme() {
144         return informe;
145     }
146
147     public void setInforme(String informe) {
148         this.informe = informe;
149     }
150
151     public Medico getMedico() {
152         return medico;
153     }
154
155     public void setMedico(Medico medico) {
156         this.medico = medico;
157     }
158
159     //Métodos para el historial de ecocardiografías
160     public Paciente getEcocardiografiasLocal() {
161
162         return ecocardiografiasLocal;
163     }
164
165     public void setEcocardiografiasLocal(Paciente ecocardiografiasLocal) {
166         this.ecocardiografiasLocal = ecocardiografiasLocal;
167     }
168 }
169

```

Captura 179: Captura de pantalla de la clase *Ecocardiografia.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *Ecocardiografia.java*. Lo indicamos con el siguiente código:

```

439     //se deshabilita la opción borrar desde el formulario
440     @RemoveAction("")
441     //se deshabilita la opción borrar desde la lista
442     @RemoveSelectedAction("")
443     //si se borra paciente, se
444     //borrarán sus pruebas médicas de ecocardiografías
445     @OneToMany (mappedBy="ecocardiografiasLocal", cascade=CascadeType.ALL)
446     //para poder visualizar
447     //la lista de pruebas ecocardiografia
448     private Collection<Ecocardiografia> ecocardiografia = new ArrayList<Ecocardiografia>();

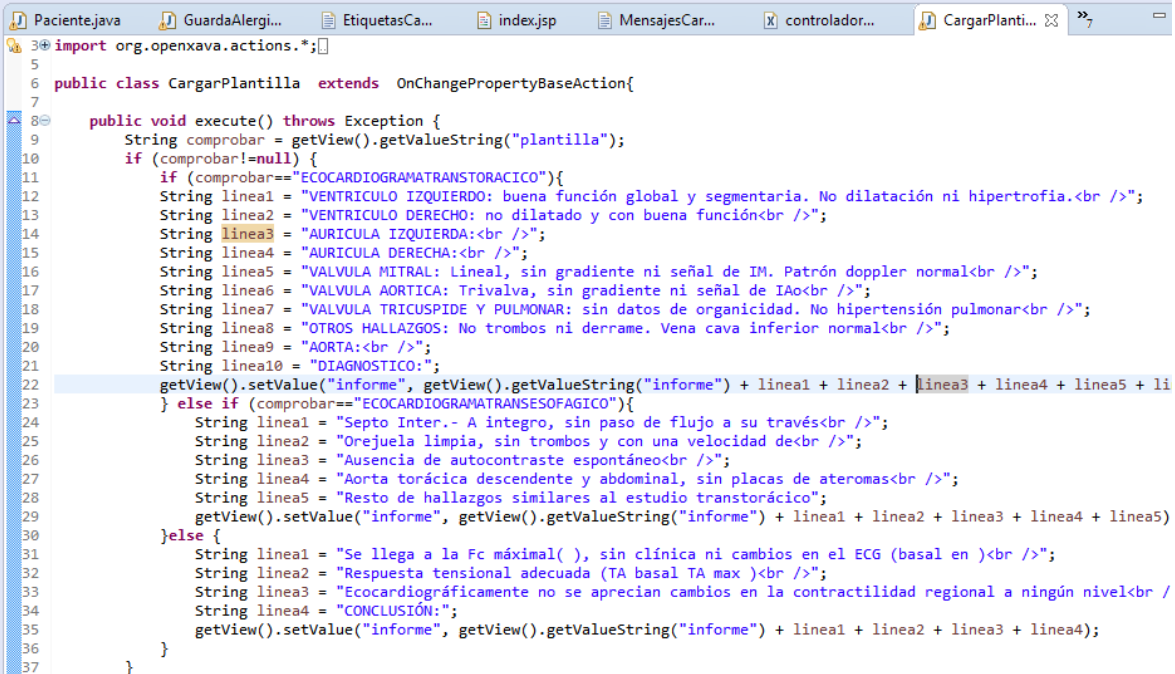
```

Captura 180: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.7.2.1. Carga de plantillas para la prueba de ecocardiografía.

Para facilitar el trabajo a los especialistas de cardiología, se les añadió la opción de cargar una plantilla de texto en el área de texto en función del valor un campo enumerado llamado Plantilla. Por defecto se añade un texto al crear una nueva prueba, que pueden editar, y volver a añadir más plantillas al cambiar este campo.

Para ello fue necesario escribir una clase acción *CargarPlantilla.java* e incluirla en el paquete de acciones del proyecto, añadir la propiedad y métodos transitorios anterior en la clase *Ecocardiografia.java*, y añadir las etiquetas para la internacionalización de las plantillas.



```
1  Paciente.java  GuardaAlergi...  EtiquetasCa...  index.jsp  MensajesCar...  controlador...  CargarPlantilla...  »
2  import org.openxava.actions.*;
3
4
5
6  public class CargarPlantilla extends OnChangePropertyBaseAction{
7
8  public void execute() throws Exception {
9      String comprobar = getView().getValueString("plantilla");
10     if (comprobar!=null) {
11         if (comprobar=="ECOCARDIOGRAMATRANSTORACICO"){
12             String linea1 = "VENTRICULO IZQUIERDO: buena función global y segmentaria. No dilatación ni hipertrofia.<br />";
13             String linea2 = "VENTRICULO DERECHO: no dilatado y con buena función<br />";
14             String linea3 = "AURICULA IZQUIERDA:<br />";
15             String linea4 = "AURICULA DERECHA:<br />";
16             String linea5 = "VALVULA MITRAL: lineal, sin gradiente ni señal de IM. Patrón doppler normal<br />";
17             String linea6 = "VALVULA AORTICA: Trivalva, sin gradiente ni señal de IAo<br />";
18             String linea7 = "VALVULA TRICUSPIDE Y PULMONAR: sin datos de organicidad. No hipertensión pulmonar<br />";
19             String linea8 = "OTROS HALLAZGOS: No trombos ni derrame. Vena cava inferior normal<br />";
20             String linea9 = "AORTA:<br />";
21             String linea10 = "DIAGNOSTICO:";
22             getView().setValue("informe", getView().getValueString("informe") + linea1 + linea2 + linea3 + linea4 + linea5 + li
23         } else if (comprobar=="ECOCARDIOGRAMATRANSESOFGAGICO"){
24             String linea1 = "Septo Inter.- A íntegro, sin paso de flujo a su través<br />";
25             String linea2 = "Orejuela limpia, sin trombos y con una velocidad de<br />";
26             String linea3 = "Ausencia de autocontraste espontáneo<br />";
27             String linea4 = "Aorta torácica descendente y abdominal, sin placas de ateromas<br />";
28             String linea5 = "Resto de hallazgos similares al estudio transtorácico";
29             getView().setValue("informe", getView().getValueString("informe") + linea1 + linea2 + linea3 + linea4 + linea5)
30         } else {
31             String linea1 = "Se llega a la Fc máxima( ), sin clínica ni cambios en el ECG (basal en )<br />";
32             String linea2 = "Respuesta tensional adecuada (TA basal TA max )<br />";
33             String linea3 = "Ecocardiográficamente no se aprecian cambios en la contractilidad regional a ningún nivel<br />";
34             String linea4 = "CONCLUSIÓN:";
35             getView().setValue("informe", getView().getValueString("informe") + linea1 + linea2 + linea3 + linea4);
36         }
37     }
38 }
```

Captura 181: Captura de pantalla del código de la clase *CargarPlantilla.java*.

La línea de código `getView().setValue("informe", getView().getValueString("informe") + linea1 + linea2 + linea3 + linea4);` sirve para que se pueda añadir varias plantillas en un único informe, porque si no lo sobrescribe.

```
85 # Componentes de Ecocardiografía
86 Ecocardiografia= Ecocardiografia
87 ECOCARDIOGRAMATRANSTORACICO=Ecocardiograma transtorácico
88 ECOCARDIOGRAMATRANSESOFGAGICO=Ecocardiograma transesofágico
89 ECOCARDIOGRAMADESTRESS=Ecocardiograma de stress
90
```

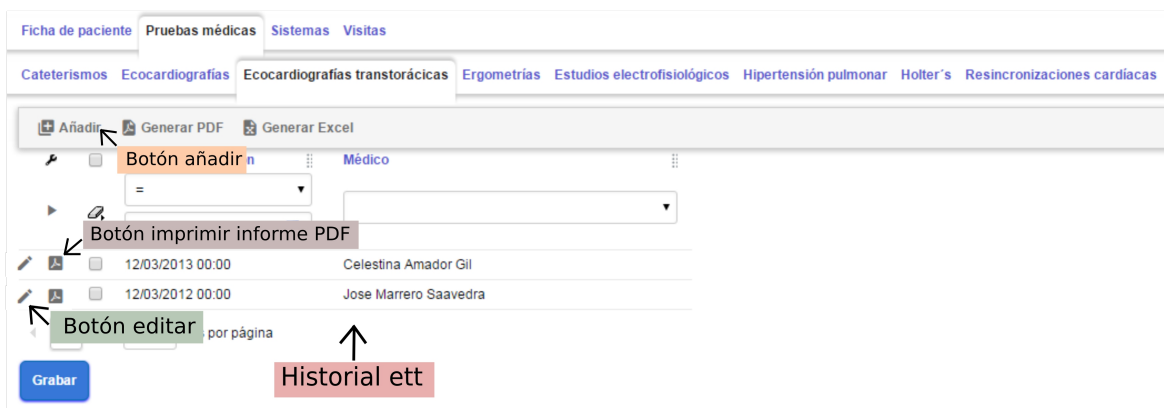
Captura 182: Captura de pantalla de fragmento de código de *EtiquetasCardiologia_es.properties*.

5.7.3. Ecocardiografía Transtorácica.

El usuario puede grabar pruebas de ecocardiografías transtorácicas del paciente, visualizar el historial de éstas, los datos de cada una, y también modificarlas.

Para registrar una prueba de ecocardiografía transtorácica en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Si desea imprimir el informe *pdf*, puede hacerlo a través de un botón.



Captura 183: Captura de pantalla de la sección Pruebas médicas, apartado Ecocardiografías Transtorácicas.

Crea una nueva entidad - Ecocardiografía transtorácica

▼ Ecocardiografía transtorácica

Fecha de realización 07/05/2017 23:40

Médico

DTDVI	<input type="text"/> mm	DTSVI	<input type="text"/> mm
Teicholtz	<input type="text"/> %	TIV	<input type="text"/> mm
PP	<input type="text"/> mm	DTDVD	<input type="text"/> mm
AI	<input type="text"/> mm	Raíz Ao	<input type="text"/> mm
GM Mitral	<input type="text"/>	GP Aórtico	<input type="text"/>
GM Aórtico	<input type="text"/>	PAPs	<input type="text"/>
E-TV	<input type="text"/> mm	Onda E	<input type="text"/>
Onda A	<input type="text"/>	FE	<input type="text"/> Simpson

Captura 184: Captura de pantalla del formulario crear Ecocardiografía transtorácica.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.

Editar - Ecocardiografía transtorácica

▼ Ecocardiografía transtorácica

Fecha de realización 12/03/2012 00:00

Médico

DTDVI	<input type="text" value="2,00"/> mm	DTSVI	<input type="text" value="2,00"/> mm
Teicholtz	<input type="text" value="2,00"/> %	TIV	<input type="text" value="2,00"/> mm
PP	<input type="text" value="2,00"/> mm	DTDVD	<input type="text" value="2,00"/> mm
AI	<input type="text" value="2,00"/> mm	Raíz Ao	<input type="text" value="2,00"/> mm
GM Mitral	<input type="text" value="2"/>	GP Aórtico	<input type="text" value="2"/>
GM Aórtico	<input type="text" value="2"/>	PAPs	<input type="text" value="2"/>
E-TIV	<input type="text" value="2,00"/> mm	Onda E	<input type="text" value="2"/>
Onda A	<input type="text" value="2"/>	FE	<input type="text" value="2,00"/> Simpson

Captura 185: Captura de pantalla del formulario editar Ecocardiografía transtorácica.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *EcocardiografiaTranstoracica.java* que se ha escrito para este apartado, es el siguiente:

```

ecocardiografiatranstoracica.java
1 package org.openxava cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8 import org.openxava.calculators.*;
9
10 //librería para trabajar con fechas
11 import java.util.Date;
12
13 @Entity
14 @View(members= //disponemos los parámetros para las vistas
15 "EcocardiografiaTranstoracica [#" +
16 "fechaRealizacion;" +
17 "medico;" +
18 "dtdvi, dtsvi;" +
19 "teicholtz, tiv;" +
20 "pp, dtdvd;" +

```

```

21 "ai, raizAo;" +
22 "gmMitral, gpAortico;" +
23 "gmAortico, paps;" +
24 "eTiv, ondaE;" +
25 "ondaA, fe;" +
26 "]"
27 )
28 @Table(name = "ecocardiografiatranstoracica") //se referencia a la tabla de la BB.DD.
29 //llamada ecocardiografiatranstoracica
30 public class EcocardiografiaTranstoracica extends Identificable{ //hereda de la clase
31 //Identificable, para tener un identificador universal único, que se autogenera con funciones
32
33 //PROPIEDADES
34 @DefaultValueCalculator(CurrentDateCalculator.class) //calculador para darle al campo
35 //por defecto la fecha actual
36 @Column(name="FECHAREALIZACION")
37 @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
38 @Required // Se mostrará un error de validación si la propiedad se deja en blanco
39 private Date fechaRealizacion;
40

```

... se omiten los nombres de los médicos...

```

85
86 private String nombreMedico;
87
88 private Medico (String nombreMedico){
89     this.nombreMedico = nombreMedico;
90 }
91
92 public String getNombreMedico() {
93     return nombreMedico;
94 }
95
96
97 //PARÁMETROS DE LA PRUEBA
98 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
99 @Column(name="DTDVI", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
100 private int dtdvi;

```

```

101
102 @Stereotype("PORCENTAJE") //Anotación Openxava para añadir unidad de medida %.
103 @Column(name="TEICHOLTZ", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
104 private int teicholtz;
105
106 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
107 @Column(name="PP", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
108 private int pp;
109
110 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
111 @Column(name="AI", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
112 private int ai;
113
114 @Column(name="GMMITRAL", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
115 private int gmMitral;
116
117 @Column(name="GMAORTICO", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
118 private int gmAortico;
119
120 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.

```

```

121 @Column(name="ETIV", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
122 private int eTiv;
123
124 @Column(name="ONDAA", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
125 private int ondaA;
126
127 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
128 @Column(name="DTSVI", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
129 private int dtsvi;
130
131 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
132 @Column(name="TIV", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
133 private int tiv;
134
135 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
136 @Column(name="DTDVD", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
137 private int dtdvd;
138
139 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
140 @Column(name="RAIZAO", length=7) // La longitud de columna se usa a nivel UI y a nivel DB

```

```

141 private int raizAo;
142
143 @Column(name="GPAORTICO", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
144 private int gpAortico;
145
146 @Column(name="PAPS", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
147 private int paps;
148
149 @Column(name="ONDAE", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
150 private int ondaE;
151
152 @Stereotype("SIMPSON") //Anotación Openxava para añadir unidad de medida Simpson
153 @Column(name="FE", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
154 private int fe;
155
156
157 //REFERENCIAS
158 @ManyToOne //Referencia que no permite Ecocardiografia TT sin Paciente
159 @JoinColumn( //se define la clave foránea de la tabla
160             name="numeroNhc", //columna para la clave foránea

```

```

161             nullable = true,
162             foreignKey = @ForeignKey(name = "fk_ett_numeronhc"))
163             //fk_ett_numeronhc es la la clave foránea en la BB.DD de cardiología
164 private Paciente ettLocal; // Una referencia Java convencional
165
166 //MÉTODOS
167
168 public Date getFechaRealizacion() {
169     return fechaRealizacion;
170 }
171
172 public void setFechaRealizacion(Date fechaRealizacion) {
173     this.fechaRealizacion = fechaRealizacion;
174 }
175
176 public Medico getMedico() {
177     return medico;
178 }
179
180 public void setMedico(Medico medico) {

```

```

181     this.medico = medico;
182 }
183
184 public int getDtdvi() {
185     return dtdvi;
186 }
187
188 public void setDtdvi(int dtdvi) {
189     this.dtdvi = dtdvi;
190 }
191
192 public int getTeicholtz() {
193     return teicholtz;
194 }
195
196 public void setTeicholtz(int teicholtz) {
197     this.teicholtz = teicholtz;
198 }
199
200 public int getPp() {

```

```

201     return pp;
202 }
203
204 public void setPp(int pp) {
205     this.pp = pp;
206 }
207
208 public int getAi() {
209     return ai;
210 }
211
212 public void setAi(int ai) {
213     this.ai = ai;
214 }
215
216 public int getGmMitral() {
217     return gmMitral;
218 }
219
220 public void setGmMitral(int gmMitral) {

```

```

221     this.gmMitral = gmMitral;
222     }
223
224     public int getGmAortico() {
225         return gmAortico;
226     }
227
228     public void setGmAortico(int gmAortico) {
229         this.gmAortico = gmAortico;
230     }
231
232     public int geteTiv() {
233         return eTiv;
234     }
235
236     public void seteTiv(int eTiv) {
237         this.eTiv = eTiv;
238     }
239
240     public int getOndaA() {
241
242         return ondaA;
243     }
244
245     public void setOndaA(int ondaA) {
246         this.ondaA = ondaA;
247     }
248
249     public int getDtsvi() {
250         return dtsvi;
251     }
252
253     public void setDtsvi(int dtsvi) {
254         this.dtsvi = dtsvi;
255     }
256
257     public int getTiv() {
258         return tiv;
259     }
260
261     public void setTiv(int tiv) {
262
263         this.tiv = tiv;
264     }
265
266     public int getDtdvd() {
267         return dtdvd;
268     }
269
270     public void setDtdvd(int dtdvd) {
271         this.dtdvd = dtdvd;
272     }
273
274     public int getRaizAo() {
275         return raizAo;
276     }
277
278     public void setRaizAo(int raizAo) {
279         this.raizAo = raizAo;
280     }
281
282     public int getGpAortico() {
283
284         return gpAortico;
285     }
286
287     public void setGpAortico(int gpAortico) {
288         this.gpAortico = gpAortico;
289     }
290
291     public int getPaps() {
292         return paps;
293     }
294
295     public void setPaps(int paps) {
296         this.paps = paps;
297     }
298
299     public int getOndaE() {
300         return ondaE;
301     }
302
303     public void setOndaE(int ondaE) {

```

```

301         this.ondaE = ondaE;
302     }
303
304     public int getFe() {
305         return fe;
306     }
307
308     public void setFe(int fe) {
309         this.fe = fe;
310     }
311
312     //Métodos para el historial de ecocardiografías TT
313     public Paciente getEttLocal() {
314         return ettLocal;
315     }
316
317     public void setEttLocal(Paciente ettLocal) {
318         this.ettLocal = ettLocal;
319     }
320
321 }
322

```

Capturas 186: Capturas de pantalla del código de la clase *EcocardiografiaTranstoracica.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *EcocardiografiaTranstoracica.java*. Lo indicamos con el siguiente código:

```

449
450     //se deshabilita la opción borrar desde el formulario de ett
451     @RemoveAction("")
452     //se deshabilita la opción borrar desde la lista de ett
453     @RemoveSelectedAction("")
454     //si se borra paciente, se
455     //borrarán sus pruebas médicas de ecocardiografías transtorácicas
456     @OneToMany (mappedBy="ettLocal", cascade=CascadeType.ALL)
457     //para poder visualizar la lista de pruebas ecocardiografías transtorácicas
458     private Collection<EcocardiografiaTranstoracica> ecocardiografiaTranstoracica =
459     new ArrayList<EcocardiografiaTranstoracica>();

```

Captura 187: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.7.3.1 Estereotipos necesarios para la clase.

Por último, definimos los estereotipos mm, % y Simpson:

- Fichero *editores.xml*:

```

17<⊖ <editor nombre="milimetros" url="milimetrosEditor.jsp">
18     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
19     <para-estereotipo estereotipo="MILIMETROS"/>
20 </editor>
21
22<⊖ <editor nombre="porcentaje" url="porcentajeEditor.jsp">
23     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
24     <para-estereotipo estereotipo="PORCENTAJE"/>
25 </editor>
26
27<⊖ <editor nombre="simpson" url="simpsonEditor.jsp">
28     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
29     <para-estereotipo estereotipo="SIMPSON"/>
30 </editor>
31

```

Captura 188: Captura de pantalla de fragmento de código del script *editores.xml*.

- Clase formateadora (sólo se ha de crear una sóla vez):

```

1 package org.openxava.cardiologia.formateadores;
2
3 import java.math.*;
4
5
6
7
8
9 public class PesoFormateador implements IFormatter {
10
11     public String format(HttpServletRequest request, Object object) throws Exception {
12         if (Is.empty(object)) return "";
13         return getFormat().format(object);
14     }
15
16     public Object parse(HttpServletRequest request, String string) throws Exception {
17         if (Is.emptyString(string)) return null;
18         string = Strings.change(string, " ", ""); // In order to work with Polish
19         return new BigDecimal(getFormat().parse(string).toString()).setScale(2);
20     }
21
22     private NumberFormat getFormat() {
23         NumberFormat f = DecimalFormat.getNumberInstance(Locales.getCurrent());
24         f.setMinimumFractionDigits(2);
25         f.setMaximumFractionDigits(2);
26         return f;
27     }
28
29 }
30

```

Captura 189: Captura de pantalla de fragmento de la clase Formateadora *PesoFormateador*.

- *Scripts* de los editores:

```

1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7     symbol = "mm"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 symbol = "?";
11 }
12 %>
13
14 <%=symbol%>

```

Captura 190: Captura de pantalla del script *milimetrosEditor.jsp*.

```

1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7     symbol = "%"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 symbol = "?";
11 }
12 %>
13
14 <%=symbol%>

```

Captura 191: Captura de pantalla del script *porcentajeEditor.jsp*.

```
1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7 symbol = "Simpson"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 symbol = "?";
11 }
12 %>
13
14 <%=symbol%>
```

Captura 192: Captura de pantalla del script *simpsonEditor.jsp*.

5.7.4 Ergometrías.

El usuario puede grabar pruebas de ergometrías del paciente, visualizar el historial de éstas, los datos de cada una, y también modificarlas.

Para registrar una prueba de ergometría en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Si desea imprimir el informe pdf, puede hacerlo a través de un botón.

The screenshot displays the 'Ergometrías' section of a medical software interface. At the top, there are navigation tabs: 'Ficha de paciente', 'Pruebas médicas', 'Sistemas', and 'Visitas'. Below these, there are sub-tabs for various medical tests: 'Cateterismos', 'Ecocardiografías', 'Ecocardiografías transtorácicas', 'Ergometrías', 'Estudios electrofisiológicos', 'Hipertensión pulmonar', 'Holter's', and 'Resincronizaciones cardíacas'. The 'Ergometrías' tab is active. Below the sub-tabs, there are three buttons: 'Añadir', 'Generar PDF', and 'Generar Excel'. The main area contains a table with the following columns: 'Fecha de realización', 'Médico', 'Resultado', and 'Notas'. The first row shows a test performed on 17/03/2014 at 00:00 by Enrique Hernández Ortega, with a 'Negativa' result and notes 'dawybha feh h'. The second row shows a test performed on 17/03/2014 at 00:00 by Celestina Amador Gil, with notes 'ffff'. The third row shows a test performed on 17/05/2016 at 00:00. Below the table, there is a pagination control showing '1' of '10' rows per page. A blue 'Grabar' button is located at the bottom left. A red box labeled 'Historial ergometrías' points to the table. Three callouts highlight specific buttons: 'Botón añadir' (orange), 'Botón imprimir informe PDF' (grey), and 'Botón editar' (green).

Captura 193: Captura de pantalla de la sección Pruebas médicas, apartado Ergometrías.

Captura 195: Captura de pantalla del formulario editar Ergometría.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *Ergometria.java* que se ha escrito para este apartado, es el siguiente:

```

1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8 import org.openxava.calculators.*;
9
10 //librería para trabajar con fechas
11 import java.util.Date;
12
13 @Entity
14 @View(members= //disponemos los parámetros para las vistas
15 "Ergometria [#" +
16 "fechaRealizacion;" +
17 "medico, protocolo, ritmoBasal;" +
18 "fcBasal, fcMaxima, porcentaje;" +
19 "taBasal, taMaxima;" +
20 "tem, tes, mets;" +

```

```

21 "resultado;" +
22 "notas;" +
23 "]"
24 )
25 @Table(name = "ergometria") //se referencia a la tabla de la BB.DD. llamada ergometria
26 public class Ergometria extends Identificable{ //hereda de la clase
27 //Identificable, para tener un identificador universal único, que se autogenera con funciones
28
29 //PROPIEDADES
30 @DefaultValueCalculator(CurrentDateCalculator.class) //calculador para darle al campo
31 //por defecto la fecha actual
32 @Column(name="FECHAREALIZACION")
33 @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
34 @Required // Se mostrará un error de validación si la propiedad se deja en blanco
35 private Date fechaRealizacion;
36
37 @Column (name="MEDICO")

```

...se omiten los nombres de los médicos...

```

80
81 private String nombreMedico;
82
83 private Medico (String nombreMedico){
84     this.nombreMedico = nombreMedico;
85 }
86
87 public String getNombreMedico() {
88     return nombreMedico;
89 }
90
91
92
93 //PARÁMETROS DE LA PRUEBA
94 @Column(name="PROTOCOLO")
95 private Protocolo protocolo;
96 public enum Protocolo //se describe un tipo enumerado que se visualiza como un SELECT
97 {
98     BRUCE ("Bruce"), MODIFICADO ("Modificado"), MANUAL("Manual");
99
100 private String nombreProtocolo;

```

```

101
102 private Protocolo (String nombreProtocolo){
103     this.nombreProtocolo = nombreProtocolo;
104 }
105
106 public String getNombreProtocolo() {
107     return nombreProtocolo;
108 }
109
110
111 @Column(name="RITMOBASAL")
112 private RitmoBasal ritmoBasal;
113 public enum RitmoBasal //se describe un tipo enumerado que se visualiza como un SELECT
114 {
115     SINUSAL ("Sinusal"), FA ("Fa"), BETABLOQUEANTE ("Beta-Bloqueante");
116
117 private String nombreRitmoBasal;
118
119 private RitmoBasal (String nombreRitmoBasal){
120     this.nombreRitmoBasal = nombreRitmoBasal;

```

```

121
122
123 public String getRitmoBasal() {
124     return nombreRitmoBasal;
125 }
126
127
128 @Stereotype("LPM") //Anotación Openxava para añadir unidad de medida lpm.
129 @Column(name="FCBASAL", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
130 private int fcBasal;
131
132 @Stereotype("LPM") //Anotación Openxava para añadir unidad de medida lpm.
133 @Column(name="FCMAXIMA", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
134 private int fcMaxima;
135
136 @Stereotype("PORCENTAJE") //Anotación Openxava para añadir unidad de medida %.
137 @Column(name="PORCENTAJE", length=3) // La longitud de columna se usa a nivel UI y a nivel DB
138 private int porcentaje;
139
140 @Stereotype("MMHG") //Anotación Openxava para añadir unidad de medida mmhg.

```

```

141 @Column(name="TABASAL", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
142 private int taBasal;
143
144 @Stereotype("MMHG") //Anotación Openxava para añadir unidad de medida mmhg.
145 @Column(name="TAMAXIMA", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
146 private int taMaxima;
147
148 @Stereotype("MINUTOS") //Anotación Openxava para añadir unidad de medida min.
149 @Column(name="TEM", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
150 private int tem;
151
152 @Stereotype("SEGUNDOS") //Anotación Openxava para añadir unidad de medida seg.
153 @Column(name="TES", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
154 private int tes;
155
156 @Column(name="METS", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
157 private int mets;
158
159 @Column(name="RESULTADO")
160 private Resultado resultado;

```

```

161 public enum Resultado //se describe un tipo enumerado que se visualiza como un SELECT
162 {
163     POSITIVA ("Positiva"), NEGATIVA ("Negativa"), NOCONCLUYENTE ("No Concluyente");
164
165     private String nombreResultado;
166
167     private Resultado (String nombreResultado){
168         this.nombreResultado = nombreResultado;
169     }
170
171     public String getNombreResultado() {
172         return nombreResultado;
173     }
174 }
175
176 @Column(name="NOTAS")
177 @Stereotype("HTML_TEXT") // Estereotipo que permite visualizar el editor de texto
178 private String notas;
179
180

```

```

181 //REFERENCIAS
182 @ManyToOne //Referencia que no permite Ecocardiografía TT sin Paciente
183 @JoinColumn( //se define la clave foránea de la tabla
184             name="numeroNhc", //columna para la clave foránea
185             nullable = true,
186             foreignKey = @ForeignKey(name = "fk_ergometria_numeronhc"))
187 //fk_ergometria_numeronhc es la la clave foránea en la BB.DD de cardiología
188 private Paciente ergometriasLocal; // Una referencia Java convencional
189
190 //MÉTODOS
191 public Date getFechaRealizacion() {
192     return fechaRealizacion;
193 }
194
195 public void setFechaRealizacion(Date fechaRealizacion) {
196     this.fechaRealizacion = fechaRealizacion;
197 }
198
199 public Medico getMedico() {
200     return medico;

```

```

201 }
202
203 public void setMedico(Medico medico) {
204     this.medico = medico;
205 }
206
207 public Protocolo getProtocolo() {
208     return protocolo;
209 }
210
211 public void setProtocolo(Protocolo protocolo) {
212     this.protocolo = protocolo;
213 }
214
215 public RitmoBasal getRitmoBasal() {
216     return ritmoBasal;
217 }
218
219 public void setRitmoBasal(RitmoBasal ritmoBasal) {
220     this.ritmoBasal = ritmoBasal;

```

```

222
223     public int getFcBasal() {
224         return fcBasal;
225     }
226
227     public void setFcBasal(int fcBasal) {
228         this.fcBasal = fcBasal;
229     }
230
231     public int getFcMaxima() {
232         return fcMaxima;
233     }
234
235     public void setFcMaxima(int fcMaxima) {
236         this.fcMaxima = fcMaxima;
237     }
238
239     public int getPorcentaje() {
240         return porcentaje;
241
242     }
243     public void setPorcentaje(int porcentaje) {
244         this.porcentaje = porcentaje;
245     }
246
247     public int getTaBasal() {
248         return taBasal;
249     }
250
251     public void setTaBasal(int taBasal) {
252         this.taBasal = taBasal;
253     }
254
255     public int getTaMaxima() {
256         return taMaxima;
257     }
258
259     public void setTaMaxima(int taMaxima) {
260         this.taMaxima = taMaxima;
261
262     }
263     public int getTem() {
264         return tem;
265     }
266
267     public void setTem(int tem) {
268         this.tem = tem;
269     }
270
271     public int getTes() {
272         return tes;
273     }
274
275     public void setTes(int tes) {
276         this.tes = tes;
277     }
278
279     public int getMets() {
280         return mets;
281
282     }
283     public void setMets(int mets) {
284         this.mets = mets;
285     }
286
287     public Resultado getResultado() {
288         return resultado;
289     }
290
291     public void setResultado(Resultado resultado) {
292         this.resultado = resultado;
293     }
294
295     public String getNotas() {
296         return notas;
297     }
298
299     public void setNotas(String notas) {
300         this.notas = notas;

```

```

301     }
302
303     //Métodos para el historial de ergometrias
304     public Paciente getErgometriasLocal() {
305         return ergometriasLocal;
306     }
307
308     public void setErgometriasLocal(Paciente ergometriasLocal) {
309         this.ergometriasLocal = ergometriasLocal;
310     }
311 }
312

```

Captura 196: Capturas de pantalla del código de la clase *Ergometria.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *Ergometria.java*. Lo indicamos con el siguiente código:

```

461     //se deshabilita la opción borrar desde el formulario de ergometría
462     @RemoveAction("")
463     //se deshabilita la opción borrar desde la lista de ergometrias
464     @RemoveSelectedAction("")
465     //si se borra paciente, se
466     //borrarán sus pruebas médicas de ecocardiografías transtorácicas
467     @OneToMany (mappedBy="ergometriasLocal", cascade=CascadeType.ALL)
468     private Collection<Ergometria> ergometria = new ArrayList<Ergometria>();

```

Captura 197: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.7.4.1. Estereotipos necesarios para la clase.

Por último, definimos los estereotipos que faltan lpm, mmHg, min, y seg:

- Fichero *editores.xml*:

```

--
32 <editor nombre="lpm" url="lpmEditor.jsp">
33     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
34     <para-estereotipo estereotipo="LPM"/>
35 </editor>
36
37 <editor nombre="mmhg" url="mmhgEditor.jsp">
38     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
39     <para-estereotipo estereotipo="MMHG"/>
40 </editor>
41
42 <editor nombre="minutos" url="minutosEditor.jsp">
43     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
44     <para-estereotipo estereotipo="MINUTOS"/>
45 </editor>
46
47 <editor nombre="segundos" url="segundosEditor.jsp">
48     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
49     <para-estereotipo estereotipo="SEGUNDOS"/>
50 </editor>
51

```

Captura 198: Captura de pantalla de fragmento de código del script *editores.xml*.

- Clase formateadora (sólo se ha de crear una sólo vez):

```

1 package org.openxava.cardiologia.formateadores;
2
3 import java.math.*;
4
5
6
7
8
9 public class PesoFormateador implements IFormatter {
10
11     public String format(HttpServletRequest request, Object object) throws Exception {
12         if (Is.empty(object)) return "";
13         return getFormat().format(object);
14     }
15
16     public Object parse(HttpServletRequest request, String string) throws Exception {
17         if (Is.emptyString(string)) return null;
18         string = Strings.change(string, " ", ""); // In order to work with Polish
19         return new BigDecimal(getFormat().parse(string).toString()).setScale(2);
20     }
21
22     private NumberFormat getFormat() {
23         NumberFormat f = DecimalFormat.getNumberInstance(Locales.getCurrent());
24         f.setMinimumFractionDigits(2);
25         f.setMaximumFractionDigits(2);
26         return f;
27     }
28
29 }
30

```

Captura 199: Captura de pantalla de la clase formateadora *PesoFormateador*.

- Scripts de los editores:

```

1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7     symbol = "lpm"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10     symbol = "?";
11 }
12 %>
13
14 <%=symbol%>

```

Captura 200: Captura de pantalla del script *lpmEditor.jsp*


```

1 |<%@page import="java.util.Currency"%>
2 |<%@page import="java.util.Locale"%>
3 |<%@ include file="textEditor.jsp"%>
4 |
5 |<% String symbol = null;
6 | try {
7 | symbol = "mmHg"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 | }
9 | catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 | symbol = "?";
11 | }
12 | %>
13 |
14 | <%=symbol%>

```

Captura 201: Captura de pantalla del script *mmhgEditor.jsp*.

```

1 |<%@page import="java.util.Currency"%>
2 |<%@page import="java.util.Locale"%>
3 |<%@ include file="textEditor.jsp"%>
4 |
5 |<% String symbol = null;
6 | try {
7 | symbol = "min"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 | }
9 | catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 | symbol = "?";
11 | }
12 | %>
13 |
14 | <%=symbol%>

```

Captura 202: Captura de pantalla del script *minutosEditor.jsp*.

```

1 |<%@page import="java.util.Currency"%>
2 |<%@page import="java.util.Locale"%>
3 |<%@ include file="textEditor.jsp"%>
4 |
5 |<% String symbol = null;
6 | try {
7 | symbol = "seg"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 | }
9 | catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 | symbol = "?";
11 | }
12 | %>
13 |
14 | <%=symbol%>

```

Captura 203: Captura de pantalla del script *segundosEditor.jsp*.

Por último, añadimos las etiquetas para la internacionalización de los componentes del apartado Ergometrías:

```
110 #Componentes Ergometría
111 BRUCE=Bruce
112 MODIFICADO=Modificado
113 MANUAL=Manual
114 SINUSAL=Sinusal
115 FA=FA
116 BETABLOQUEANTE=Beta-Bloqueante
117 POSITIVA=Positiva
118 NEGATIVA=Negativa
119 NOCONCLUYENTE=No Concluyente
120 fechaRealizacion=Fecha de realización
121 protocolo=Protocolo
122 ritmoBasal= Ritmo Basal
123 fcBasal=FC Basal
124 fcMaxima=FC Máxima
125 porcentaje=Porcentaje
126 taBasal=TA Basal
127 taMaxima=TA Máxima
128 tem=Tiempo de ejercicio
129 tes=
130 mets=METS
131 resultado=Resultado
132 notas=Notas
```

Captura 204: Captura de pantalla de fragmento del script *EtiquetasCardiologias_es.properties*.

5.7.5 Estudios electrofisiológicos.

El usuario puede grabar estudios electrofisiológicos del paciente, visualizar el historial de éstos, los datos de cada uno, y también modificarlos.

Para registrar un estudio electrofisiológico (EEF) en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Si desea imprimir el informe *pdf*, puede hacerlo a través de un botón.



Captura 205: Captura de pantalla de la sección Pruebas médicas, apartado Estudios electrofisiológicos.

Crea una nueva entidad - Estudio eletrofisiológico

Fecha de realización 08/05/2017 19:53

Médico

▼ Informe

Fuente HTML

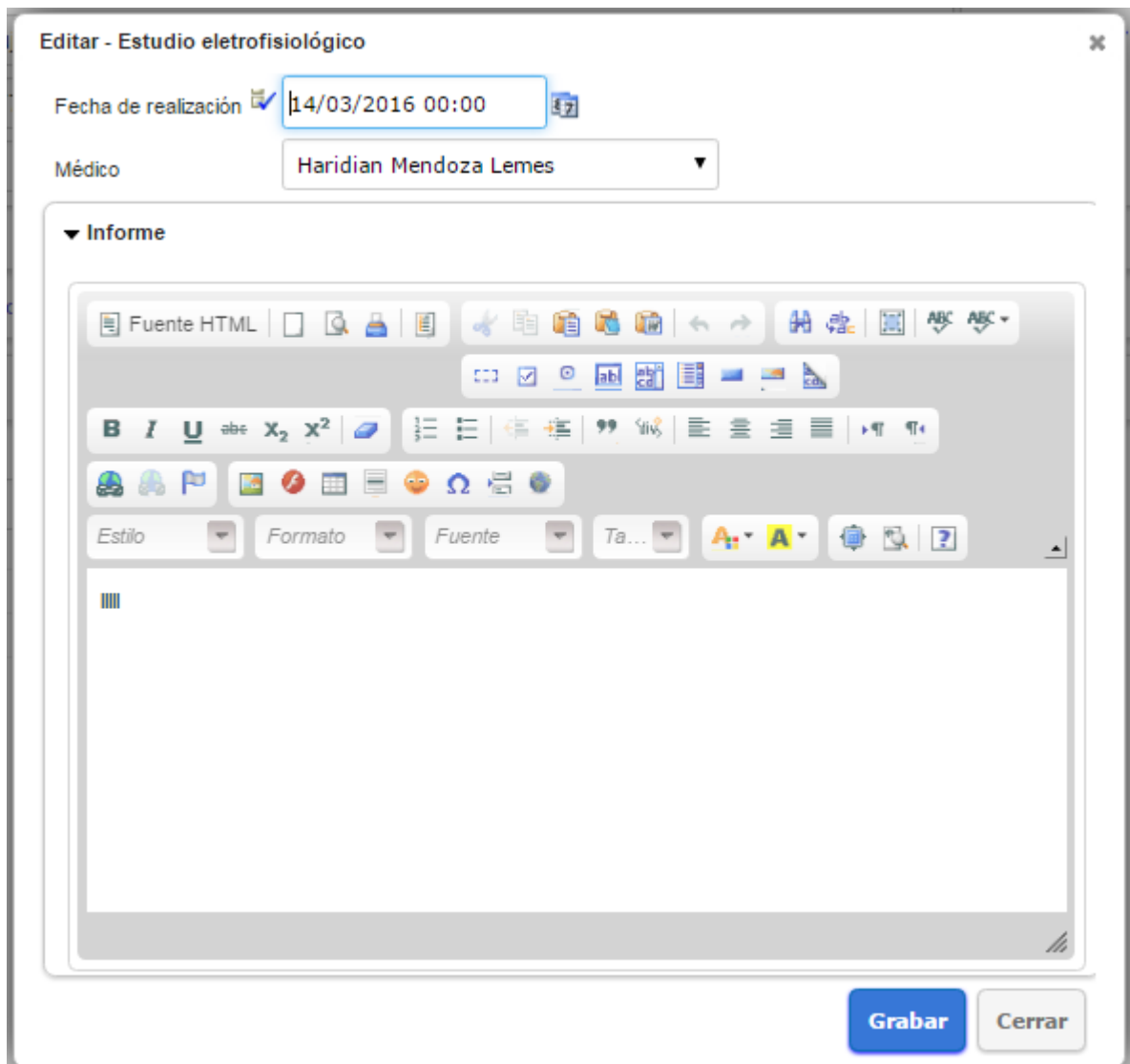
B *I* U abc x_2 x^2

Estilo Formato Fuente Ta...

Grabar Grabar y continuar Cerrar

Captura 206: Captura de pantalla del formulario crear EEF.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.



Captura 207: Captura de pantalla del formulario editar EEF.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *EstudioElectrofisiologico.java* que se ha escrito para este apartado, es el siguiente:

```

estudioelectrofisiologico.java x
1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8 import org.openxava.calculators
9
10 //librería para trabajar con fechas
11 import java.util.Date;
12
13 @Entity
14 @Table(name = "estudioelectrofisiologico") //se referencia a la tabla de la BB.DD.
15 //llamada estudioelectrofisiologico
16 public class EstudioElectrofisiologico extends Identificable{ //hereda de la clase Identificable,
17 //para tener un identificador universal único, que se autogenera con funciones
18
19 //PROPIEDADES
20 @DefaultValueCalculator(CurrentDateCalculator.class) //calculador para darle al campo
21
22 //por defecto la fecha actual
23 @Column(name="FECHAREALIZACION")
24 @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
25 @Required // Se mostrará un error de validación si la propiedad se deja en blanco
26 private Date fechaRealizacion;
27
28 @Column (name="MEDICO")
29 private Medico medico;
30 public enum Medico { //se describe un tipo enumerado que se visualiza como un SELECT

```

... se omiten los nombres de los médicos...

```

81
82 @Column(name="INFORME", length = 250, nullable = true)
83 @Stereotype("HTML_TEXT") // Estereotipo que permite visualizar el editor de texto
84 private String informe;
85
86
87 //REFERENCIA
88 @ManyToOne //Referencia que no permite EEF sin Paciente
89 @JoinColumn( //se define la clave foránea de la tabla
90 name="numeroNhc", //columna para la clave foránea
91 nullable = true,
92 foreignKey = @ForeignKey(name = "fk_eef_numeronhc"))
93 //fk_eef_numeronhc es la la clave foránea en la BB.DD de cardiologia
94 private Paciente eefsLocal; // Una referencia Java convencional
95
96 //MÉTODOS
97 public Date getFechaRealizacion() {
98     return fechaRealizacion;
99 }
100

```

```

100
101 public void setFechaRealizacion(Date fechaRealizacion) {
102     this.fechaRealizacion = fechaRealizacion;
103 }
104
105 public Medico getMedico() {
106     return medico;
107 }
108
109 public void setMedico(Medico medico) {
110     this.medico = medico;
111 }
112
113 public String getInforme() {
114     return informe;
115 }
116
117 public void setInforme(String informe) {
118     this.informe = informe;
119 }
120

```

Capturas 208: Capturas de pantalla del código de la clase *EstudioElectrofisiologico.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *EstudioElectrofisiologico.java*. Lo indicamos con el siguiente código:

```
470 //se deshabilita la opción borrar desde el formulario de eef
471 @RemoveAction("")
472 //se deshabilita la opción borrar desde la lista de eef's
473 @RemoveSelectedAction("")
474 //si se borra paciente, se
475 //borrarán sus pruebas médicas de estudios electrofisiológicos
476 @OneToMany (mappedBy="eefsLocal", cascade=CascadeType.ALL)
477 //para poder visualizar la lista de pruebas ecocardiografias transtorácicas
478 private Collection<EstudioElectrofisiologico> estudioElectrofisiologico =
479     new ArrayList<EstudioElectrofisiologico>();
480
```

Captura 209: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

Por último, añadimos las etiquetas para la internacionalización de los componentes del apartado Estudios electrofisiológicos:

```
121 //Métodos para el historial de eef
122 publ. 134 #Componentes de Estudio Electrofisiológico
123     135 EstudioElectrofisiologico=Estudio eletrofisiológico
124     }
125     136
126 public void setEefsLocal(Paciente eefsLocal) {
127     this.eefsLocal = eefsLocal;
128 }
129 }
130
```

Captura 210: Captura de pantalla de fragmento del script *EtiquetasCardiologias_es.properties*.

5.7.6 Hipertensión pulmonar.

El usuario puede grabar pruebas de hipertensión pulmonar (HTP) del paciente, visualizar el historial de éstas, los datos de cada una, y también modificarlas.

Para registrar una prueba de hipertensión pulmonar en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Si desea imprimir el informe *pdf*, puede hacerlo a través de un botón.



Captura 211: Captura de pantalla de la sección Pruebas médicas, apartado Hipertensión pulmonar.

Crea una nueva entidad - Hipertensión pulmonar

▼ Hipertensión pulmonar

Fecha de realización 08/05/2017 20:22

Médico

AI	<input type="text"/>	mm	Raíz Ao	<input type="text"/>	mm
DTDVI	<input type="text"/>	mm	DTSVI	<input type="text"/>	mm
TIV	<input type="text"/>	mm	PP	<input type="text"/>	mm
E-TIV	<input type="text"/>	mm	DTDVD	<input type="text"/>	mm
Teicholtz	<input type="text"/>	%	Arteria Pulmonar	<input type="text"/>	mm
Frecuencia Cardíaca	<input type="text"/>	lpm	Tensión Arterial	<input type="text"/>	mmHg
T1	<input type="text"/>		T2	<input type="text"/>	
Área AD	<input type="text"/>	cm2	D1	<input type="text"/>	mm
Área VD	<input type="text"/>	cm2	D2	<input type="text"/>	mm
TAPSE	<input type="text"/>	mm	Índice de excentr VI	<input type="text"/>	
GP	<input type="text"/>		Vel E	<input type="text"/>	
dP/dt	<input type="text"/>		Vel A	<input type="text"/>	
G1	<input type="text"/>		GP	<input type="text"/>	
G2	<input type="text"/>		GM	<input type="text"/>	

Captura 212: Captura de pantalla del formulario crear HTP.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.

Editar - Hipertensión pulmonar

▼ Hipertensión pulmonar

Fecha de realización 16/05/2016 00:00

Médico

AI	<input type="text" value="65"/> mm	Raíz Ao	<input type="text"/> mm
DTDVI	<input type="text"/> mm	DTSVI	<input type="text"/> mm
TIV	<input type="text" value="34"/> mm	PP	<input type="text"/> mm
E-TIV	<input type="text"/> mm	DTDVD	<input type="text"/> mm
Teicholtz	<input type="text" value="5"/> %	Arteria Pulmonar	<input type="text"/> mm
Frecuencia Cardíaca	<input type="text"/> lpm	Tensión Arterial	<input type="text"/> mmHg
T1	<input type="text" value="0"/>	T2	<input type="text" value="0"/>
Área AD	<input type="text"/> cm2	D1	<input type="text" value="53"/> mm
Área VD	<input type="text" value="5"/> cm2	D2	<input type="text" value="2"/> mm
TAPSE	<input type="text" value="4"/> mm	Índice de excentr VI	<input type="text" value="0"/>
GP	<input type="text" value="0"/>	Vel E	<input type="text" value="0"/>
dP/dt	<input type="text" value="0"/>	Vel A	<input type="text" value="0"/>
G1	<input type="text" value="0"/>	GP	<input type="text" value="0"/>
G2	<input type="text" value="0"/>	GM	<input type="text" value="0"/>

Captura 213: Captura de pantalla del formulario editar HTP.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *HipertensionPulmonar.java* que se ha escrito para este apartado, es el siguiente:

```

hipertensionpulmonar.java
1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8 import org.openxava.calculators.*;
9
10 //librería para trabajar con fechas
11 import java.util.*;
12
13
14 @Entity
15 @View(members= //disposición de los parámetros de la prueba
16 "HipertensionPulmonar [#" +
17 "fechaRealizacion;" +
18 "medico;" +
19 "ai, raizAo;" +
20 "dtdvi, dtsvi;" +
21
22 "tiv, pp;" +
23 "etiv, dtdvd;" +
24 "teicholtz, arteriaPulmonar;" +
25 "frecuenciaCardiaca, tensionArterial;" +
26 "t1, t2;" +
27 "areaAd, d1;" +
28 "areaVd, d2;" +
29 "tapse, indiceExcentrVi;" +
30 "gp, velE;" +
31 "dpdt, velA;" +
32 "g1, gp2;" +
33 "g2, gm;" +
34 "]"
35 )
36 @Table(name = "hipertensionpulmonar") //se referencia a la tabla de la BB.DD.
37 //llamada hipertensionpulmonar
38 public class HipertensionPulmonar extends Identificable{ //hereda de la clase Identificable,
39 //para tener un identificador universal único, que se autogenera con funciones
40
41 //PROPIEDADES

```

...se omiten los nombres de los médicos...

```

101 }
102
103 //parámetros de la prueba
104 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
105 @Column(name="AI", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
106 private int ai;
107
108 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
109 @Column(name="DTDVI", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
110 private int dtdvi;
111
112 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
113 @Column(name="TIV", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
114 private int tiv;
115
116 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
117 @Column(name="ETIV", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
118 private int etiv;
119
120 @Stereotype("PORCENTAJE") //Anotación Openxava para añadir unidad de medida %.
121
122 @Column(name="TEICHOLTZ", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
123 private int teicholtz;
124
125 @Stereotype("LPM") //Anotación Openxava para añadir unidad de medida lpm.
126 @Column(name="FRECUENCIACARDIACA", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
127 private int frecuenciaCardiaca;
128
129 @Column(name="T1", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
130 private int t1;
131
132 @Stereotype("CM2") //Anotación Openxava para añadir unidad de medida cm2.
133 @Column(name="AREAAD", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
134 private int areaAd;
135
136 @Stereotype("CM2") //Anotación Openxava para añadir unidad de medida cm2.
137 @Column(name="AREAVD", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
138 private int areaVd;
139
140 @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
141 @Column(name="TAPSE", length=7) // La longitud de columna se usa a nivel UI y a nivel DB

```

```

141     private int tapse;
142
143     @Column(name="GP", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
144     private int gp;
145
146     @Column(name="DPDT", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
147     private int dpdt;
148
149     @Column(name="G1", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
150     private int g1;
151
152     @Column(name="GP2", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
153     private int gp2;
154
155     @Column(name="G2", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
156     private int g2;
157
158     @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
159     @Column(name="RAIZAO", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
160     private int raizaO;

```

```

161
162     @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
163     @Column(name="DTSVI", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
164     private int dtsvi;
165
166     @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
167     @Column(name="PP", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
168     private int pp;
169
170     @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
171     @Column(name="DTDVD", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
172     private int dtdvd;
173
174     @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
175     @Column(name="ARTERIA PULMONAR", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
176     private int arteriaPulmonar;
177
178     @Stereotype("MMHG") //Anotación Openxava para añadir unidad de medida mmHg.
179     @Column(name="TENSIONARTERIAL", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
180     private int tensionArterial;

```

```

181
182     @Column(name="T2", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
183     private int t2;
184
185     @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
186     @Column(name="D1", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
187     private int d1;
188
189     @Stereotype("MILIMETROS") //Anotación Openxava para añadir unidad de medida mm.
190     @Column(name="D2", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
191     private int d2;
192
193     @Column(name="INDICEEXCENTRVI", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
194     private int indiceExcentrVi;
195
196     @Column(name="VELE", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
197     private int vele;
198
199     @Column(name="VELA", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
200     private int vela;

```

```

201
202
203     @Column(name="GM", length=7) // La longitud de columna se usa a nivel UI y a nivel DB
204     private int gm;
205
206
207     //REFERENCIA
208     @ManyToOne //Referencia que no permite Ecocardiografía IT sin Paciente
209     @JoinColumn( //se define la clave foránea de la tabla
210         name="numeroNhc", //columna para la clave foránea
211         nullable = true,
212         foreignKey = @ForeignKey(name = "fk_hipertensionpulmonar_numeronhc"))
213         //fk_hipertensionpulmonar_numeronhc es la la clave foránea en la BB.DD de cardiología
214     private Paciente hipertensionespulmonaresLocal; // Una referencia Java convencional
215
216     //MÉTODOS
217     public Date getFechaRealizacion() {
218         return fechaRealizacion;
219     }
220

```

```

221 public void setFechaRealizacion(Date fechaRealizacion) {
222     this.fechaRealizacion = fechaRealizacion;
223 }
224
225 public Medico getMedico() {
226     return medico;
227 }
228
229 public void setMedico(Medico medico) {
230     this.medico = medico;
231 }
232
233 public int getAi() {
234     return ai;
235 }
236
237 public void setAi(int ai) {
238     this.ai = ai;
239 }
240

```

```

241 public int getDtdvi() {
242     return dtdvi;
243 }
244
245 public void setDtdvi(int dtdvi) {
246     this.dtdvi = dtdvi;
247 }
248
249 public int getTiv() {
250     return tiv;
251 }
252
253 public void setTiv(int tiv) {
254     this.tiv = tiv;
255 }
256
257 public int getEtiv() {
258     return etiv;
259 }
260

```

```

261 public void setEtiv(int etiv) {
262     this.etiv = etiv;
263 }
264
265 public int getTeicholtz() {
266     return teicholtz;
267 }
268
269 public void setTeicholtz(int teicholtz) {
270     this.teicholtz = teicholtz;
271 }
272
273 public int getFrecuenciaCardiaca() {
274     return frecuenciaCardiaca;
275 }
276
277 public void setFrecuenciaCardiaca(int frecuenciaCardiaca) {
278     this.frecuenciaCardiaca = frecuenciaCardiaca;
279 }
280

```

```

281 public int getT1() {
282     return t1;
283 }
284
285 public void setT1(int t1) {
286     this.t1 = t1;
287 }
288
289 public int getAreaAd() {
290     return areaAd;
291 }
292
293 public void setAreaAd(int areaAd) {
294     this.areaAd = areaAd;
295 }
296
297 public int getAreaVd() {
298     return areaVd;
299 }
300

```

...

```

321 public int getDpdt() {
322     return dpdt;
323 }
324
325 public void setDpdt(int dpdt) {
326     this.dpdt = dpdt;
327 }
328
329 public int getG1() {
330     return g1;
331 }
332
333 public void setG1(int g1) {
334     this.g1 = g1;
335 }
336
337 public int getGp2() {
338     return gp2;
339 }
340

```

```

341 public void setGp2(int gp2) {
342     this.gp2 = gp2;
343 }
344
345 public int getG2() {
346     return g2;
347 }
348
349 public void setG2(int g2) {
350     this.g2 = g2;
351 }
352
353 public int getRaizAo() {
354     return raizAo;
355 }
356
357 public void setRaizAo(int raizAo) {
358     this.raizAo = raizAo;
359 }
360

```

...

```

381 public void setDtdvd(int dtdvd) {
382     this.dtdvd = dtdvd;
383 }
384
385 public int getArteriaPulmonar() {
386     return arteriaPulmonar;
387 }
388
389 public void setArteriaPulmonar(int arteriaPulmonar) {
390     this.arteriaPulmonar = arteriaPulmonar;
391 }
392
393 public int getTensionArterial() {
394     return tensionArterial;
395 }
396
397 public void setTensionArterial(int tensionArterial) {
398     this.tensionArterial = tensionArterial;
399 }
400

```

```

401 public int getT2() {
402     return t2;
403 }
404
405 public void setT2(int t2) {
406     this.t2 = t2;
407 }
408
409 public int getD1() {
410     return d1;
411 }
412
413 public void setD1(int d1) {
414     this.d1 = d1;
415 }
416
417 public int getD2() {
418     return d2;
419 }
420

```

```

421     public void setD2(int d2) {
422         this.d2 = d2;
423     }
424
425     public int getIndiceExcentrVi() {
426         return indiceExcentrVi;
427     }
428
429     public void setIndiceExcentrVi(int indiceExcentrVi) {
430         this.indiceExcentrVi = indiceExcentrVi;
431     }
432
433     public int getVelE() {
434         return velE;
435     }
436
437     public void setVelE(int velE) {
438         this.velE = velE;
439     }
440
441     public int getVelA() {
442         return velA;
443     }
444
445     public void setVelA(int velA) {
446         this.velA = velA;
447     }
448
449     public int getGm() {
450         return gm;
451     }
452
453     public void setGm(int gm) {
454         this.gm = gm;
455     }
456
457     //Métodos para el historial de HTP
458     public Paciente getHipertensionespulmonaresLocal() {
459         return hipertensionespulmonaresLocal;
460     }
461
462     public void setHipertensionespulmonaresLocal(Paciente hipertensionespulmonaresLocal) {
463         this.hipertensionespulmonaresLocal = hipertensionespulmonaresLocal;
464     }
465
466 }

```

Capturas 214: Capturas de pantalla del código de la clase *HipertensionPulmonar.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *HipertensionPulmonar.java*. Lo indicamos con el siguiente código:

```

481     //se deshabilita la opción borrar desde el formulario de htp
482     @RemoveAction("")
483     //se deshabilita la opción borrar desde la lista de htp's
484     @RemoveSelectedAction("")
485     //si se borra paciente, se
486     //borrarán sus pruebas médicas de hipertensión pulmonar
487     @OneToMany (mappedBy="hipertensionespulmonaresLocal", cascade=CascadeType.ALL)
488     //para poder visualizar la lista de pruebas hipertensión pulmonar
489     private Collection<HipertensionPulmonar> hipertensionPulmonar =
490     new ArrayList<HipertensionPulmonar>();
491

```

Captura 215: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.7.6.1. Estereotipos para la clase.

Por último, definimos el estereotipo cm2:

- Fichero *editores.xml*:

```

52 <editor nombre="cm2" url="cm2Editor.jsp">
53     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
54     <para-estereotipo estereotipo="CM2"/>
55 </editor>

```

Captura 216: Captura de pantalla de fragmento de código del script *editores.xml*.

- Clase formateadora (sólo se ha de crear una sólo vez):

```

1 package org.openxava.cardiologia.formateadores;
2
3 import java.math.*;
4
5
6
7
8
9 public class PesoFormateador implements IFormatter {
10
11     public String format(HttpServletRequest request, Object object) throws Exception {
12         if (Is.empty(object)) return "";
13         return getFormat().format(object);
14     }
15
16     public Object parse(HttpServletRequest request, String string) throws Exception {
17         if (Is.emptyString(string)) return null;
18         string = Strings.change(string, " ", ""); // In order to work with Polish
19         return new BigDecimal(getFormat().parse(string).toString()).setScale(2);
20     }
21
22     private NumberFormat getFormat() {
23         NumberFormat f = DecimalFormat.getNumberInstance(Locale.getCurrent());
24         f.setMinimumFractionDigits(2);
25         f.setMaximumFractionDigits(2);
26         return f;
27     }
28
29 }
30

```

Captura 217: Captura de pantalla de la clase formateadora *PesoFormateador*.

- Scripts del editor:

```

1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7 symbol = "cm2"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 symbol = "?";
11 }
12 %>
13
14 <%=symbol%>

```

Captura 218: Captura de pantalla del script *cm2Editor.jsp*.

5.7.6.2 Internacionalización para la clase.

Además, añadimos las etiquetas para la internacionalización de los componentes del apartado Hipertensión pulmonar:

```
137 #Componentes Hipertensión Pulmonar
138 HipertensionPulmonar=Hipertensión pulmonar
139 etiv=E-TIV
140 frecuenciaCardiaca= Frecuencia Cardíaca
141 t1=T1
142 areaAd=Área AD
143 areaVd=Área VD
144 tapse=TAPSE
145 gp=GP
146 dpdt=dP/dt
147 g1=G1
148 g2=G2
149 arteriaPulmonar=Arteria Pulmonar
150 tensionArterial=Tensión Arterial
151 t2=T2
152 d1=D1
153 d2=D2
154 indiceExcentrVi=Índice de excentr VI
155 velE=Vel E
156 VelA=Vel A
157 gm=GM
158 gp2=GP
```

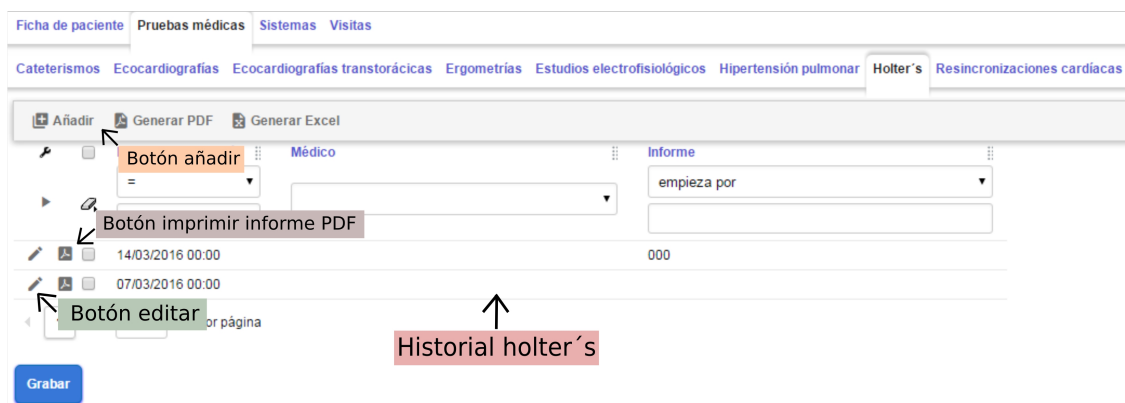
Captura 219: Captura de pantalla de fragmento del script *EtiquetasCardiologias_es.properties*.

5.7.7 Holter's.

El usuario puede grabar pruebas de holter del paciente, visualizar el historial de éstos, los datos de cada uno, y también modificarlos.

Para registrar una prueba de holter en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Si desea imprimir el informe *pdf*, puede hacerlo a través de un botón.



Captura 220: Captura de pantalla de la sección Pruebas médicas, apartado *Holter's*.

Crea una nueva entidad - Holter

Fecha de realización 08/05/2017 21:25

Médico

▼ Informe

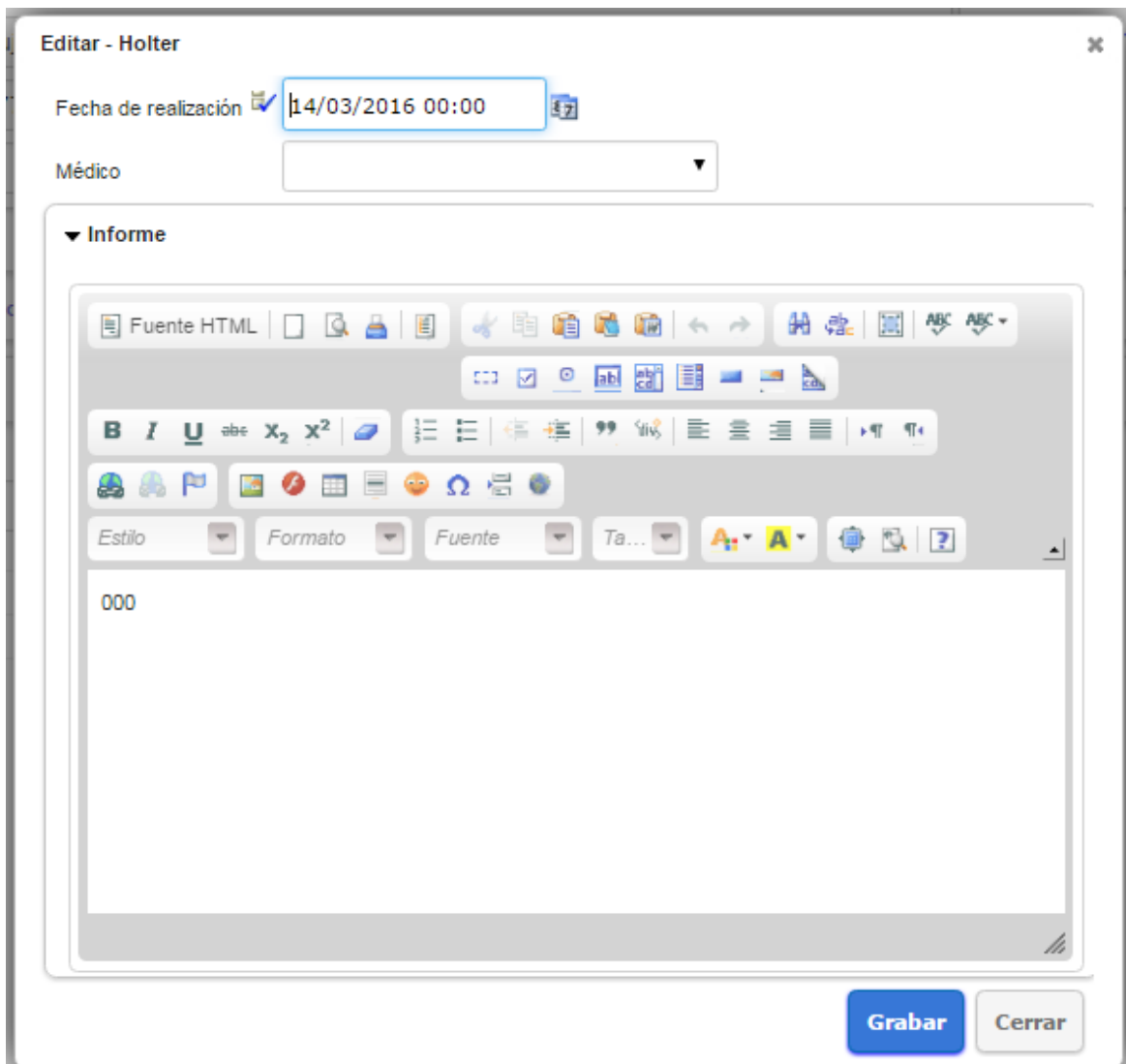
Fuente HTML

Estilo Formato Fuente Ta...

Grabar Grabar y continuar Cerrar

Captura 221: Captura de pantalla del formulario crear *Holter*.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.



Captura 222: Captura de pantalla del formulario editar *Holter*.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *Holter.java* que se ha escrito para este apartado, es el siguiente:

```

holter.java x
1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8 import org.openxava.calculators.*;
9
10 //librería para trabajar con fechas
11 import java.util.Date;
12
13 @Entity
14 @Table(name = "holter") //se referencia a la tabla de la BB.DD. llamada holter
15 public class Holter extends Identificable{ //hereda de la clase Identificable, para
16 //tener un identificador universal único, que se autogenera con funciones
17
18 //PROPIEDADES
19 @DefaultValueCalculator(CurrentDateCalculator.class) //calculador para darle al campo
20 //por defecto la fecha actual

```

... se omiten los médicos...

```

81 @Column(name="INFORME", length = 250, nullable = true)
82 @Stereotype("HTML_TEXT") // Estereotipo que permite visualizar el editor de texto
83 private String informe;
84
85
86 //REFERENCIA
87 @ManyToOne //Referencia que no permite Holter sin Paciente
88 @JoinColumn( //se define la clave foránea de la tabla
89             name="numeroNhc", //columna para la clave foránea
90             nullable = true,
91             foreignKey = @ForeignKey(name = "fk_holter_numeronhc"))
92 //fk_holter_numeronhc es la la clave foránea en la BB.DD de cardiología
93 private Paciente holtersLocal; // Una referencia Java convencional
94
95
96 //MÉTODOS
97 public Date getFechaRealizacion() {
98     return fechaRealizacion;
99 }
100
101 public void setFechaRealizacion(Date fechaRealizacion) {
102     this.fechaRealizacion = fechaRealizacion;
103 }
104
105 public Medico getMedico() {
106     return medico;
107 }
108
109 public void setMedico(Medico medico) {
110     this.medico = medico;
111 }
112
113 public String getInforme() {
114     return informe;
115 }
116
117 public void setInforme(String informe) {
118     this.informe = informe;
119 }
120
121 //Métodos para el historial de holter's
122 public Paciente getHoldersLocal() {
123     return holtersLocal;
124 }
125
126 public void setHoldersLocal(Paciente holtersLocal) {
127     this.holdersLocal = holtersLocal;
128 }
129
130 }
131

```

Captura 223: Capturas de pantalla del código de la clase *Holter.java*.

En la clase `Paciente.java` también debemos indicar la relación 1 a * con la clase `Holter.java`. Lo indicamos con el siguiente código:

```
492 //se deshabilita la opción borrar desde el formulario de holter
493 @RemoveAction("")
494 //se deshabilita la opción borrar desde la lista de holter's
495 @RemoveSelectedAction("")
496 //si se borra paciente, se
497 //borrarán sus pruebas médicas de holter's
498 @OneToMany (mappedBy="holtersLocal", cascade=CascadeType.ALL)
499 //para poder visualizar la lista de pruebas médicas de holter's
500 private Collection<Holter> holter = new ArrayList<Holter>();
501
```

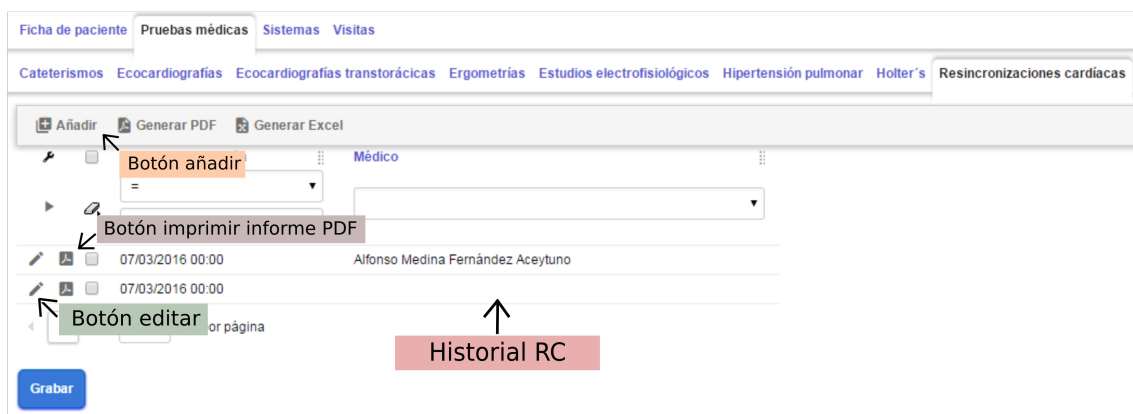
Captura 224: Captura de pantalla de fragmento de código de la clase `Paciente.java`.

5.7.8. Resincronización cardíaca.

El usuario puede grabar pruebas de resincronización cardíaca del paciente, visualizar el historial de éstas, los datos de cada una, y también modificarlos.

Para registrar una prueba de resincronización cardíaca en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

Si desea imprimir el informe *pdf*, puede hacerlo a través de un botón.



Captura 225: Captura de pantalla de la sección Pruebas médicas, apartado Resincronización cardíaca.

Captura 226: Captura de pantalla del formulario crear Resincronización cardíaca.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.

Editar - Resincronización cardíaca

▼ Resincronización cardíaca

Fecha de realización 07/03/2016 00:00

Médico

Retraso interventricular (Modo M)	<input type="text" value="8"/>	Tiempo preeyectivo aórtico	<input type="text" value="7"/>
Retraso interventricular (tipo preeyectivo pulmonar)	<input type="text" value="7"/>	Retraso entre el SIV y la PL	<input type="text" value="7"/>
Fin del acortamiento tras cierre valvular aórtico	<input type="text"/>	Movimiento en M del SIV por DTI color	<input type="text" value="7"/>
Apreciación subjetiva del movimiento asincrónico del SIV (4C)	<input type="text" value="7"/>	Tiempo de llenado diastólico con respecto al RR(4C)	<input type="text" value="7"/>
Insuficiencia mitral presistólica	<input type="text" value="7"/>	dP/dt(4C) < 800mmHg/s	<input type="text"/>
Insuficiencia mitral > grado II	<input type="text"/>		

Captura 227: Captura de pantalla del formulario editar Resincronización cardíaca.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *ResincronizaciónCardíaca.java* que se ha escrito para este apartado, es el siguiente:

```

1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8 import org.openxava.calculators.*;
9
10 //librería para trabajar con fechas
11 import java.util.Date;
12
13 @Entity
14 @View(members= //disposición de los parámetros de la prueba
15 "ResincronizacionCardiaca [#" +
16 "fechaRealizacion;" +
17 "medico;" +
18 "rimm, tpa;" +
19 "ritpp, rsp;" +
20 "fatcva, mmsdc;" +

```

```

21 "asmas, tldrrr;" +
22 "imp, dpdt4c;" +
23 "img2;" +
24 "]"
25 )
26 //se referencia a la tabla de la BB.DD. llamada resincronizacioncardiaca
27 @Table ( name = "resincronizacioncardiaca" )
28 public class ResincronizacionCardiaca extends Identificable{ //hereda de la clase Identificable, para
29 //tener un identificador universal único, que se autogenera con funciones
30
31 //PROPIEDADES
32 @DefaultValueCalculator(CurrentDateCalculator.class) //calculador para darle al campo
33 @Column(name="FECHAREALIZACION")
34 @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
35 @Required // Se mostrará un error de validación si la propiedad se deja en blanco
36 private Date fechaRealizacion;
37
38 @Column (name="MEDICO")
39 private Medico medico;
40 public enum Medico { //se describe un tipo enumerado que se visualiza como un SELECT

```

... se omiten los médicos ...

```

81
82 private String nombreMedico;
83
84 private Medico (String nombreMedico){
85     this.nombreMedico = nombreMedico;
86 }
87
88 public String getNombreMedico() {
89     return nombreMedico;
90 }
91 }
92
93 //retraso interventricular (Modo M)
94 @Column(name="RIMM", length=7)
95 private String rimm;
96
97 //retraso interventricular (tipo preeyectivo pulmonar)
98 @Column(name="RITPP", length=7)
99 private String ritpp;
100
101 //fin del acortamiento tras cierre valvular aórtico
102 @Column(name="FATCVA", length=7)
103 private String fatcva;
104
105 //apreciación subjetiva del movimiento asincrónico del SIV (4C)
106 @Column(name="ASMAS", length=7)
107 private String asmas;
108
109 //insuficiencia mitral presistólica
110 @Column(name="IMP", length=7)
111 private String imp;
112
113 //insuficiencia mitral > grado II
114 @Column(name="IMG2", length=7)
115 private String img2;
116
117 //tiempo preeyectivo aórtico
118 @Column(name="TPA", length=7)
119 private int tpa;
120

```

```

121 //retraso entre el SIV y la PL
122 @Column(name="RSP", length=7)
123 private int rsp;
124
125 //movimiento en M del SIV por DTI color
126 @Column(name="MMSDC", length=7)
127 private String mmsdc;
128
129 //tiempo de llenado diastólico con respecto al RR(4C)
130 @Column(name="TLDRRR", length=7)
131 private int tldr;
132
133 //dP/dt(4C) < 800mmHg/s
134 @Column(name="DPDT4C", length=7)
135 private String dpdt4c;
136
137
138 //REFERENCIA
139 @ManyToOne //Referencia que no permite Holter sin Paciente
140 @JoinColumn( //se define la clave foránea de la tabla

```

```

141         name="numeroNhc", //columna para la clave foránea
142         nullable = true,
143         foreignKey = @ForeignKey(name = "fk_resincro_numeronhc"))
144         //fk_resincro_numeronhc es la la clave foránea en la BB.DD de cardiología
145 private Paciente resincrosLocal; // Una referencia Java convencional
146
147
148 //MÉTODOS
149 public Date getFechaRealizacion() {
150     return fechaRealizacion;
151 }
152
153 public void setFechaRealizacion(Date fechaRealizacion) {
154     this.fechaRealizacion = fechaRealizacion;
155 }
156
157 public Medico getMedico() {
158     return medico;
159 }
160

```

```

161 public void setMedico(Medico medico) {
162     this.medico = medico;
163 }
164
165 public String getRimm() {
166     return rimm;
167 }
168
169 public void setRimm(String rimm) {
170     this.rimm = rimm;
171 }
172
173 public String getRitpp() {
174     return ritpp;
175 }
176
177 public void setRitpp(String ritpp) {
178     this.ritpp = ritpp;
179 }
180

```

```

181 public String getFatcva() {
182     return fatcva;
183 }
184
185 public void setFatcva(String fatcva) {
186     this.fatcva = fatcva;
187 }
188
189 public String getAsmas() {
190     return asmas;
191 }
192
193 public void setAsmas(String asmas) {
194     this.asmas = asmas;
195 }
196
197 public String getImp() {
198     return imp;
199 }
200

```

```

201 public void setImp(String imp) {
202     this.imp = imp;
203 }
204
205 public String getImg2() {
206     return img2;
207 }
208
209 public void setImg2(String img2) {
210     this.img2 = img2;
211 }
212
213 public int getTpa() {
214     return tpa;
215 }
216
217 public void setTpa(int tpa) {
218     this.tpa = tpa;
219 }
220

```

```

221 public int getResp() {
222     return rsp;
223 }
224
225 public void setResp(int rsp) {
226     this.rsp = rsp;
227 }
228
229 public String getMmsdc() {
230     return mmsdc;
231 }
232
233 public void setMmsdc(String mmsdc) {
234     this.mmsdc = mmsdc;
235 }
236
237 public int getTldrrr() {
238     return tldrrr;
239 }
240

```

```

241 public void setTldrrr(int tldrrr) {
242     this.tldrrr = tldrrr;
243 }
244
245 public String getDpdt4c() {
246     return dpdt4c;
247 }
248
249 public void setDpdt4c(String dpdt4c) {
250     this.dpdt4c = dpdt4c;
251 }
252
253
254 //Métodos para el historial de resincronizaciones cardíacas
255 public Paciente getResincrosLocal() {
256     return resincrosLocal;
257 }
258
259 public void setResincrosLocal(Paciente resincrosLocal) {
260     this.resincrosLocal = resincrosLocal;
261 }
262

```

Capturas 228: Capturas de pantalla del código de la clase *ResincronizacionCardiaca.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *ResincronizacionCardiaca.java*. Lo indicamos con el siguiente código:

```
502 //se deshabilita la opción borrar desde el formulario de RC
503 @RemoveAction("")
504 //se deshabilita la opción borrar desde la lista de RC
505 @RemoveSelectedAction("")
506 //si se borra paciente, se
507 //borrarán sus pruebas de resincronización cardíaca
508 @OneToMany(mappedBy="resincrosLocal", cascade=CascadeType.ALL)
509 //para poder visualizar la lista de pruebas resincronización cardíaca
510 private Collection<ResincronizacionCardiaca> resincronizacionCardiaca =
511 new ArrayList<ResincronizacionCardiaca>();
512
```

Captura 229: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.8 Sección de sistemas.

La aplicación consta de una sección con tres apartados, dos dedicados a cada tipo de sistema cardiológico que tenga instalado el paciente: generadores y electrodos, y además otro apartado donde se filtran los problemas activos que tiene el paciente en relación a éstos aparatos.

Para cada tipo de sistema se desarrolló un formulario donde se registran sus parámetros para informar del estado de éste, indicando sus unidades de medida para la mejor interpretación de los datos.

Además, el paciente puede tener colecciones de generadores y electrodos.

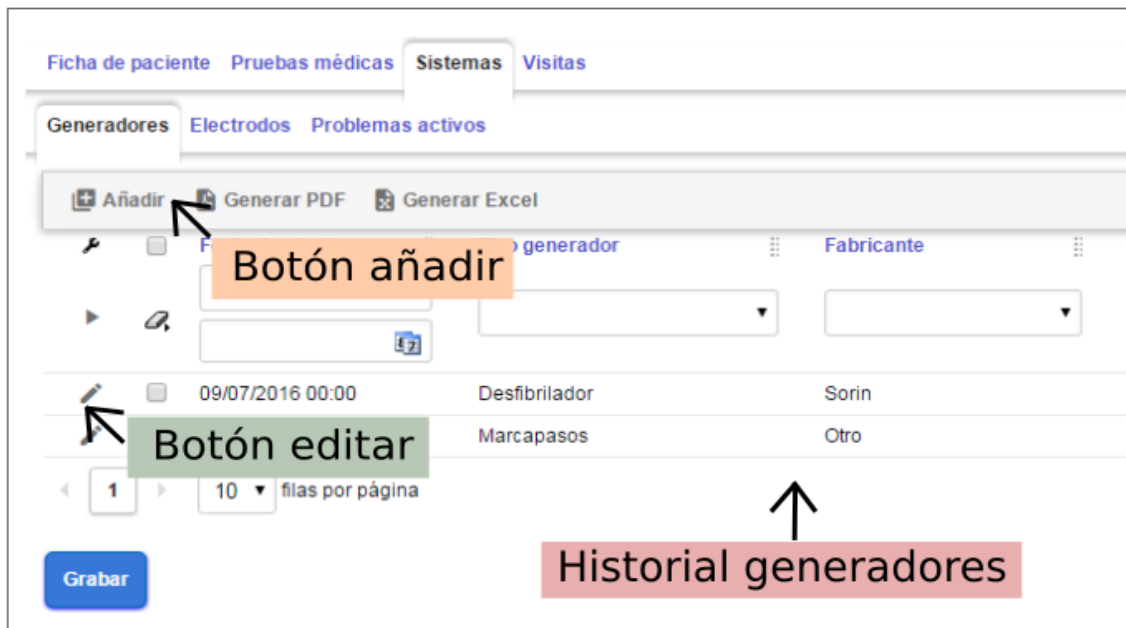
Seguidamente, veremos el código con el que se implementó esta parte de la aplicación web de cardiología.

5.8.1 Generadores.

El usuario puede grabar generadores que tenga implantado el paciente, visualizar el historial de éstos, los datos de cada uno, y también modificarlos.

Tal como se pide, cada generador tiene 0, 1 o varios tipos de complicaciones, los cuales se implementaron como checkboxes, y también tienen 0, 1 o muchos problemas.

Para registrar un generador en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.



Captura 230: Captura de pantalla de la sección Sistemas, apartado Generadores.

▼ Generador

Modelo

Fecha implante

Tipo generador

Médico

Fecha explante

Numero serie

Región implante

Resincro

Médico Ayudante

Causa explante

Fabricante

Acceso

Lugar

Enfermero

▼ Tipo de complicación

Demama Pericárdico Disecación Coronaria Estimulación Frénica Hematomas Hemitorax Infección Insuficiencia Renal Muerte Neurorax Perforación Coronaria Perforación Seno Coronario Sepsis Shock Taponamiento Trombosis Subclavia

▼ Problema

Añadir Generar PDF Generar Excel

Fecha comienzo

Tipo problema

Grado problema

Fecha resolución

Comentario problema

empieza por

No hay registros

filas por página

Hay 0 registros en la lista

▼ Comentario generador

Captura 231: Captura de pantalla del formulario crear generador.

Editar - Generador

▼ Generador

Modelo Numero serie Fabricante Sorin ▼

Fecha implante Región implante Femoral ▼ Acceso Subclavia Derecha ▼

Tipo generador Resincro No ▼ Lugar Quirófano CCV ▼

Médico Médico Asistente Enrique Hernández Ortega ▼ Enfermero Manuel Brito García ▼

Fecha explante Causa explante

▼ Tipo de complicación

Derrame Pericárdico Disección Coronaria Estimulación Frénica Hematoma Hemotorax Infección Insuficiencia Renal Insuficiencia Cardíaca

Muerte Neumotorax Perforación Coronaria Perforación Seno Coronario Sepsis Shock Tapamiento Trombosis Subclavia

▼ Problema

⊞ Añadir

Fecha comienzo Tipo problema Grado problema Fecha resolución Comentario problema

13/07/2016 00:00 Otro 14/07/2016 00:00 funciona?

1 10 ▼ filas por página

Comentario generador

Hay 1 registros en la lista

Captura 232: Captura de pantalla del formulario editar generador.

El código para la clase *Generador.java* que se ha escrito para este apartado, es el

siguiente. En él se reflejan la clase incrustada *TipoComplicación.java*, la colección de problemas con *Problema.java* y las excepciones descritas en los requisitos:

```

1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8 import org.openxava.calculators.*;
9
10 //librería para trabajar con fechas
11 import java.util.*;
12
13 @Entity
14 @View(members= //disposición de los parámetros del dispositivo
15 "Generador [#" +
16 "modelo, numeroSerie, fabricante;" +
17 "fechaImplante, regionImplante, acceso;" +
18 "tipoGenerador, resincro, lugar;" +
19 "medico, medicoAyudante, enfermero;" +
20 "fechaExplante, causaExplante;" +
21 "tipoComplicacion;" +
22 "problema;" +
23 //"complicacionImplante;" +
24 "comentarioGenerador;" +
25 "]"
26 )
27
28 @Table(name = "generador") //se referencia a la tabla de la BB.DD. llamada generador
29 public class Generador extends Identificable { //hereda de la clase Identificable, para
30 //tener un identificador universal único, que se autogenera con funciones
31
32 //PROPIEDADES
33 @Column(name="MODELO", length=20)
34 private String modelo;
35
36 @Column(name="NUMEROSERIE", length=20)
37 private String numeroSerie;
38
39 //A nivel de interfaz gráfico la implementación web actual usa un combo.
40 //La etiqueta para cada valor se obtienen de los archivos i18n (wikispace model_es OX)
41
42 //@Required
43 @Column(name="FABRICANTE")
44 private Fabricante fabricante; //declaro tipo enumerado fabricante que solo puede tener
45 //valores fijos, y además tiene valor predeterminado "otro"
46 public enum Fabricante //se describe un tipo enumerado que se visualiza como un SELECT
47 {
48 BIOTRONIC("Biotronic"), BOSTONSCIENTIFIC ("Boston Scientific"),
49 ELAMEDICAL("Ela Medical"), GUIDANT("Guidant"), INTERMEDICS("Intermedics"),
50 MEDICO("Medico"), MEDTRONIC("Medtronic"), SORIN ("Sorin"), STJUDE("ST Jude"),
51 VITATRON("Vitatron"), OTRO("Otro");
52
53 private String nombreFabricante;
54
55 private Fabricante (String nombreFabricante){
56 this.nombreFabricante = nombreFabricante;
57 }
58
59 public String getNombreFabricante() {
60 return nombreFabricante;
61 }
62
63 }
64
65 @DefaultValueCalculator(CurrentDateCalculator.class) //calculador para darle al campo
66 //por defecto la fecha actual
67 @Column(name="FECHAIMPLANTE")
68 @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
69 @Required // Se mostrará un error de validación si la propiedad se deja en blanco
70 private Date fechaImplante;
71
72 @Column(name="FECHAEXPLANTE")
73 @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario
74 private Date fechaExplante;
75
76 @Column(name="REGIONIMPLANTE")
77 private RegionImplante regionImplante;
78 public enum RegionImplante { //se describe un tipo enumerado que se visualiza como un SELECT
79 ABDOMINAL ("Abdominal"), FEMORAL("Femoral"), PREPECTORAL("Prepectoral"), SUBPECTORAL("Subpectoral");
80
81 private String nombreRegionImplante;

```

```

81
82     private RegionImplante (String nombreRegionImplante){
83         this.nombreRegionImplante = nombreRegionImplante;
84     }
85
86     public String getNombreRegionImplante() {
87         return nombreRegionImplante;
88     }
89 }
90
91
92 @Column(name="LUGAR")
93 private Lugar lugar;
94 public enum Lugar { //se describe un tipo enumerado que se visualiza como un SELECT
95     QUIROFANOCV("Quirófano CCV"), QUIROFANOEEF("Quirófano EEF"),
96     QUIROFANOMCP("Quirófano MCP");
97
98     private String nombreLugar;
99
100    private Lugar (String nombreLugar){
101
102        this.nombreLugar = nombreLugar;
103    }
104
105    public String getNombreTipoLugar() {
106        return nombreLugar;
107    }
108 }
109
110 @Column(name="ACCESO")
111 private Acceso acceso;
112 public enum Acceso { //se describe un tipo enumerado que se visualiza como un SELECT
113     CEFALICAIZQUIERDA("Cefálica Izquierda"), EPICARDICO("Epicárdico"),
114     FEMORALDERECHA("Femoral Derecha"), FEMORALIZQUIERDA("Femoral Izquierda"),
115     SUBCLAVIADERECHA("Subclavia Derecha"), SUBCLAVIAIZQUIERDA("Subclavia Izquierda"),
116     YUGULARDERECHA("Yugular Derecha"), YUGULARIZQUIERDA("Yugular Izquierda");
117
118     private String nombreAcceso;
119
120    private Acceso (String nombreAcceso){
121
122        this.nombreAcceso = nombreAcceso;
123    }
124
125    public String getNombreAcceso() {
126        return nombreAcceso;
127    }
128 }
129
130 @Column(name="TIPOGENERADOR")
131 private TipoGenerador tipoGenerador;
132 public enum TipoGenerador { //se describe un tipo enumerado que se visualiza como un SELECT
133     DESFIBRILADOR("Desfibrilador"), HOLTER("Holter"), MARCAPASOS("Marcapasos"), OTRO("Otro");
134
135     private String nombreTipoGenerador;
136
137    private TipoGenerador (String nombreTipoGenerador){
138        this.nombreTipoGenerador = nombreTipoGenerador;
139    }
140
141    public String getNombreTipoGenerador() {
142
143        return nombreTipoGenerador;
144    }
145 }
146
147 @Column(name="RESINCRO")
148 private Resincro resincro;
149 public enum Resincro { NO("No"), SI("Sí"); //se describe un tipo enumerado que se visualiza como un SELECT
150
151     private String nombreResincro;
152
153    private Resincro (String nombreResincro){
154        this.nombreResincro = nombreResincro;
155    }
156
157    public String getNombreResincro() {
158        return nombreResincro;
159    }
160 }

```

... se omiten médicos, médicos ayudantes y enfermeros...

```

281
282
283     private Enfermero (String nombreEnfermero){
284         this.nombreEnfermero = nombreEnfermero;
285     }
286
287     public String getNombreEnfermero() {
288         return nombreEnfermero;
289     }
290
291     //añadir Lista de Médicos, Médicos ayudantes y Enfermeros activos en la base de datos.
292
293
294     @Column(name="CAUSAEXPLANTE")
295     private CausaExplante causaExplante;
296     public enum CausaExplante { //se describe un tipo enumerado que se visualiza como un SELECT
297         AGOTAMIENTOBAERIA("Agotamiento de Bateria"), DECUBITO("Decúbito"),
298         DISFUNCION("Disfunción"), DISFUNCIONDISPOSITIVO("Disfunción del Dispositivo"),
299         ENDOCARDITIS("Endocarditis"), EOL("EOL"), ERI("ERI"),
300         INFECCIONBOLSA("Infección Bolsa"), RECALL("Recall");
301
302     private String nombreCausaExplante;
303
304     private CausaExplante (String nombreCausaExplante){
305         this.nombreCausaExplante = nombreCausaExplante;
306     }
307
308     public String getNombreCausaExplante() {
309         return nombreCausaExplante;
310     }
311 }
312
313 @Stereotype("MEMO") // Área de texto o equivalente
314 @Column(name="COMENTARIOGENERADOR")
315 private String comentarioGenerador;
316
317
318 //REFERENCIAS
319 @ManyToOne //La referencia tiene que tener valor siempre. No permite Generadores sin Paciente
320 @JoinColumn( //se define la clave foránea de la tabla
321
322     name="numeroNhc", //columna para la clave foránea
323     nullable = true,
324     foreignKey = @ForeignKey(name = "fk_generador_numeroNhc"))
325     //fk_generador_numeroNhc es la la clave foránea en la BB.DD de cardiología
326     private Paciente generadoresLocal; // Una referencia Java convencional
327
328 //CLASE TipoComplicacion IMPLEMENTADA COMO CLASE INCRUSTADA
329 @OneToOne //Anotación para indicar que hay una relación 1 a 1
330 @JoinColumn( //se define la clave foránea de la tabla
331     name = "idtc", //columna para la clave foránea
332     nullable = true,
333     foreignKey = @ForeignKey(name = "fk_generador_idtc"))
334     //fk_generador_idtc es la la clave foránea en la BB.DD de cardiología
335     @AsEmbedded //incrusta la clase TipoComplicacion en la clase Generador
336     private TipoComplicacion tipoComplicacion;
337
338 //@NewAction es una anotación para crear nueva acción que recoge el valor de vista raíz,
339 //para indicarlo en clase Problemas y obtener la colección calculada de problemas Activos
340 @NewAction("calcularpacienteproblema.Añadir")
341 @Collapsed //Anotación que minimiza la pestaña de tipo complicación del generador
342
343
344 @RemoveAction("") //se deshabilita la opción borrar desde el formulario de generador
345 @RemoveSelectedAction("") //se deshabilita la opción borrar desde la lista de generador
346 @OneToMany (mappedBy="problemasLocal", cascade=CascadeType.ALL) //si se borra generador, se
347 //borrarán sus problemas
348 private Collection<Problema> problema = new ArrayList<Problema>();
349 //para poder visualizar la lista de pruebas resincronización cardiaca
350
351 //MÉTODOS
352 public String getModelo() {
353     return modelo;
354 }
355
356 public void setModelo(String modelo) {
357     this.modelo = modelo;
358 }
359
360 public String getNumeroSerie() {
361     return numeroSerie;
362 }

```

```

361
362 public void setNumeroSerie(String numeroSerie) {
363     this.numeroSerie = numeroSerie;
364 }
365
366 public Fabricante getFabricante() {
367     return fabricante;
368 }
369
370 public void setFabricante(Fabricante fabricante) {
371     this.fabricante = fabricante;
372 }
373
374 public Date getFechaImplante() {
375     return fechaImplante;
376 }
377
378 public void setFechaImplante(Date fechaImplante) {
379     this.fechaImplante = fechaImplante;
380 }

```

```

381
382 public Date getFechaExplante() {
383     return fechaExplante;
384 }
385
386 public void setFechaExplante(Date fechaExplante) {
387     this.fechaExplante = fechaExplante;
388 }
389
390 public RegionImplante getRegionImplante() {
391     return regionImplante;
392 }
393
394 public void setRegionImplante(RegionImplante regionImplante) {
395     this.regionImplante = regionImplante;
396 }
397
398 public Lugar getLugar() {
399     return lugar;
400 }

```

```

401
402 public void setLugar(Lugar lugar) {
403     this.lugar = lugar;
404 }
405
406 public Acceso getAcceso() {
407     return acceso;
408 }
409
410 public void setAcceso(Acceso acceso) {
411     this.acceso = acceso;
412 }
413
414 public TipoGenerador getTipoGenerador() {
415     return tipoGenerador;
416 }
417
418 public void setTipoGenerador(TipoGenerador tipoGenerador) {
419     this.tipoGenerador = tipoGenerador;
420 }

```

```

421
422 public Resincro getResincro() {
423     return resincro;
424 }
425
426 public void setResincro(Resincro resincro) {
427     this.resincro = resincro;
428 }
429
430 public Medico getMedico() {
431     return medico;
432 }
433
434 public void setMedico(Medico medico) {
435     this.medico = medico;
436 }
437
438 public MedicoAyudante getMedicoAyudante() {
439     return medicoAyudante;
440 }

```

```

441
442 //método para lanzar excepción cuando medicoAyudante=medico
443 //y/o fechaExplant<fechaImplante al crear objeto
444 @PrePersist //Ejecutado justo antes de grabar el objeto por primera vez
445 private void onPersist() throws Exception{
446     if (medico != null && medicoAyudante != null
447         && medicoAyudante.toString().equals(medico.toString())) { //lógica de validación
448         throw new javax.validation.ValidationException("los_medicos_deben_ser_distintos");
449     }
450     if (fechaExplant != null && fechaExplant.before(fechaImplante) ){
451         throw new javax.validation.ValidationException("fecha_explant_deben_ser_mayor_que_fecha_implante");
452     }
453 }
454
455 //método para lanzar excepción cuando medicoAyudante=medico
456 //y/o fechaExplant<fechaImplante al actualizar objeto
457 @PreUpdate //Ejecutado al momento de actualizar posteriormente la entidad
458 private void onUpdate() throws Exception{
459     if (medico != null && medicoAyudante != null
460         && medicoAyudante.toString().equals(medico.toString())) { //lógica de validación

```

```

461         throw new javax.validation.ValidationException("los_medicos_deben_ser_distintos");
462     }
463     if (fechaExplant != null && fechaExplant.before(fechaImplante) ){
464         throw new javax.validation.ValidationException("fecha_explant_deben_ser_mayor_que_fecha_implante");
465     }
466 }
467
468 public void setMedicoAyudante(MedicoAyudante medicoAyudante) {
469     this.medicoAyudante = medicoAyudante;
470 }
471
472 public Enfermero getEnfermero() {
473     return enfermero;
474 }
475
476 public void setEnfermero(Enfermero enfermero) {
477     this.enfermero = enfermero;
478 }
479
480 public CausaExplant getCausaExplant() {

```

```

481     return causaExplant;
482 }
483
484 public void setCausaExplant(CausaExplant causaExplant) {
485     this.causaExplant = causaExplant;
486 }
487
488 public String getComentarioGenerador() {
489     return comentarioGenerador;
490 }
491
492 public void setComentarioGenerador(String comentarioGenerador) {
493     this.comentarioGenerador = comentarioGenerador;
494 }
495
496 public Paciente getGeneradoresLocal() {
497     return generadoresLocal;
498 }
499
500 public void setGeneradoresLocal(Paciente generadoresLocal) {

```

```

501         this.generadoresLocal = generadoresLocal;
502     }
503
504     public TipoComplicacion getTipoComplicacion() {
505         return tipoComplicacion;
506     }
507
508     public void setTipoComplicacion(TipoComplicacion tipoComplicacion) {
509         this.tipoComplicacion = tipoComplicacion;
510     }
511
512     //Métodos para el historial de generadores
513     public Collection<Problema> getProblema() {
514         return problema;
515     }
516
517     public void setProblema(Collection<Problema> problema) {
518         this.problema = problema;
519     }
520 }
521

```

Capturas 233: Capturas de pantalla del código de la clase *Generador.java*.

El código de la clase incrustada *TipoComplicacion.java* es el siguiente, con sus respectivos comentarios:

```

1  package org.openxava.cardiologia.model;
2
3  //librerías para persistencia de los datos
4  import javax.persistence.*;
5
6  //librerías para usar las funciones Openxava
7  import org.openxava.annotations.*;
8
9  @Entity
10 @View(members= //disposición de los parámetros del dispositivo
11 //si le quitamos # no se alinean
12 "#derramePericardico, diseccionCoronaria, estimulacionFrenica, hematoma, hemotorax, infeccion, insufi
13 "muerte, neumotorax, perforacionCoronaria, perforacionSenoCoronario, sepsis, shock, taponamiento, trom
14 )
15 @Table(name="tipocomplicacion") //se referencia a la tabla de la BB.DD. llamada tipocomplicacion
16

```

```

20
21     @Column(name="diseccioncoronaria")
22     private Boolean diseccionCoronaria;
23
24     @Column(name="estimulacionfrenica")
25     private Boolean estimulacionFrenica;
26
27     @Column(name="hematoma")
28     private Boolean hematoma;
29
30     @Column(name="hemotorax")
31     private Boolean hemotorax;
32
33     @Column(name="infeccion")
34     private Boolean infeccion;
35
36     @Column(name="insuficienciarenal")
37     private Boolean insuficienciaRenal;
38
39     @Column(name="insuficienciacardiaca")
40     private Boolean insuficienciaCardiaca;
41
42     @Column(name="muerte")
43     private Boolean muerte;
44
45     @Column(name="neumotorax")
46     private Boolean neumotorax;
47
48     @Column(name="perforacioncoronaria")
49     private Boolean perforacionCoronaria;
50

```



```

51     @Column(name="perforacionesenocoronario")
52     private Boolean perforacionSenoCoronario;
53
54     @Column(name="sepsis")
55     private Boolean sepsis;
56
57     @Column(name="shock")
58     private Boolean shock;
59
60     @Column(name="taponamiento")
61     private Boolean taponamiento;
62
63     @Column(name="trombosissubclavia")
64     private Boolean trombosisSubclavia;
65
66
67     //MÉTODOS
68     public Boolean getDerramePericardico() {
69         return derramePericardico;
70     }
71
72     public void setDerramePericardico(Boolean derramePericardico) {
73         this.derramePericardico = derramePericardico;
74     }
75
76     public Boolean getDisecccionCoronaria() {
77         return disecccionCoronaria;
78     }
79
80     public void setDisecccionCoronaria(Boolean disecccionCoronaria) {
81         this.disecccionCoronaria = disecccionCoronaria;
82     }
83
84     public Boolean getEstimulacionFrenica() {
85         return estimulacionFrenica;
86     }
87
88     public void setEstimulacionFrenica(Boolean estimulacionFrenica) {
89         this.estimulacionFrenica = estimulacionFrenica;
90     }
91
92     public Boolean getHematoma() {
93         return hematoma;
94     }
95
96     public void setHematoma(Boolean hematoma) {
97         this.hematoma = hematoma;
98     }
99
100    public Boolean getHemotorax() {
101        return hemotorax;
102    }
103
104    public void setHemotorax(Boolean hemotorax) {
105        this.hemotorax = hemotorax;
106    }
107
108    public Boolean getInfeccion() {
109        return infeccion;
110    }

```

```

111
112  public void setInfeccion(Boolean infeccion) {
113      this.infeccion = infeccion;
114  }
115
116  public Boolean getInsuficienciaRenal() {
117      return insuficienciaRenal;
118  }
119
120  public void setInsuficienciaRenal(Boolean insuficienciaRenal) {
121      this.insuficienciaRenal = insuficienciaRenal;
122  }
123
124  public Boolean getInsuficienciaCardiaca() {
125      return insuficienciaCardiaca;
126  }
127
128  public void setInsuficienciaCardiaca(Boolean insuficienciaCardiaca) {
129      this.insuficienciaCardiaca = insuficienciaCardiaca;
130  }
131
132  public Boolean getMuerte() {
133      return muerte;
134  }
135
136  public void setMuerte(Boolean muerte) {
137      this.muerte = muerte;
138  }
139
140  public Boolean getNeumotorax() {
141      return neumotorax;
142  }
143
144  public void setNeumotorax(Boolean neumotorax) {
145      this.neumotorax = neumotorax;
146  }
147
148  public Boolean getPerforacionCoronaria() {
149      return perforacionCoronaria;
150  }
151
152  public void setPerforacionCoronaria(Boolean perforacionCoronaria) {
153      this.perforacionCoronaria = perforacionCoronaria;
154  }
155
156  public Boolean getPerforacionSenoCoronario() {
157      return perforacionSenoCoronario;
158  }
159
160  public void setPerforacionSenoCoronario(Boolean perforacionSenoCoronario) {
161      this.perforacionSenoCoronario = perforacionSenoCoronario;
162  }
163
164  public Boolean getSepsis() {
165      return sepsis;
166  }
167
168  public void setSepsis(Boolean sepsis) {
169      this.sepsis = sepsis;
170  }

```

```

170     }
171
172     public Boolean getShock() {
173         return shock;
174     }
175
176     public void setShock(Boolean shock) {
177         this.shock = shock;
178     }
179
180     public Boolean getTaponamiento() {
181         return taponamiento;
182     }
183
184     public void setTaponamiento(Boolean taponamiento) {
185         this.taponamiento = taponamiento;
186     }
187
188     public Boolean getTrombosisSubclavia() {
189         return trombosisSubclavia;
190     }
191
192     public void setTrombosisSubclavia(Boolean trombosisSubclavia) {
193         this.trombosisSubclavia = trombosisSubclavia;
194     }
195
196 }
197

```

Capturas 234: Capturas de pantalla de la clase *TipoComplicacion.java*.

Añadir - Problema

▼ Problema

Fecha comienzo 09/05/2017 22:44 Fecha resolución

Tipo problema Grado problema Nhc

▼ Comentario problema

Captura 235: Captura de pantalla del formulario añadir problema.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.

Captura 236: Captura de pantalla del formulario editar problema.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

A su vez, la clase *Problema.java* que describe la colección de problemas que puede sufrir el generador se describe a continuación:

```

1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.calculators.*;
8 import org.openxava.annotations.*;
9
10 //librería para trabajar con fechas
11 import java.util.*;
12
13
14 @Entity
15 @View(members= //disposición de los parámetros del dispositivo
16 "Problema [#" +
17 "fechaComienzo, fechaResolucion;" +
18 "tipoProblema, gradoProblema, nhc;" +
19 "comentarioProblema;" +
20 "]"

```

```

21 )
22 //se referencia a la tabla de la BB.DD. llamada problema
23 @Table(name="problema")
24 //hereda de la clase Identificable, para
25 //tener un identificador universal único, que se autogenera con funciones
26 public class Problema extends Identificable {
27
28     //PROPIEDADES
29
30     //calculador para darle al campo
31     //por defecto la fecha actual
32     @DefaultValueCalculator(CurrentDateCalculator.class)
33     @Column( name= "FECHACOMIENZO")
34     //Anotación Openxava para añadir un calendario
35     @Stereotype("FECHAHORA")
36     // Se mostrará un error de validación si la propiedad se deja en blanco
37     @Required
38     private Date fechaComienzo;
39
40     @Column( name= "FECHARESOLUCION")
41     @Stereotype("FECHAHORA") //Anotación Openxava para añadir un calendario

```

```

42     private Date fechaResolucion;
43
44     //definir tipo problema
45     @Column( name= "TIPOPROBLEMA")
46     private TipoProblema tipoProblema;
47     //se describe un tipo enumerado que se visualiza como un SELECT
48     public enum TipoProblema {
49         ALTERACIONESELECTRODOS("Alteraciones Electrodo"),
50         ARRITMIASFRECUENTES("Arritmias Frecuentes"),
51         DESCARGASINAPROPIADAS ("Descargas Inapropiadas"),
52         DETERIOROCLINICO("Deterioro Clínico"),
53         ERI("ERI"),
54         INFECCIONBOLSAELECTRODO("Infección Bolsa Electrodo"),
55         OTRO("Otro"), RECALL("Recall");
56
57         private String nombreTipoProblema;
58
59         private TipoProblema (String nombreTipoProblema){
60             this.nombreTipoProblema = nombreTipoProblema;
61         }
62

```

```

63         public String getNombreTipoProblema() {
64             return nombreTipoProblema;
65         }
66     }
67 }
68
69 //definir grado problema
70 @Column( name= "GRADOPROBLEMA")
71 private GradoProblema gradoProblema;
72 //se describe un tipo enumerado que se visualiza como un SELECT
73 public enum GradoProblema {
74     LEVE("Leve"), MODERADO("Moderado"), GRAVE("Grave");
75

```

```

76         private String nombreGradoProblema;
77
78         private GradoProblema (String nombreGradoProblema){
79             this.nombreGradoProblema = nombreGradoProblema;
80         }
81
82         public String getNombreGradoProblema() {
83             return nombreGradoProblema;
84         }
85     }
86
87
88     @Column( name= "COMENTARIOPROBLEMA")
89     @Stereotype("MEMO") // Área de texto o equivalente
90     private String comentarioProblema;
91
92     //añadido 15/06
93     @ReadOnly
94     @Column( name= "nhc", nullable= true)
95     private int nhc;
96
97
98     //REFERENCIAS
99
100
101     // La referencia tiene que tener valor siempre. No permite Problema sin Generador
102     @ManyToOne
103     @JoinColumn( //se define la clave foránea de la tabla
104                 name="idGenerador", //columna para la clave foránea
105                 nullable = true,
106                 foreignKey = @ForeignKey(name = "fk_problema_numeronhc2"))
107     private Generator problemasLocal; // Una referencia Java convencional
108
109
110
111     //MÉTODOS
112     public Date getFechaComienzo() {
113         return fechaComienzo;
114     }
115
116     public void setFechaComienzo(Date fechaComienzo) {
117         this.fechaComienzo = fechaComienzo;
118     }
119
120     public Date getFechaResolucion() {
121         return fechaResolucion;
122     }
123
124     public void setFechaResolucion(Date fechaResolucion) {
125         this.fechaResolucion = fechaResolucion;
126     }
127
128     @PrePersist //Ejecutado justo antes de grabar el objeto por primera vez
129     private void onPersist() throws Exception{
130         if (fechaResolucion != null && fechaResolucion.before(fechaComienzo) ){
131             //con esta línea se consigue lanzar una alerta de la excepción ocurrida
132             throw new javax.validation.ValidationException
133                 ("fecha_resolucion_deben_ser_mayor_que_fecha_comienzo");
134         }
135     }
136
137     @PreUpdate //Ejecutado al momento de actualizar posteriormente la entidad
138     private void onUpdate() throws Exception{
139         if (fechaResolucion != null && fechaResolucion.before(fechaComienzo) ){
140             //con esta línea se consigue lanzar una alerta de la excepción ocurrida
141             throw new javax.validation.ValidationException
142                 ("fecha_resolucion_deben_ser_mayor_que_fecha_comienzo");
143         }
144     }

```

```

145
146 public TipoProblema getTipoProblema() {
147     return tipoProblema;
148 }
149
150 public void setTipoProblema(TipoProblema tipoProblema) {
151     this.tipoProblema = tipoProblema;
152 }
153
154 public GradoProblema getGradoProblema() {
155     return gradoProblema;
156 }
157
158 public void setGradoProblema(GradoProblema gradoProblema) {
159     this.gradoProblema = gradoProblema;
160 }
161
162 public String getComentarioProblema() {
163     return comentarioProblema;
164 }
165
166 public void setComentarioProblema(String comentarioProblema) {
167     this.comentarioProblema = comentarioProblema;
168 }
169

```

```

170 //añadido 15/06
171 public int getNhc() {
172     return nhc;
173 }
174
175 public void setNhc(int nhc) {
176     this.nhc = nhc;
177 }
178
179 //Métodos para el historial de problemas de generador
180 public Generador getProblemasLocal() {
181     return problemasLocal;
182 }
183
184 public void setProblemasLocal(Generador problemasLocal) {
185     this.problemasLocal = problemasLocal;
186 }
187
188
189 }
190

```

Capturas 237: Capturas de pantalla del código de la clase *Problema.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *Generador.java*. Lo indicamos con el siguiente código:

```

513
514 //se deshabilita la opción borrar desde el formulario de Generador
515 @RemoveAction("")
516 //se deshabilita la opción borrar desde la lista de Generador
517 @RemoveSelectedAction("")
518 //si se borra paciente, se
519 //borrarán sus generadores
520 @OneToMany(mappedBy="generadoresLocal", cascade=CascadeType.ALL)
521 //para poder visualizar la lista de generadores
522 private Collection<Generador> generador = new ArrayList<Generador>();
523

```

Captura 238: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

Por último, vemos las etiquetas que necesitamos para la internacionalización de este apartado:

```

174 #Componentes de Generador.
175 fechaResolucion=Fecha resolución
176 BIOTRONIC=Biotronic
177 BOSTONSCIENTIFIC=Boston Scientific
178 ELAMEDICAL=Elá Medical
179 GUIDANT=Guidant
180 INTERMEDICS=Intermedics
181 MEDICO=Medico
182 MEDTRONIC=Medtronic
183 OTRO=Otro
184 SORIN=Sorin
185 STJUDE=ST Jude
186 VITATRON=Vitatron
187 ABDOMINAL=Abdominal
188 FEMORAL=Femoral
189 PREPECTORAL=Prepectoral
190 SUBPECTORAL=Subpectoral
191 QUIROFANOCCV=Quirófano CCV
192 QUIROFANOEEF=Quirófano EEF
193 QUIROFANOMCP=Quirófano MCP
194 CEFALICAIZQUIERDA=Cefálica Izquierda

195 EPICARDICO=Epicárdico
196 FEMORALDERECHA=Femoral Derecha
197 FEMORALIZQUIERDA=Femoral Izquierda
198 SUBCLAVIADERECHA=Subclavia Derecha
199 SUBCLAVIAIZQUIERDA=Subclavia Izquierda
200 YUGULARDERECHA=Yugular Derecha
201 YUGULARIZQUIERDA=Yugular Izquierda
202 DESFIBRILADOR=Desfibrilador
203 HOLTER=Holter
204 MARCAPASOS=Marcapasos
205 NO=No
206 SI=Sí
207 AGOTAMIENTOBATERIA=Agotamiento de Batería
208 DECUBITO=Decúbito
209 DISFUNCION=Disfunción
210 DISFUNCIONDISPOSITIVO=Disfunción del Dispositivo
211 ENDOCARDITIS=Endocarditis
212 EOL=EOL
213 ERI=ERI
214 INFECCIONBOLSA=Infección Bolsa
215 RECALL=Recall
216 tipoComplicacion=Tipo de complicación

```



```

238 #Componentes de Problema
239 ALTERACIONESELECTRODOS=Alteraciones Electroodos
240 ARRITMIASFRECUENTES=Arritmias Frecuentes
241 DESCARGASINAPROPIADAS=Descargas Inapropiadas
242 DETERIOROCLINICO=Deterioro Clínico
243 ERI=ERI
244 INFECCIONBOLSAELECTRODO=Infección Bolsa Electrodo
245 LEVE=Leve
246 MODERADO=Moderado
247 GRAVE=Grave
248

```

Capturas 239: Captura de pantalla de fragmento del script *EtiquetasCardiologias_es.properties*.

5.8.2 Electroodos.

El usuario puede grabar cada electrodo que tenga implantado el paciente, visualizar el historial de éstos, los datos de cada uno, y también modificarlos.

Para registrar un electrodo en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.

The screenshot shows the 'Electrodos' section of a medical system. At the top, there are tabs for 'Generadores', 'Electrodos', and 'Problemas activos'. Below the tabs, there are buttons for 'Añadir', 'Generar PDF', and 'Generar Excel'. A table lists electrode records with columns: 'Fecha implante', 'Cámara', 'Amplitud', 'Impedancia', and 'Fecha explante'. Annotations highlight the 'Añadir' button, an edit icon, and the table itself.

Fecha implante	Cámara	Amplitud	Impedancia	Fecha explante
18/03/2013 00:00	Ventriculo Derecho		2,00	09/03/2015 00:00
22/03/2016 00:00	Aurícula Derecha			26/03/2016 00:00
30/05/2016 00:00	Aurícula Derecha	324,00	234,00	31/05/2016 00:00
		Σ	Σ	

Captura 240: Captura de pantalla de la sección Sistemas, apartado Electroodos.

The screenshot shows a form titled 'Crea una nueva entidad - Electrodo'. It contains several input fields and dropdown menus for recording electrode data. The fields are: 'Modelo', 'Numero serie', 'Fabricante', 'Fecha implante' (with a date picker), 'Cámara', 'Deflexión intrínsecoide' (with units 'mV/ms'), 'Amplitud' (with units 'mV'), 'Impedancia' (with units 'Ohmios'), 'Fecha explante' (with a date picker), and 'Causa explante'. At the bottom, there are three buttons: 'Grabar', 'Grabar y continuar', and 'Cerrar'.

Captura 241: Captura de pantalla del formulario crear electrodo.

El botón Grabar cierra el formulario al grabar el objeto en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.

Captura 242: Captura de pantalla del formulario editar electrodo.

El botón Grabar cierra el formulario al modificar el objeto en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

El código para la clase *Electrodo.java* que se ha escrito para este apartado, es el siguiente:

```

1  package org.openxava.cardiologia.model;
2
3  //librerías para persistencia de los datos
4  import javax.persistence.*;
5
6  //librerías para usar las funciones Openxava
7  import org.openxava.annotations.*;
8  import org.openxava.calculators.*;
9
10 //librería para trabajar con fechas
11 import java.util.Date;
12
13 @Entity
14 @View(members= //disposición de los parámetros del dispositivo
15 "Electrodo [#" +
16 "modelo, numeroSerie, fabricante;" +
17 "fechaImplante, camara;" +
18 "deflexionIntrinsecoide, amplitud, impedancia ;" +
19 "fechaExplante, causaExplante;" +
20 "]"
21 )

```

```

22 //se referencia a la tabla de la BB.DD. llamada electrodo
23 @Table (name="electrodo")
24 //hereda de la clase Identificable, para
25 //tener un identificador universal único, que se autogenera con funciones
26 public class Electrodo extends Identificable{
27
28     //PROPIEDADES
29     @Column(name= "MODELO", length=20)
30     private String modelo;
31
32     @Column(name= "NUMEROSERIE", length=20)
33     private String numeroSerie;

```

```

35
36     @Column(name= "FABRICANTE")
37     //declaro tipo enumerado fabricante que solo puede tener valores fijos,
38     //y además tiene valor predeterminado "otro"
39     private Fabricante fabricante;
40     public enum Fabricante
41     {
42         BIOTRONIC("Biotronic"), BOSTONSCIENTIFIC ("Boston Scientific"),
43         ELAMEDICAL("Ela Medical"), GUIDANT("Guidant"),
44         INTERMEDICS("Intermedics"), MEDICO("Medico"),
45         MEDTRONIC("Medtronic"), SORIN ("Sorin"),
46         STJUDE("ST Jude"), VITATRON("Vitatron"), OTRO("Otro");
47
48         private String nombreFabricante;
49
50         private Fabricante (String nombreFabricante){
51             this.nombreFabricante = nombreFabricante;
52         }
53
54         public String getNombreFabricante() {
55             return nombreFabricante;
56         }
57     }
58 }

```

```

59
60     @DefaultValueCalculator(CurrentDateCalculator.class)
61     @Column(name= "FECHAIMPLANTE")
62     //Anotación Openxava para añadir un calendario
63     @Stereotype("FECHAHORA")
64     // Se mostrará un error de validación si la propiedad se deja en blanco
65     @Required
66     private Date fechaImplante;
67
68     @Column(name= "FECHAEXPLANTE")
69     //Anotación Openxava para añadir un calendario
70     @Stereotype("FECHAHORA")
71     private Date fechaExplante;
72
73     @Required
74     @Column(name= "CAMARA")
75     private Camara camara;
76     //declaro tipo enumerado cámara que solo puede tener valores fijos
77     public enum Camara
78     {

```

```

79     AURICULADERECHA("Aurícula Derecha"),
80     VENTRICULODERECHO("Ventrículo Derecho"),
81     VENTRICULOIZQUIERDO("Ventrículo Izquierdo");
82
83     private String nombreCamara;
84
85     private Camara (String nombreCamara){
86         this.nombreCamara = nombreCamara;
87     }
88
89     public String getNombreCamara() {
90         return nombreCamara;
91     }
92
93 }
94
95
96 @Column(name= "CAUSAEXPLANTE")
97 //se describe un tipo enumerado que se visualiza como un SELECT
98 private CausaExplante causaExplante;
99 public enum CausaExplante { DECUBITO("Decúbito"),
100     DISFUNCION("Disfunción"), ENDOCARDITIS("Endocarditis"),
101     FRACTURA("Fractura"), INFECCIONBOLSA("Infección Bolsa"),
102     MIGRACION("Migración"), PERDIDAAISLAMIENTO("Pérdida Aislamiento"),
103     PERDIDACAPTURA("Pérdida Captura"), RECALL("Recall");
104

```

```

105     private String nombreCausaExplante;
106
107     private CausaExplante (String nombreCausaExplante){
108         this.nombreCausaExplante = nombreCausaExplante;
109     }
110
111     public String getCausaExplante() {
112         return nombreCausaExplante;
113     }
114
115 }
116
117 //Anotación Openxava para añadir unidad de medida mvms.
118 @Stereotype("MVMS")
119 @Column(name= "DEFLEXIONINTRINSECOIDE", length=5)
120 private int deflexionIntrinsecoide;
121
122 //Anotación Openxava para añadir unidad de medida mv.
123 @Stereotype("MV")
124 @Column(name= "AMPLITUD", length=5)
125 private int amplitud;
126

```

```

127 //Anotación Openxava para añadir unidad de medida ohmios.
128 @Stereotype("OHMIOS")
129 @Column(name= "IMPEDANCIA", length=5)
130 private int impedancia;
131
132

```

```

133 //REFERENCIAS
134 //La referencia tiene que tener valor siempre.
135 //No permite Electrodo sin Paciente
136 @ManyToOne
137 @JoinColumn( //se define la clave foránea de la tabla
138             name="numeroNhc", //columna para la clave foránea
139             nullable = true,
140             foreignKey = @ForeignKey(name = "fk_electrodo_numeronhc"))
141             //fk_electrodo_numeronhc es la la clave foránea
142             //en la BB.DD de cardiología
143 private Paciente electrodosLocal; // Una referencia Java convencional
144
145
146 //MÉTODOS
147 public String getModelo() {
148     return modelo;
149 }
150
151 public void setModelo(String modelo) {
152     this.modelo = modelo;
153 }
154
155 public String getNumeroSerie() {
156     return numeroSerie;
157 }

```

```

158
159 public void setNumeroSerie(String numeroSerie) {
160     this.numeroSerie = numeroSerie;
161 }
162
163 public Fabricante getFabricante() {
164     return fabricante;
165 }
166
167 public void setFabricante(Fabricante fabricante) {
168     this.fabricante = fabricante;
169 }
170
171 public Date getFechaImplante() {
172     return fechaImplante;
173 }
174

```

```

175 public void setFechaImplante(Date fechaImplante) {
176     this.fechaImplante = fechaImplante;
177 }
178
179 public Date getFechaExplante() {
180     return fechaExplante;
181 }
182
183 public void setFechaExplante(Date fechaExplante) {
184     this.fechaExplante = fechaExplante;
185 }

```

```

186
187 //se cumple la excepción para que fechaImplante<fechaImplante
188 @PrePersist //Ejecutado justo antes de grabar el objeto por primera vez
189 private void onPersist() throws Exception{
190     if (fechaExplante != null && fechaExplante.before(fechaImplante) ){
191         throw new javax.validation.ValidationException
192             ("fecha_resolucion_deben_ser_mayor_que_fecha_comienzo");
193     }
194 }
195
196 @PreUpdate //Ejecutado al momento de actualizar posteriormente la entidad
197 private void onUpdate() throws Exception{
198     if (fechaExplante != null && fechaExplante.before(fechaImplante) ){
199         throw new javax.validation.ValidationException
200             ("fecha_resolucion_deben_ser_mayor_que_fecha_comienzo");
201     }
202 }
203
204
205 public Camara getCamara() {
206     return camara;
207 }
208

```

```

209 public void setCamara(Camara camara) {
210     this.camara = camara;
211 }
212
213 public CausaExplante getCausaExplante() {
214     return causaExplante;
215 }
216
217 public void setCausaExplante(CausaExplante causaExplante) {
218     this.causaExplante = causaExplante;
219 }
220
221 public int getDeflexionIntrinsecoide() {
222     return deflexionIntrinsecoide;
223 }
224
225 public void setDeflexionIntrinsecoide(int deflexionIntrinsecoide) {
226     this.deflexionIntrinsecoide = deflexionIntrinsecoide;
227 }
228
229 public int getAmplitud() {
230     return amplitud;
231 }

```

```

232
233 public void setAmplitud(int amplitud) {
234     this.amplitud = amplitud;
235 }
236
237 public int getImpedancia() {
238     return impedancia;
239 }
240
241 public void setImpedancia(int impedancia) {
242     this.impedancia = impedancia;
243 }

```

```

244
245 public Paciente getElectrodosLocal() {
246     return electrodosLocal;
247 }
248
249 public void setElectrodosLocal(Paciente electrodosLocal) {
250     this.electrodosLocal = electrodosLocal;
251 }
252 }

```

Capturas 243: Capturas de pantalla del código de la clase *Electrodo.java*.

En la clase *Paciente.java* también debemos indicar la relación 1 a * con la clase *Electrodo.java*. Lo indicamos con el siguiente código:

```

525 //se deshabilita la opción borrar desde el formulario de Electrodo
526 @RemoveAction("")
527 //se deshabilita la opción borrar desde la lista de Electrodo
528 @RemoveSelectedAction("")
529 //si se borra paciente, se
530 //borrarán sus electrodos
531 @OneToMany(mappedBy="electrodosLocal", cascade=CascadeType.ALL)
532 //para poder visualizar la lista de electrodos
533 private Collection<Electrodo> electrodo = new ArrayList<Electrodo>();

```

Captura 244: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

5.8.2.1 Estereotipos para la clase.

Además, definimos los estereotipos mv/ms, ohmios y mv:

- Fichero *editores.xml*:

```

57⊖ <editor nombre="mv" url="mvEditor.jsp">
58     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
59     <para-estereotipo estereotipo="MV"/>
60 </editor>
61
62⊖ <editor nombre="ohmios" url="ohmiosEditor.jsp">
63     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
64     <para-estereotipo estereotipo="OHMIOS"/>
65 </editor>
66
67⊖ <editor nombre="mvms" url="mvmsEditor.jsp">
68     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
69     <para-estereotipo estereotipo="MVMS"/>
70 </editor>
71

```

Captura 245: Captura de pantalla de fragmento de código del script *editores.xml*.

- Clase formateadora (sólo se ha de crear una sola vez):

```

1 package org.openxava.cardiologia.formateadores;
2
3⊕ import java.math.*;
4
5
6 public class PesoFormateador implements IFormatter {
7
8     public String format(HttpServletRequest request, Object object) throws Exception {
9         if (Is.empty(object)) return "";
10        return getFormat().format(object);
11    }
12
13    public Object parse(HttpServletRequest request, String string) throws Exception {
14        if (Is.emptyString(string)) return null;
15        string = Strings.change(string, " ", ""); // In order to work with Polish
16        return new BigDecimal(getFormat().parse(string).toString()).setScale(2);
17    }
18
19    private NumberFormat getFormat() {
20        NumberFormat f = DecimalFormat.getNumberInstance(Locales.getCurrent());
21        f.setMinimumFractionDigits(2);
22        f.setMaximumFractionDigits(2);
23        return f;
24    }
25 }
26
27
28
29
30

```

Captura 246: Captura de pantalla de la clase formateadora *PesoFormateador*.

- Scripts de los editores:

```
1 |<%@page import="java.util.Currency"%>
2 |<%@page import="java.util.Locale"%>
3 |<%@ include file="textEditor.jsp"%>
4 |
5 |<% String symbol = null;
6 | try {
7 | symbol = "mV/ms"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 | }
9 | catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 | symbol = "?";
11 | }
12 | %>
13 |
14 | <%=symbol%>
```

Captura 247: Captura de pantalla del script *mvms.jsp*.

```
1 |<%@page import="java.util.Currency"%>
2 |<%@page import="java.util.Locale"%>
3 |<%@ include file="textEditor.jsp"%>
4 |
5 |<% String symbol = null;
6 | try {
7 | symbol = "Ohmios"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 | }
9 | catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 | symbol = "?";
11 | }
12 | %>
13 |
14 | <%=symbol%>
```

Captura 248: Captura de pantalla del script *ohmios.jsp*.

```
1 |<%@page import="java.util.Currency"%>
2 |<%@page import="java.util.Locale"%>
3 |<%@ include file="textEditor.jsp"%>
4 |
5 |<% String symbol = null;
6 | try {
7 | symbol = "mV"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 | }
9 | catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 | symbol = "?";
11 | }
12 | %>
13 |
14 | <%=symbol%>
```

Captura 249: Captura de pantalla del script *mv.jsp*

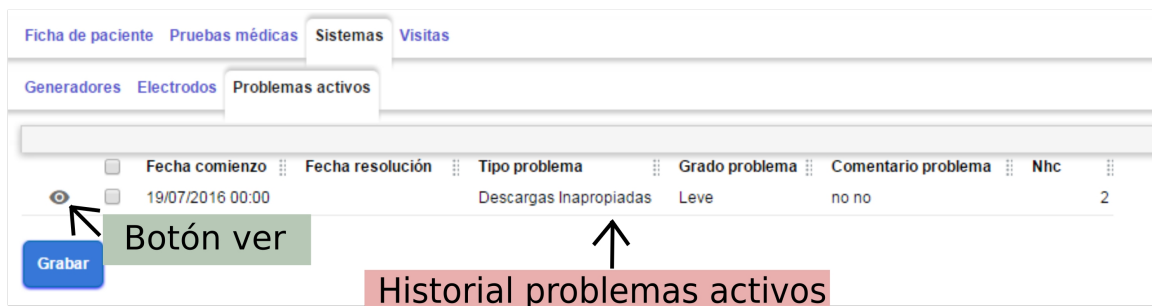
Por último, vemos las etiquetas que necesitamos para la internacionalización de este apartado:

```
249 #Componentes de Electrodo
250 amplitud=Amplitud
251 impedancia=Impedancia
252 camara=Cámara
253 deflexionIntrinsecoide=Deflexión intrinsecoide
254 AURICULADERECHA=Aurícula Derecha
255 VENTRICULODERECHO=Ventrículo Derecho
256 VENTRICULOIZQUIERDO=Ventrículo Izquierdo
257 FRACTURA=Fractura
258 MIGRACION=Migración
259 PERDIDAAISLAMIENTO=Pérdida Aislamiento
260 PERDIDACAPTURA=Pérdida Captura
```

Captura 250: Captura de pantalla de fragmento del script *EtiquetasCardiologias_es.properties*.

5.8.3 Problemas activos.

En esta sección se visualizan los problemas activos del paciente en modo lectura, es decir aquellos en los que la fecha de resolución de problemas es nula. También se puede visualizar el historial de problemas activos en una lista, y se puede leer los datos de cada problema (clase ya explicada con anterioridad).



Captura 251: Captura de pantalla de la Sección Sistemas, apartado Problemas activos.

Al picar en el botón ver, podemos ver toda la información del problema activo:

Captura 252: Captura de pantalla del formulario ver Problemas activos.

Para poder obtener esta colección calculada de problemas activos del pacientes, fue necesario recoger el valor del número de historia clínica del paciente de vista raíz y ponerlo en clase Problemas para hacer luego colección calculada.

No pudimos realizarlo con la anotación en el parámetro calculado **@DefaultValueCalculator** (calculador por defecto) porque no nos permite las acciones tipo View para recoger valores de las vistas, en nuestro caso, numeroNhc.

Tampoco con la anotación **@OnChange**, porque al abrir el formulario de Problema, nunca podríamos acceder a valores de la última vista, ya que el getView realmente coge la subvista (en nuestro caso, generadores y no problema)

Tras varios intentos, finalmente se optó por la anotación **@NewAction** ya que nos da la solución a nuestro problema. Es una acción que sustituye la acción "new" sobre la colección. Los pasos que se siguieron y el código modificado está a continuación:

Primer paso. Acción que hereda de nuevo elemento de colección, y además rellena el campo de la entidad de la vista:

```

1 package org.openxava.cardiologia.actions;
2
3 import org.openxava.actions.*;
4
5
6 public class CalcularPacienteEnProblema extends CreateNewElementInCollectionAction{
7
8     @Override
9     public void execute() throws Exception
10    {
11        // Here you are still on parent's view using getView()
12        Integer numeroPaciente = getView().getRoot().getValueInt("numeroNhc");
13        super.execute();
14        // Here you have the view for the collection in getCollectionElementView().
15        getCollectionElementView().setValue("nhc", numeroPaciente);
16    }
17

```

Captura 253: Captura de pantalla de la acción *CalcularPacienteEnProblema*.

Anotar que el código de la línea 12, `getView().getRoot().getValueInt("numeroNhc")` sin `getRoot()` no accede a la vista y vuelca 0. Esto es porque el dato se recoge de la vista raíz (paciente).

Segundo paso. Añadir en *controladores.xml*:

```

25 <controlador nombre="Problema">
26     <hereda-de controlador="Typical" />
27 </controlador>
28
29 <!-- controlador con acción nueva para Problema, para calcular numeroNhc y ponerlo en el campo nhc de problema.
30 Esta acción llama a la clase java que contiene la acción de volcar valor de vista:
31 clase="org.openxava.cardiologia.actions.CalcularPacienteEnProblema -->
32 <controlador nombre="calcularpacienteproblema">
33     <accion nombre="Añadir" icono="library-plus"
34         clase="org.openxava.cardiologia.actions.CalcularPacienteEnProblema" >
35     </accion>
36 </controlador>
37

```

Captura 254: Captura de pantalla de fragmento de código del script *controladores.xml*.

Tercer paso. Añadir en clase Problema, en el parámetro *nhc* una anotación `@ReadOnly` para que el usuario no pueda modificar la propiedad:

```

84 //añadido 15/06
85 @ReadOnly
86 @Column( name= "nhc", nullable= true)
87 private int nhc;
88

```

```

159 //añadido 15/06
160 public int getNhc() {
161     return nhc;
162 }
163
164 public void setNhc(int nhc) {
165     this.nhc = nhc;
166 }
167

```

Captura 255: Capturas de pantalla de fragmentos de código de la clase *Problema.java*.

Cuarto paso. Añadir en la clase padre (generador), en la colección de problemas la línea:

@NewAction("calcularpacienteproblema.Añadir")

Una vez hecho esto, para poder visualizar los problemas definimos en las vistas de la clase *Paciente.java* dónde se van a ubicar, y también la clase calculada:

```

54 "Sistemas {" +
55 "Generadores { generador }" +
56 "Electrodos { electrodo }" +
57 "ProblemasActivos { problemilla }" +
58 "}" +
59 "Visitas { visita }"

```

Captura 256: Fragmento de código de las vistas de la clase *Paciente.java*.

```

543 //COLECCIONES CALCULADAS
544 //servirá para visualizar los problemas activos que tenga el paciente de sus generadores
545 @Transient
546 @ReadOnly // El miembro nunca será editable por el usuario final en la vista paciente
547 public Collection<Problema> getProblemilla() throws Exception {
548     int numero = (int) getNumeroNhc();
549     Query query = XPersistence.getManager().createQuery
550     ("from Problema p where p.nhc = :numero and p.fechaResolucion = null");
551     //consulta a la BB.DD, y filtra los problemas que no tengan fecha de resolución
552     query.setParameter("numero", numero);
553     if (query.getResultList().size()==0) return Collections.EMPTY_LIST;//si no hay resultados,
554     //se muestra una lista vacía
555     return query.getResultList(); //devuelve la lista de problemas activos
556 }

```

Captura 257: Fragmento de código de la colección calculada de la clase *Paciente.java*.

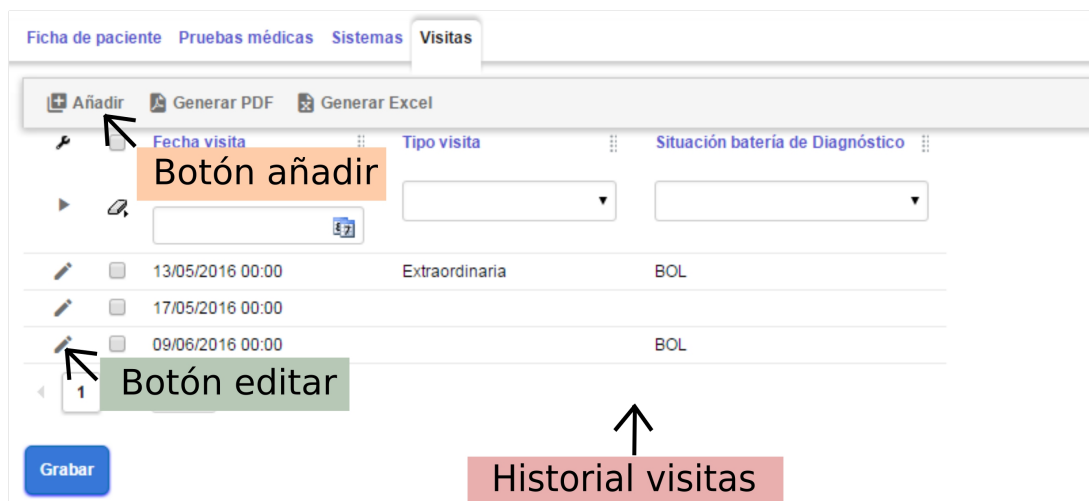
5.9 Sección de visitas.

En esta sección se pueden visualizar las visitas programadas y las no programadas del paciente por motivos cardiológicos, los datos de cada una de éstas, y también modificarlas.

La clase *Visita.java* tiene una relación 1 a 1 con la clase *Diagnostico.java*, y ésta puede tener 0 ó 1 evento médico, 0 ó 1 programación bradicardia, y 0 ó 1 programación taquicardia, implementadas como clases incrustadas.

El usuario puede grabar cada del paciente, visualizar el historial de éstas, los datos de cada una, y también modificarlas.

Para registrar una visita en la BB.DD de cardiología, el usuario dispone de un botón añadir para ello. A través de sencillos formularios se registran y modifican los datos.



Captura 258: Captura de pantalla de la sección Visitas.

Crea una nueva entidad - Visita

▼ Visita

Fecha visita Tipo visita

▼ Diagnóstico

Modo taqui <input type="text"/>	Presencial <input type="text"/>		
Voltaje batería <input type="text"/> V	Situación batería <input type="text"/>	Indicador batería <input type="text"/> %	
Estado Electrodo VD <input type="text"/>	Estimulación VD <input type="text"/> %		
Amplitud VD <input type="text"/> mV	Umbral VD <input type="text"/>	Vrms Impedancia <input type="text"/>	Orniticos <input type="text"/>

▼ Comentario

▼ Tratamiento y Recomendaciones

Tipo agenda No acudio Firma

► Evento médico

► Programación bradicardia

► Programación taquicardia

Captura 259: Captura de pantalla del formulario crear visita.

El botón Grabar cierra el formulario al grabar el objeto visita en la BB.DD, y el botón Grabar y continuar, guarda el objeto y permite salvar uno nuevo. El botón Cerrar no guarda el objeto y cierra el formulario.

Evento médico, programación bradicardia, y taquicardia están minimizados. Si el usuario pulsa en la flecha que está en cada sección, se desplegará su contenido.

Visita

Fecha visita 13/05/2016 00:00

Tipo visita

Extraordinaria

Diagnostico

Modo taqui

Apagado

Presencial

Si

Voltaje bateria

22,00 V

Situacion bateria

BOL

Indicador bateria

32,00 %

Estado Electrodo VD

Estimulacion VD

23,00 %

Amplitud VD

23,00 mV

Umbral VD

23,00 Vrms

Impedancia

23,00 Ohmios

Comentario

2332

Tratamiento y Recomendaciones

XTW r

Tipo agenda

ECAR 79

No acudio

Firma

Jose Ortega Trujillo

Evento médico

Programación bradicardia

Programación taquicardia

Grabar

Cerrar

Captura 260: Captura de pantalla del formulario editar visita.

El botón Grabar cierra el formulario al modificar el objeto visita en la BB.DD, y el botón Cerrar no modifica el objeto y cierra el formulario.

La clase *Visita.java* que describe la colección de visitas que puede tener el paciente se describe a continuación:

```
1 package org.openxava.cardiologia.model;
2
3
4 //librerías para persistencia de los datos
5 import javax.persistence.*;
6
7 //librerías para usar las funciones Openxava
8 import org.openxava.annotations.*;
9 import org.openxava.calculators.*;
10
11 //librería para trabajar con fechas
12 import java.util.*;
13
14 @Entity
15 @View(members= //disposición de los parámetros del dispositivo
16 "Visita [#" +
17 "fechaVisita, tipoVisita;" +
18 "diagnostico;" +
19 "]"
20 )
21
22 //se referencia a la tabla de la BB.DD. llamada visita
23 @Table (name="visita")
24 //hereda de la clase Identificable, para
25 //tener un identificador universal único, que se autogenera con funciones
26 public class Visita extends Identificable{
27
28     //calculador para darle al campo
29     //por defecto la fecha actual
30     @DefaultValueCalculator(CurrentDateCalculator.class)
31     @Column (name="FECHAVISITA")
32     //Anotación Openxava para añadir un calendario
33     @Stereotype("FECHAHORA")
34     // Se mostrará un error de validación si la propiedad se deja en blanco
35     @Required
36     private Date fechaVisita;
37
38     @Column (name="TIPOVISITA")
39     private TipoVisita tipoVisita;
40     //se describe un tipo enumerado que se visualiza como un SELECT
41     public enum TipoVisita {
42         EXTRAORDINARIA ("Extraordinaria"), HOSPITALIZADO("Hospitalizado"),
43         PREALTA("Prealta"), REGULAR("Regular");
44
45         private String nombreTipoVisita;
46
47         private TipoVisita (String nombreTipoVisita){
48             this.nombreTipoVisita = nombreTipoVisita;
49         }
50 }
```

```

51     public String getNombreTipoVisita() {
52         return nombreTipoVisita;
53     }
54
55 }
56
57
58 //Clase incrustada, relación 1 a 1.
59 @OneToOne
60 //se define la clave foránea de la tabla visita
61 @JoinColumn(
62     //columna que corresponde con el id de diagnostico
63     name = "iddiagnostico",
64     nullable = true,
65     foreignKey = @ForeignKey(name = "fk_visita_id"))
66 //fk_visita_id es la clave foránea en la tabla paciente de la BB.DD.
67 @AsEmbedded //Anotación Openxava para incrustar una clase en otra.
68 private Diagnostico diagnostico;
69
70 //REFERENCIAS
71 // La referencia tiene que tener valor siempre. No permite Visita sin Paciente
72 @ManyToOne
73 @JoinColumn( //se define la clave foránea de la tabla visita
74     name="numeroNhc", //columna para la clave foránea
75     nullable = true,
76     foreignKey = @ForeignKey(name = "fk_visita_numeronhc"))
77 //fk_visita_numeronhc es la la clave foránea en la BB.DD de cardiología
78 private Paciente visitasLocal; // Una referencia Java convencional
79
80 //MÉTODOS
81 public Date getFechaVisita() {
82     return fechaVisita;
83 }
84
85 public void setFechaVisita(Date fechaVisita) {
86     this.fechaVisita = fechaVisita;
87 }
88
89 public TipoVisita getTipoVisita() {
90     return tipoVisita;
91 }
92
93 public void setTipoVisita(TipoVisita tipoVisita) {
94     this.tipoVisita = tipoVisita;
95 }
96
97 public Diagnostico getDiagnostico() {
98     return diagnostico;
99 }
100
101 public void setDiagnostico(Diagnostico diagnostico) {
102     this.diagnostico = diagnostico;
103 }
104
105 public Paciente getVisitasLocal() {
106     return visitasLocal;
107 }
108
109 public void setVisitasLocal(Paciente visitasLocal) {
110     this.visitasLocal = visitasLocal;
111 }
112
113 }

```

Captura 261: Capturas de pantalla del código de la clase *Visita.java*.

A su vez, debemos indicar en la clase *Paciente.java* que tiene una relación 1 a muchos con la clase *Visita.java*:

```
536 //se deshabilita la opción borrar desde el formulario de Visita
537 @RemoveAction("")
538 //se deshabilita la opción borrar desde la lista de Visita
539 @RemoveSelectedAction("")
540 //si se borra paciente, se
541 //borrarán sus visitas
542 @OneToMany (mappedBy="visitasLocal", cascade=CascadeType.ALL)
543 //para poder visualizar la lista de visitas
544 private Collection<Visita> visita = new ArrayList<Visita>();
545
```

Captura 262: Captura de pantalla de fragmento de código de la clase *Paciente.java*.

A continuación se muestra el código de la clase *Diagnostico.java*, clase incrustada en la clase *Visita.java*:

```
1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8
9
10 @Entity
11 @View(members= //disposición de los parámetros del diagnóstico
12 " #" +
13 "modoTaqui, presencial;" +
14 "voltajeBateria, situacionBateria, indicadorBateria;" +
15 "estadoElectrodoVd, estimulacionVd;" +
16 "amplitudMvDv, umbralVd, impedancia;" +
17 "comentario, tratamientoYRecomendaciones;" +
18 "tipoAgenda, noAcudio, firma;" +
19 "eventoMedico;" +
20 "programacionBradicardia;" +
21 "programacionTaquicardia;"
22 )
23 //se referencia a la tabla de la BB.DD. llamada diagnostico
24 @Table(name="diagnostico")
25 //hereda de la clase Identificable, para
26 //tener un identificador universal único, que se autogenera con funciones
```

```

27 public class Diagnostico extends Identificable{
28
29     @Column(name="modotaqui")
30     private ModoTaqui modoTaqui;
31     //se describe un tipo enumerado que se visualiza como un SELECT
32     public enum ModoTaqui {
33         APAGADO("Apagado"), MONITOR ("Monitor"),
34         MONITORYTERAPIA("Monitor y Terapia");
35
36         private String nombreModoTaqui;
37
38         private ModoTaqui (String nombreModoTaqui){
39             this.nombreModoTaqui = nombreModoTaqui;
40         }
41
42         public String getNombreModoTaqui() {
43             return nombreModoTaqui;
44         }
45
46     }
47
48
49     @Column(name="situacionbateria")
50     private SituacionBateria situacionBateria;
51     //se describe un tipo enumerado que se visualiza como un SELECT
52     public enum SituacionBateria {
53         BOL ("BOL"), EOL("EOL"), ERI("ERI"), MOL1("MOL1"), MOL2("MOL2");
54
55         private String nombreSituacionBateria;
56

```

```

57         private SituacionBateria (String nombreSituacionBateria){
58             this.nombreSituacionBateria = nombreSituacionBateria;
59         }
60
61         public String getNombreSituacionBateria() {
62             return nombreSituacionBateria;
63         }
64
65     }
66
67     //Anotación Openxava para añadir unidad de medida Voltios.
68     @Stereotype("VOLTIOS")
69     // La longitud de columna se usa a nivel UI y a nivel DB
70     @Column(name="voltajebateria", length=5)
71     private float voltajeBateria;
72
73     //Anotación Openxava para añadir unidad de medida %.
74     @Stereotype("PORCENTAJE")
75     // La longitud de columna se usa a nivel UI y a nivel DB
76     @Column(name="indicadorbateria", length=3)
77     private int indicadorBateria;
78
79     @Column(name="presencial")
80     private Presencial presencial;
81     //se describe un tipo enumerado que se visualiza como un SELECT
82     public enum Presencial {
83         SI ("Sí"), NO("No");
84

```

```

84
85     private String nombrePresencial;
86
87     private Presencial (String nombrePresencial){
88         this.nombrePresencial = nombrePresencial;
89     }
90
91     public String getNombrePresencial() {
92         return nombrePresencial;
93     }
94 }
95
96
97 //Anotación Openxava para añadir unidad de medida mV.
98 @Stereotype("MV")
99 // La longitud de columna se usa a nivel UI y a nivel DB
100 @Column(name="amplitudmvdv", length=5)
101 private int amplitudMvDv;
102
103 //Anotación Openxava para añadir unidad de medida vms.
104 @Stereotype("VMS")
105 // La longitud de columna se usa a nivel UI y a nivel DB
106 @Column(name="umbralvd", length=5)
107 private int umbralVd;
108
109
110 //Anotación Openxava para añadir unidad de medida Ohmios.
111 @Stereotype("OHMIOS")
112 // La longitud de columna se usa a nivel UI y a nivel DB
113 @Column(name="impedancia", length=5)
114 private int impedancia;
115
116 //Anotación Openxava para añadir unidad de medida %.
117 @Stereotype("PORCENTAJE")
118 // La longitud de columna se usa a nivel UI y a nivel DB
119 @Column(name="estimulacionvd", length=5)
120 private int estimulacionVd;
121
122
123 // La longitud de columna se usa a nivel UI y a nivel DB
124 @Column(name="estadoelectrodovd", length=25)
125 private String estadoElectrodoVd;
126
127 @Stereotype("MEMO") // Área de texto o equivalente
128 private String comentario;
129
130 @Column(name="tratamientoyrecomendaciones")
131 @Stereotype("MEMO") // Área de texto o equivalente
132 private String tratamientoYRecomendaciones;
133
134
135 @Column(name="tipoagenda")
136 private TipoAgenda tipoAgenda;
137 //se describe un tipo enumerado que se visualiza como un SELECT
138 public enum TipoAgenda {
139     ECAR79 ("ECAR 79"), EUMI("EUMI"), URG("URG");
140
141     private String nombreTipoAgenda;
142

```

```

143     private TipoAgenda (String nombreTipoAgenda){
144         this.nombreTipoAgenda = nombreTipoAgenda;
145     }
146
147     public String getTipoAgenda() {
148         return nombreTipoAgenda;
149     }
150
151 }
152
153
154 @Column(name="noacudio") //Es un checkbox
155 private Boolean noAcudio;
156
157 @Column (name="FIRMA")
158 private Firma firma;
159 public enum Firma { //se describe un tipo enumerado que se visualiza como un SELECT

```

...se omiten los nombres de los médicos que firman el diagnóstico...

```

201     private String nombreFirma;
202
203     private Firma (String nombreFirma){
204         this.nombreFirma = nombreFirma;
205     }
206
207     public String getNombreFirma() {
208         return nombreFirma;
209     }
210 }
211
212
213 //CLASES INCRUSTABLES
214 //Diagnóstico tiene solo un EventoMedico
215 @Collapsed //Anotación para minimizar el apartado
216 @OneToOne //relación 1 a 1 entre Diagnostico y eventoMedico
217 @JoinColumn( //se define la clave foránea de la tabla diagnostico
218     name = "idevento", //columna que corresponde con el id de evento
219     nullable = true,
220     foreignKey = @ForeignKey(name = "fk_diagnostico_id"))
221     //fk_diagnostico_id es la clave foránea en la tabla diagnostico de la BB.DD.
222 @AsEmbedded //Anotación Openxava para incrustar una clase en otra.
223 private EventoMedico eventoMedico;
224
225 //Diagnóstico tiene solo una programación bradicardia
226 @Collapsed //Anotación para minimizar el apartado
227 @OneToOne //relación 1 a 1 entre Diagnostico y programacionBradicardia
228 @JoinColumn( //se define la clave foránea de la tabla diagnostico
229     name = "idpb", //columna que corresponde con el id de programacionBradicardia
230     nullable = true,
231     foreignKey = @ForeignKey(name = "fk_diagnosticopb_id"))

```

```

232     //fk_diagnosticopb_id es la clave foránea en la tabla diagnostico de la BB.DD.
233 @AsEmbedded //Anotación Openxava para incrustar una clase en otra.
234 private ProgramacionBradicardia programacionBradicardia;
235
236 //Diagnóstico tiene solo una programación taquicardia
237 @Collapsed //Anotación para minimizar el apartado
238 @OneToOne //relación 1 a 1 entre Diagnostico y programacionTaquicardia
239 @JoinColumn( //se define la clave foránea de la tabla diagnostico
240     name = "idpt", //columna que corresponde con el id de programacionTaquicardia
241     nullable = true,
242     foreignKey = @ForeignKey(name = "fk_diagnosticopt_id"))
243     //fk_diagnosticopt_id es la clave foránea en la tabla diagnostico de la BB.DD.
244 @AsEmbedded //Anotación Openxava para incrustar una clase en otra.
245 private ProgramacionTaquicardia programacionTaquicardia;
246

```



```

248 //MÉTODOS
249 public ModoTaqui getModoTaqui() {
250     return modoTaqui;
251 }
252 public void setModoTaqui(ModoTaqui modoTaqui) {
253     this.modoTaqui = modoTaqui;
254 }
255 public SituacionBateria getSituacionBateria() {
256     return situacionBateria;
257 }
258 public void setSituacionBateria(SituacionBateria situacionBateria) {
259     this.situacionBateria = situacionBateria;
260 }
261 public float getVoltajeBateria() {
262     return voltajeBateria;
263 }
264 public void setVoltajeBateria(float voltajeBateria) {
265     this.voltajeBateria = voltajeBateria;
266 }
267 public int getIndicadorBateria() {
268     return indicadorBateria;
269 }
270 public void setIndicadorBateria(int indicadorBateria) {
271     this.indicadorBateria = indicadorBateria;
272 }
273 public Presencial getPresencial() {
274     return presencial;
275 }

```

```

276 public void setPresencial(Presencial presencial) {
277     this.presencial = presencial;
278 }
279 public int getAmplitudMvDv() {
280     return amplitudMvDv;
281 }
282 public void setAmplitudMvDv(int amplitudMvDv) {
283     this.amplitudMvDv = amplitudMvDv;
284 }
285 public int getUmbralVd() {
286     return umbralVd;
287 }
288 public void setUmbralVd(int umbralVd) {
289     this.umbralVd = umbralVd;
290 }
291 public int getImpedancia() {
292     return impedancia;
293 }
294 public void setImpedancia(int impedancia) {
295     this.impedancia = impedancia;
296 }
297 public int getEstimulacionVd() {
298     return estimulacionVd;
299 }
300 public void setEstimulacionVd(int estimulacionVd) {
301     this.estimulacionVd = estimulacionVd;
302 }

```

```

303 public String getEstadoElectrodoVd() {
304     return estadoElectrodoVd;
305 }
306 public void setEstadoElectrodoVd(String estadoElectrodoVd) {
307     this.estadoElectrodoVd = estadoElectrodoVd;
308 }
309 public String getComentario() {
310     return comentario;
311 }
312 public void setComentario(String comentario) {
313     this.comentario = comentario;
314 }
315 public String getTratamientoYRecomendaciones() {
316     return tratamientoYRecomendaciones;
317 }
318 public void setTratamientoYRecomendaciones(String tratamientoYRecomendaciones) {
319     this.tratamientoYRecomendaciones = tratamientoYRecomendaciones;
320 }
321 public TipoAgenda getTipoAgenda() {
322     return tipoAgenda;
323 }
324 public void setTipoAgenda(TipoAgenda tipoAgenda) {
325     this.tipoAgenda = tipoAgenda;
326 }
327 public Boolean getNoAcudio() {
328     return noAcudio;
329 }

```

```

330 public void setNoAcudio(Boolean noAcudio) {
331     this.noAcudio = noAcudio;
332 }
333
334 public Firma getFirma() {
335     return firma;
336 }
337 public void setFirma(Firma firma) {
338     this.firma = firma;
339 }
340 public EventoMedico getEventoMedico() {
341     return eventoMedico;
342 }
343 public void setEventoMedico(EventoMedico eventoMedico) {
344     this.eventoMedico = eventoMedico;
345 }
346 public ProgramacionBradicardia getProgramacionBradicardia() {
347     return programacionBradicardia;
348 }
349 public void setProgramacionBradicardia(ProgramacionBradicardia programacionBradicardia) {
350     this.programacionBradicardia = programacionBradicardia;
351 }
352 public ProgramacionTaquicardia getProgramacionTaquicardia() {
353     return programacionTaquicardia;
354 }
355 public void setProgramacionTaquicardia(ProgramacionTaquicardia programacionTaquicardia) {
356     this.programacionTaquicardia = programacionTaquicardia;
357 }
358
359 }

```

Capturas 263: Capturas de pantalla del código de la clase *Diagnostico.java*.

Al desplegar evento médico, tendremos el siguiente formulario incrustado en el diagnóstico de la visita:

▼ Evento médico

Fibrilación auricular	FA Permanente ▼	Taquicardia ventricular	TVMS ▼	Otras arritmias	<input type="text"/>
Descargas	<input type="text"/>	Comentario descargas	<input type="text"/>		
ATP	<input type="text"/>	Comentario ATP	<input type="text"/>		
Reinicio contadores	<input type="text"/>	Episodios FV	<input type="text"/>	Episodios TV	<input type="text"/>
Episodios TV-1	<input type="text"/>	Episodios TNVS	<input type="text"/>	Cambios RTA	<input type="text"/>
Episodios FA	<input type="text"/>	Episodios TSV	<input type="text"/>		

Captura 264: Captura de pantalla del formulario Evento médico.

La clase incrustada *EventoMedico.java* y sus parámetros se describe en el siguiente código:

```

1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8
9 @Entity
10 @View(members= //disposición de los parámetros del evento médico
11 " #" +
12 "fibrilacionAuricular, taquicardiaVentricular, otrasArritmias;" +
13 "descargas, comentarioDescargas;" +
14 "atp, comentarioAtp;" +
15 "reinicioContadores, episodiosFv, episodiosTv;" +
16 "episodiosTvl, episodiosTnvs, cambiosRta;" +
17 "episodiosFa, episodiosTsv;"
18 )
19 //se referencia a la tabla de la BB.DD. llamada eventomedico
20 @Table(name="eventomedico")
21 //hereda de la clase Identificable, para
22 //tener un identificador universal único, que se autogenera con funciones
23 public class EventoMedico extends Identificable{
24

```

```

25
26 //PROPIEDADES
27 @Column(name="fibrilacionauricular")
28 private FibrilacionAuricular fibrilacionAuricular;
29 //se describe un tipo enumerado que se visualiza como un SELECT
30 public enum FibrilacionAuricular {
31     FAPAROXISTICA ("FA Paroxística"), FAPERMANENTE("FA Permanente"),
32     FAPER SISTENTE("FA Persistente"), NO("No");
33
34     private String nombreFibrilacionAuricular;
35
36     private FibrilacionAuricular (String nombreFibrilacionAuricular){
37         this.nombreFibrilacionAuricular = nombreFibrilacionAuricular;
38     }
39
40     public String getNombreFibrilacionAuricular() {
41         return nombreFibrilacionAuricular;
42     }
43
44 }

```

```

45
46 @Column(name="taquicardiaventricular")
47 private TaquicardiaVentricular taquicardiaVentricular;
48 //se describe un tipo enumerado que se visualiza como un SELECT
49 public enum TaquicardiaVentricular {
50     NO("No"), TVMS("TVMS"), TVNS("TVNS"), TVPS("TVPS");
51
52     private String nombreTaquicardiaVentricular;
53
54     private TaquicardiaVentricular (String nombreTaquicardiaVentricular){
55         this.nombreTaquicardiaVentricular = nombreTaquicardiaVentricular;
56     }
57
58     public String getNombreTaquicardiaVentricular() {
59         return nombreTaquicardiaVentricular;
60     }
61
62 }
63
64 // La longitud de columna se usa a nivel UI y a nivel DB
65 @Column(name="otrasarritmias", length=30)
66 private String otrasArritmias;
67
68 @Column(name="descargas")
69 private Descargas descargas;
70 //se describe un tipo enumerado que se visualiza como un SELECT
71 public enum Descargas {
72     APROPIADAEFICAZ("Apropiada Eficaz"),
73     APROPIADAINEFICAZ("Apropiada Ineficaz"),
74     INAPROPIADAEFICAZ("Inapropiada Eficaz"),
75     INAPROPIADAINEFICAZ("Inapropiada Ineficaz");

```

```

76
77     private String nombreDescargas;
78
79     private Descargas (String nombreDescargas){
80         this.nombreDescargas = nombreDescargas;
81     }
82
83     public String getNombreDescargas() {
84         return nombreDescargas;
85     }
86
87 }
88
89 // La longitud de columna se usa a nivel UI y a nivel DB
90 @Column(name="comentariodescargas", length=70)
91 private String comentarioDescargas;
92
93 @Column(name="atp")
94 private Atp atp;
95 //se describe un tipo enumerado que se visualiza como un SELECT
96 public enum Atp {
97     APROPIADAEFICAZ("Apropiada Eficaz"),
98     APROPIADAINEFICAZ("Apropiada Ineficaz"),
99     INAPROPIADAEFICAZ("Inapropiada Eficaz"),
100     INAPROPIADAINEFICAZ("Inapropiada Ineficaz");
101
102     private String nombreAtp;
103

```

```

104     private Atp (String nombreAtp){
105         this.nombreAtp = nombreAtp;
106     }
107
108     public String getNombreAtp() {
109         return nombreAtp;
110     }
111 }
112
113 // La longitud de columna se usa a nivel UI y a nivel DB
114 @Column(name="comentarioatp", length=70)
115 private String comentarioAtp;
116
117 // La longitud de columna se usa a nivel UI y a nivel DB
118 @Column(name="reiniciocontadores", length=20)
119 private String reinicioContadores;
120
121 // La longitud de columna se usa a nivel UI y a nivel DB
122 @Column(name="episodiosfv", length=20)
123 private String episodiosFv;
124
125 // La longitud de columna se usa a nivel UI y a nivel DB
126 @Column(name="episodiosTv", length=20)
127 private String episodiosTv;
128
129 // La longitud de columna se usa a nivel UI y a nivel DB
130 @Column(name="episodiosTv1", length=20)
131 private String episodiosTv1;
132

```

```

133 // La longitud de columna se usa a nivel UI y a nivel DB
134 @Column(name="episodiosTnvs", length=20)
135 private String episodiosTnvs;
136
137 // La longitud de columna se usa a nivel UI y a nivel DB
138 @Column(name="cambiosrta", length=20)
139 private String cambiosRta;
140
141 // La longitud de columna se usa a nivel UI y a nivel DB
142 @Column(name="episodiosfa", length=20)
143 private String episodiosFa;
144
145 // La longitud de columna se usa a nivel UI y a nivel DB
146 @Column(name="episodiosTsv", length=20)
147 private String episodiosTsv;
148
149
150 //MÉTODOS
151 public FibrilacionAuricular getFibrilacionAuricular() {
152     return fibrilacionAuricular;
153 }
154
155 public void setFibrilacionAuricular(FibrilacionAuricular fibrilacionAuricular) {
156     this.fibrilacionAuricular = fibrilacionAuricular;
157 }
158
159 public TaquicardiaVentricular getTaquicardiaVentricular() {
160     return taquicardiaVentricular;
161 }

```

```

162
163 public void setTaquicardiaVentricular(TaquicardiaVentricular taquicardiaVentricular) {
164     this.taquicardiaVentricular = taquicardiaVentricular;
165 }
166
167 public String getOtrasArritmias() {
168     return otrasArritmias;
169 }
170
171 public void setOtrasArritmias(String otrasArritmias) {
172     this.otrasArritmias = otrasArritmias;
173 }
174
175 public Descargas getDescargas() {
176     return descargas;
177 }
178
179 public void setDescargas(Descargas descargas) {
180     this.descargas = descargas;
181 }
182
183 public String getComentarioDescargas() {
184     return comentarioDescargas;
185 }
186
187 public void setComentarioDescargas(String comentarioDescargas) {
188     this.comentarioDescargas = comentarioDescargas;
189 }
190

```

```

191 public Atp getAtp() {
192     return atp;
193 }
194
195 public void setAtp(Atp atp) {
196     this.atp = atp;
197 }
198
199 public String getComentarioAtp() {
200     return comentarioAtp;
201 }
202
203 public void setComentarioAtp(String comentarioAtp) {
204     this.comentarioAtp = comentarioAtp;
205 }
206
207 public String getReinicioContadores() {
208     return reinicioContadores;
209 }
210
211 public void setReinicioContadores(String reinicioContadores) {
212     this.reinicioContadores = reinicioContadores;
213 }
214
215 public String getEpisodiosFv() {
216     return episodiosFv;
217 }
218
219 public void setEpisodiosFv(String episodiosFv) {
220     this.episodiosFv = episodiosFv;
221 }
222

```

```

223 public String getEpisodiosTv() {
224     return episodiosTv;
225 }
226
227 public void setEpisodiosTv(String episodiosTv) {
228     this.episodiosTv = episodiosTv;
229 }
230
231 public String getEpisodiosTv1() {
232     return episodiosTv1;
233 }
234
235 public void setEpisodiosTv1(String episodiosTv1) {
236     this.episodiosTv1 = episodiosTv1;
237 }
238
239 public String getEpisodiosTnvs() {
240     return episodiosTnvs;
241 }
242
243 public void setEpisodiosTnvs(String episodiosTnvs) {
244     this.episodiosTnvs = episodiosTnvs;
245 }
246
247 public String getCambiosRta() {
248     return cambiosRta;
249 }

```

```

250
251 public void setCambiosRta(String cambiosRta) {
252     this.cambiosRta = cambiosRta;
253 }
254
255 public String getEpisodiosFa() {
256     return episodiosFa;
257 }
258
259 public void setEpisodiosFa(String episodiosFa) {
260     this.episodiosFa = episodiosFa;
261 }
262
263 public String getEpisodiosTsv() {
264     return episodiosTsv;
265 }
266
267 public void setEpisodiosTsv(String episodiosTsv) {
268     this.episodiosTsv = episodiosTsv;
269 }
270
271 }
272

```

Capturas 267: Capturas de pantalla del código de la clase *EventoMedico.java*.

Al desplegar Programación bradicardia, tendremos el siguiente formulario incrustado en el diagnóstico de la visita:

▼ Programación bradicardia			
Modo estimulación	AAI ▼		
LIF	44	LMS	44 LSF 33
Tipo estimulación ventricular	BIV ▼	Retraso ventrículos	44
Intervalo AV estimulación	443	Intervalo AV detección	33
Ciclos búsqueda AV	33	Histéresis AV	33
Sensibilidad VD	44	Salida VD	55 Config. Estimulación/Detección VD 7777

Captura 268: Captura de pantalla del formulario Programación Bradicardia.

La clase incrustada *ProgramaciónBradocardia.java* y sus parámetros se describe en el siguiente código:

```

1 package org.openxava.cardiologia.model;
2
3 //librerías para persistencia de los datos
4 import javax.persistence.*;
5
6 //librerías para usar las funciones Openxava
7 import org.openxava.annotations.*;
8
9 @Entity
10 @View(members= //disposición de los parámetros de la prog. bradicardia
11 "#" +
12 "modoEstimulacion;" +
13 "lift, lms, lsf;" +
14 "tipoEstimulacionVentricular, retrasoVentriculos;" +
15 "intervaloAvEstimulacion, intervaloAvDeteccion;" +
16 "ciclosBusquedaAv, histeresisAv;" +
17 "sensibilidadVd, salidaVd, configuracionEstiDeteccionVd"
18 )
19 //se referencia a la tabla de la BB.DD. llamada programacionbradicardia
20 @Table (name="programacionbradicardia")
21 //hereda de la clase Identificable, para
22 //tener un identificador universal único, que se autogenera con funciones
23 public class ProgramacionBradocardia extends Identificable{

```



```

24
25 //PROPIEDADES
26 @Column(name="modoestimacion")
27 private ModoEstimulacion modoEstimulacion;
28 //se describe un tipo enumerado que se visualiza como un SELECT
29 public enum ModoEstimulacion {
30     AAI ("AAI"), AAT("AAT"), AOO ("AOO"),
31     DDD("DDD"), DDI("DDI"), DOO("DOO"),
32     DVI("DVI"), ODO("ODO"), OOO("OOO"),
33     VDD("VDD"), VOO("VOO"), VVI("VVI");
34
35     private String nombreModoEstimulacion;
36
37     private ModoEstimulacion (String nombreModoEstimulacion){
38         this.nombreModoEstimulacion = nombreModoEstimulacion;
39     }

```

```

40
41     public String getNombreModoEstimulacion() {
42         return nombreModoEstimulacion;
43     }
44
45 }
46
47 // La longitud de columna se usa a nivel UI y a nivel DB
48 @Column(name="lift", length=4)
49 private int lift;
50
51 // La longitud de columna se usa a nivel UI y a nivel DB
52 @Column(name="lms", length=4)
53 private int lms;
54
55 // La longitud de columna se usa a nivel UI y a nivel DB
56 @Column(name="lsf", length=4)
57 private int lsf;
58
59 // La longitud de columna se usa a nivel UI y a nivel DB
60 @Column(name="tipoestimulacionventricular", length=4)
61 private TipoEstimulacionVentricular tipoEstimulacionVentricular;
62 //se describe un tipo enumerado que se visualiza como un SELECT
63 public enum TipoEstimulacionVentricular {
64     BIV ("BIV"), LV("LV"), VD ("VD");
65
66     private String nombreTipoEstimulacionVentricular;

```

```

67
68     private TipoEstimulacionVentricular (String nombreTipoEstimulacionVentricular){
69         this.nombreTipoEstimulacionVentricular = nombreTipoEstimulacionVentricular;
70     }
71
72     public String getNombreTipoEstimulacionVentricular() {
73         return nombreTipoEstimulacionVentricular;
74     }
75
76 }
77
78 // La longitud de columna se usa a nivel UI y a nivel DB
79 @Column(name="retrasoventriculos", length=4)
80 private int retrasoVentriculos;
81
82 // La longitud de columna se usa a nivel UI y a nivel DB
83 @Column(name="intervaloavestimulacion", length=4)
84 private int intervaloAvEstimulacion;
85
86
87 // La longitud de columna se usa a nivel UI y a nivel DB
88 @Column(name="intervaloavdeteccion", length=4)
89 private int intervaloAvDeteccion;
90
91 // La longitud de columna se usa a nivel UI y a nivel DB
92 @Column(name="ciclosbusquedaav", length=4)
93 private int ciclosBusquedaAv;

```

```

94
95 // La longitud de columna se usa a nivel UI y a nivel DB
96 @Column(name="histeresisav", length=4)
97 private int histeresisAv;
98
99 // La longitud de columna se usa a nivel UI y a nivel DB
100 @Column(name="sensibilidadvd", length=4)
101 private int sensibilidadVd;
102
103 // La longitud de columna se usa a nivel UI y a nivel DB
104 @Column(name="salidavd", length=4)
105 private int salidaVd;
106
107 // La longitud de columna se usa a nivel UI y a nivel DB
108 @Column(name="configuracionestideteccionvd", length=4)
109 private int configuracionEstiDeteccionVd;

```

```

111
112 //MÉTODOS
113 public ModoEstimulacion getModoEstimulacion() {
114     return modoEstimulacion;
115 }
116
117 public void setModoEstimulacion(ModoEstimulacion modoEstimulacion) {
118     this.modoEstimulacion = modoEstimulacion;
119 }
120
121 public int getLift() {
122     return lift;
123 }
124
125 public void setLift(int lift) {
126     this.lift = lift;
127 }
128
129 public int getLms() {
130     return lms;
131 }
132
133 public void setLms(int lms) {
134     this.lms = lms;
135 }
136

```

```

137 public int getLsf() {
138     return lsf;
139 }
140
141 public void setLsf(int lsf) {
142     this.lsf = lsf;
143 }
144
145 public TipoEstimulacionVentricular getTipoEstimulacionVentricular() {
146     return tipoEstimulacionVentricular;
147 }
148
149 public void setTipoEstimulacionVentricular(TipoEstimulacionVentricular tipoEstimulacionVentricular) {
150     this.tipoEstimulacionVentricular = tipoEstimulacionVentricular;
151 }
152
153 public int getRetrasoVentriculos() {
154     return retrasoVentriculos;
155 }
156
157 public void setRetrasoVentriculos(int retrasoVentriculos) {
158     this.retrasoVentriculos = retrasoVentriculos;
159 }

```

```

160
161 public int getIntervaloAvEstimulacion() {
162     return intervaloAvEstimulacion;
163 }
164
165 public void setIntervaloAvEstimulacion(int intervaloAvEstimulacion) {
166     this.intervaloAvEstimulacion = intervaloAvEstimulacion;
167 }
168
169 public int getIntervaloAvDeteccion() {
170     return intervaloAvDeteccion;
171 }
172
173 public void setIntervaloAvDeteccion(int intervaloAvDeteccion) {
174     this.intervaloAvDeteccion = intervaloAvDeteccion;
175 }
176
177 public int getCiclosBusquedaAv() {
178     return ciclosBusquedaAv;
179 }

```

```

180
181 public void setCiclosBusquedaAv(int ciclosBusquedaAv) {
182     this.ciclosBusquedaAv = ciclosBusquedaAv;
183 }
184
185 public int getHisteresisAv() {
186     return histeresisAv;
187 }
188
189 public void setHisteresisAv(int histeresisAv) {
190     this.histeresisAv = histeresisAv;
191 }
192
193 public int getSensibilidadVd() {
194     return sensibilidadVd;
195 }
196
197 public void setSensibilidadVd(int sensibilidadVd) {
198     this.sensibilidadVd = sensibilidadVd;
199 }
200
201 public int getSalidaVd() {
202     return salidaVd;
203 }

```

```

204
205 public void setSalidaVd(int salidaVd) {
206     this.salidaVd = salidaVd;
207 }
208
209 public int getConfiguracionEstiDeteccionVd() {
210     return configuracionEstiDeteccionVd;
211 }
212
213 public void setConfiguracionEstiDeteccionVd(int configuracionEstiDeteccionVd) {
214     this.configuracionEstiDeteccionVd = configuracionEstiDeteccionVd;
215 }
216 }
217

```

Capturas 269: Capturas de pantalla del código de la clase *ProgramacionBradycardia.java*.

Al desplegar Programación taquicardia, tendremos el siguiente formulario incrustado en el diagnóstico de la visita:

▼ Programación taquicardia

Zona TV-1	<input type="text" value="222"/>	Zona TV	<input type="text" value="5"/>	Zona FV	<input type="text" value="4"/>
ATP TV-1	<input type="text" value="No"/>	ATP TV	<input type="text" value="Rampa-Scan"/>		
Núm. ráfagas ATP1 TV-1	<input type="text" value="2"/>	Núm. ráfagas ATP1 TV	<input type="text" value="4"/>		
ATP2 TV-1	<input type="text" value="232"/>	ATP2 TV	<input type="text" value="0"/>		
Núm. ráfagas ATP2 TV-1	<input type="text" value="234"/>	Núm. ráfagas ATP2 TV	<input type="text" value="4"/>		
Descarga TV-1	<input type="text" value="423"/>	Descarga TV	<input type="text" value="4"/>	Descarga FV	<input type="text" value="4"/>
Descarga 2 TV-1	<input type="text" value="342"/>	Descarga 2 TV	<input type="text" value="4"/>	Descarga 2 FV	<input type="text" value="444"/>
Máx. energía TV-1	<input type="text" value="234"/>	Máx. energía TV	<input type="text" value="4"/>	Máx. energía FV	<input type="text" value="4"/>
Prórroga TV-1	<input type="text" value="543"/>	Prórroga TV	<input type="text" value="0"/>	Prórroga FV	<input type="text" value="4"/>
Modo taqui	<input type="text" value="Apagado"/>				

▼ Comentario programación taquicardia

fedds

Captura 270: Captura de pantalla del formulario Programación Taquicardia.

La clase incrustada *ProgramaciónTaquicardia.java* y sus parámetros se describe en el siguiente código:

```

1  package org.openxava.cardiologia.model;
2
3  //librerías para persistencia de los datos
4  import javax.persistence.*;
5
6  //librerías para usar las funciones Openxava
7  import org.openxava.annotations.*;
8
9  @Entity
10 //disposición de los parámetros de la prog. taquicardia
11 @View (members=
12 " #" +
13 "zonaTv1, zonaTv, zonaFv;" +
14 "atpTv1, atpTv;" +
15 "numRafagasAtp1Tv1, numRafagasAtp1Tv;" +
16 "atp2Tv1, atp2Tv;" +
17 "numRafagasAtp2Tv1, numRafagasAtp2Tv;" +
18 "descargaTv1, descargaTv, descargaFv;" +
19 "descarga2Tv1, descarga2Tv, descarga2Fv;" +
20 "maxEnergiaTv1, maxEnergiaTv, maxEnergiaFv;" +
21 "prorrogaTv1, prorrogaTv, prorrogaFv;" +
22 "modoTaqui;" +
23 "comentarioProgramacionTaquicardia;"
24 )
25 //se referencia a la tabla de la BB.DD. llamada programaciontaquicardia
26 @Table (name="programaciontaquicardia")
27 //hereda de la clase Identificable, para
28 //tener un identificador universal único, que se autogenera con funciones
29 public class ProgramacionTaquicardia extends Identificable{
30

```

```

31 //PROPIEDADES
32 @Column(name="zonatv1")
33 private int zonaTv1;
34
35 @Column(name="atptv1")
36 private AtpTv1 atpTv1;
37 //se describe un tipo enumerado que se visualiza como un SELECT
38 public enum AtpTv1 {
39     NO ("No"), RAMPA("Rampa"),
40     RAMPASCAN("Rampa-Scan"), SCAN("Scan");
41
42     private String nombreAtpTv1;
43
44     private AtpTv1 (String nombreAtpTv1){
45         this.nombreAtpTv1 = nombreAtpTv1;
46     }
47
48     public String getNombreAtpTv1() {
49         return nombreAtpTv1;
50     }
51
52 }
53
54 // La longitud de columna se usa a nivel UI y a nivel DB
55 @Column(name="numrafagasatp1tv1", length=5)
56 private int numRafagasAtp1Tv1;
57
58 // La longitud de columna se usa a nivel UI y a nivel DB
59 @Column(name="atp2tv1", length=5)
60 private int atp2Tv1;
61
62 // La longitud de columna se usa a nivel UI y a nivel DB
63 @Column(name="numrafagasatp2tv1", length=5)
64 private int numRafagasAtp2Tv1;
65
66 // La longitud de columna se usa a nivel UI y a nivel DB
67 @Column(name="descargatv1", length=5)
68 private int descargaTv1;
69
70 // La longitud de columna se usa a nivel UI y a nivel DB
71 @Column(name="descarga2tv1", length=5)
72 private int descarga2Tv1;
73
74 // La longitud de columna se usa a nivel UI y a nivel DB
75 @Column(name="maxenergiatv1", length=5)
76 private int maxEnergiaTv1;
77
78 // La longitud de columna se usa a nivel UI y a nivel DB
79 @Column(name="prorroгатv1", length=5)
80 private int prorrogatv1;
81
82 // La longitud de columna se usa a nivel UI y a nivel DB
83 @Column(name="zonatv", length=5)
84 private int zonaTv;
85
86 // La longitud de columna se usa a nivel UI y a nivel DB
87 @Column(name="atptv")
88 private AtpTv atpTv;
89 //se describe un tipo enumerado que se visualiza como un SELECT
90 public enum AtpTv {
91     NO ("No"), RAMPA("Rampa"),
92     RAMPASCAN("Rampa-Scan"), SCAN("Scan");

```

```

93
94     private String nombreAtpTv;
95
96     private AtpTv (String nombreAtpTv){
97         this.nombreAtpTv = nombreAtpTv;
98     }
99
100    public String getNombreAtpTv() {
101        return nombreAtpTv;
102    }
103
104 }
105
106
107 // La longitud de columna se usa a nivel UI y a nivel DB
108 @Column(name="numrafagasatpltv", length=5)
109 private int numRafagasAtp1Tv;
110
111 @Column(name="atp2tv", length=5)
112 private int atp2Tv;
113
114 // La longitud de columna se usa a nivel UI y a nivel DB
115 @Column(name="numrafagasatp2tv", length=5)
116 private int numRafagasAtp2Tv;
117
118 // La longitud de columna se usa a nivel UI y a nivel DB
119 @Column(name="descargatv", length=5)
120 private int descargaTv;
121
122 // La longitud de columna se usa a nivel UI y a nivel DB
123 @Column(name="descarga2tv", length=5)
124 private int descarga2Tv;
125
126 // La longitud de columna se usa a nivel UI y a nivel DB
127 @Column(name="maxenergiatv", length=5)
128 private int maxEnergiaTv;
129
130 // La longitud de columna se usa a nivel UI y a nivel DB
131 @Column(name="prorrogatv", length=5)
132 private int prorrogaTv;
133
134 // La longitud de columna se usa a nivel UI y a nivel DB
135 @Column(name="zonafv", length=5)
136 private int zonaFv;
137
138 // La longitud de columna se usa a nivel UI y a nivel DB
139 @Column(name="descargafv", length=5)
140 private int descargaFv;
141
142 // La longitud de columna se usa a nivel UI y a nivel DB
143 @Column(name="descarga2fv", length=5)
144
145 // La longitud de columna se usa a nivel UI y a nivel DB
146 @Column(name="maxenergiafv", length=5)
147 private int maxEnergiaFv;
148
149 // La longitud de columna se usa a nivel UI y a nivel DB
150 @Column(name="prorrogafv", length=5)
151 private int prorrogaFv;
152

```

```

153     @Column(name="modotaqui")
154     private ModoTaqui modoTaqui;
155     //se describe un tipo enumerado que se visualiza como un SELECT
156     public enum ModoTaqui {
157         APAGADO("Apagado"), MONITOR ("Monitor"),
158         MONITORYTERAPIA("Monitor y Terapia");
159
160         private String nombreModoTaqui;
161
162         private ModoTaqui (String nombreModoTaqui){
163             this.nombreModoTaqui = nombreModoTaqui;
164         }
165
166         public String getNombreModoTaqui() {
167             return nombreModoTaqui;
168         }
169     }
170
171
172     @Column(name="comentarioprogramaciontaquicardia")
173     @Stereotype("MEMO") //Área de texto o equivalente
174     private String comentarioProgramacionTaquicardia;
175
176
177     //METODOS
178     public int getZonaTv1() {
179         return zonaTv1;
180     }
181
182     public void setZonaTv1(int zonaTv1) {
183         this.zonaTv1 = zonaTv1;
184     }
185
186     public AtpTv1 getAtpTv1() {
187         return atpTv1;
188     }
189
190     public void setAtpTv1(AtpTv1 atpTv1) {
191         this.atpTv1 = atpTv1;
192     }
193
194     public int getNumRafagasAtp1Tv1() {
195         return numRafagasAtp1Tv1;
196     }
197
198     public void setNumRafagasAtp1Tv1(int numRafagasAtp1Tv1) {
199         this.numRafagasAtp1Tv1 = numRafagasAtp1Tv1;
200     }
201
202     public int getAtp2Tv1() {
203         return atp2Tv1;
204     }
205
206     public void setAtp2Tv1(int atp2Tv1) {
207         this.atp2Tv1 = atp2Tv1;
208     }
209
210     public int getNumRafagasAtp2Tv1() {
211         return numRafagasAtp2Tv1;
212     }
213

```



```

214 public void setNumRafagasAtp2Tv1(int numRafagasAtp2Tv1) {
215     this.numRafagasAtp2Tv1 = numRafagasAtp2Tv1;
216 }
217
218 public int getDescargaTv1() {
219     return descargaTv1;
220 }
221
222 public void setDescargaTv1(int descargaTv1) {
223     this.descargaTv1 = descargaTv1;
224 }
225
226 public int getDescarga2Tv1() {
227     return descarga2Tv1;
228 }
229
230 public void setDescarga2Tv1(int descarga2Tv1) {
231     this.descarga2Tv1 = descarga2Tv1;
232 }
233
234 public int getMaxEnergiaTv1() {
235     return maxEnergiaTv1;
236 }
237
238 public void setMaxEnergiaTv1(int maxEnergiaTv1) {
239     this.maxEnergiaTv1 = maxEnergiaTv1;
240 }
241

```

```

242 public int getProrrogaTv1() {
243     return prorrogaTv1;
244 }
245
246 public void setProrrogaTv1(int prorrogaTv1) {
247     this.prorrogaTv1 = prorrogaTv1;
248 }
249
250 public int getZonaTv() {
251     return zonaTv;
252 }
253
254 public void setZonaTv(int zonaTv) {
255     this.zonaTv = zonaTv;
256 }
257
258 public AtpTv getAtpTv() {
259     return atpTv;
260 }
261
262 public void setAtpTv(AtpTv atpTv) {
263     this.atpTv = atpTv;
264 }
265
266 public int getNumRafagasAtp1Tv() {
267     return numRafagasAtp1Tv;
268 }
269
270 public void setNumRafagasAtp1Tv(int numRafagasAtp1Tv) {
271     this.numRafagasAtp1Tv = numRafagasAtp1Tv;
272 }

```

```
273
274 public int getAtp2Tv() {
275     return atp2Tv;
276 }
277
278 public void setAtp2Tv(int atp2Tv) {
279     this.atp2Tv = atp2Tv;
280 }
281
282 public int getNumRafagasAtp2Tv() {
283     return numRafagasAtp2Tv;
284 }
285
286 public void setNumRafagasAtp2Tv(int numRafagasAtp2Tv) {
287     this.numRafagasAtp2Tv = numRafagasAtp2Tv;
288 }
289
290 public int getDescargaTv() {
291     return descargaTv;
292 }
293
294 public void setDescargaTv(int descargaTv) {
295     this.descargaTv = descargaTv;
296 }
297
298 public int getDescarga2Tv() {
299     return descarga2Tv;
300 }
301
```

```

302 public void setDescarga2Tv(int descarga2Tv) {
303     this.descarga2Tv = descarga2Tv;
304 }
305
306 public int getMaxEnergiaTv() {
307     return maxEnergiaTv;
308 }
309
310 public void setMaxEnergiaTv(int maxEnergiaTv) {
311     this.maxEnergiaTv = maxEnergiaTv;
312 }
313
314 public int getProrrogaTv() {
315     return prorrogaTv;
316 }
317
318 public void setProrrogaTv(int prorrogaTv) {
319     this.prorrogaTv = prorrogaTv;
320 }
321
322 public int getZonaFv() {
323     return zonaFv;
324 }
325
326 public void setZonaFv(int zonaFv) {
327     this.zonaFv = zonaFv;
328 }
329
330 public int getDescargaFv() {
331     return descargaFv;
332 }

```

```

333
334 public void setDescargaFv(int descargaFv) {
335     this.descargaFv = descargaFv;
336 }
337
338 public int getDescarga2Fv() {
339     return descarga2Fv;
340 }
341
342 public void setDescarga2Fv(int descarga2Fv) {
343     this.descarga2Fv = descarga2Fv;
344 }
345
346 public int getMaxEnergiaFv() {
347     return maxEnergiaFv;
348 }
349
350 public void setMaxEnergiaFv(int maxEnergiaFv) {
351     this.maxEnergiaFv = maxEnergiaFv;
352 }
353
354 public int getProrrogaFv() {
355     return prorrogaFv;
356 }
357
358 public void setProrrogaFv(int prorrogaFv) {
359     this.prorrogaFv = prorrogaFv;
360 }

```

```

361
362     public ModoTaqui getModoTaqui() {
363         return modoTaqui;
364     }
365
366     public void setModoTaqui(ModoTaqui modoTaqui) {
367         this.modoTaqui = modoTaqui;
368     }
369
370     public String getComentarioProgramacionTaquicardia() {
371         return comentarioProgramacionTaquicardia;
372     }
373
374     public void setComentarioProgramacionTaquicardia(String comentarioProgramacionTaquicardia) {
375         this.comentarioProgramacionTaquicardia = comentarioProgramacionTaquicardia;
376     }
377 }

```

Capturas 271: Capturas de pantalla del código de la clase *ProgramacionTaquicardia.java*.

Además, definimos los estereotipos que faltan Voltios y vms:

- Fichero *editores.xml*:

```

72 <editor nombre="voltios" url="voltiosEditor.jsp">
73     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
74     <para-estereotipo estereotipo="VOLTIOS"/>
75 </editor>
76
77 <editor nombre="vms" url="vmsEditor.jsp">
78     <formateador clase="org.openxava.cardiologia.formateadores.PesoFormateador" />
79     <para-estereotipo estereotipo="VMS"/>
80 </editor>
81

```

Captura 272: Captura de pantalla de fragmento de código del script *editores.xml*.

- Clase formateadora (sólo se ha de crear una sola vez):

```

1 package org.openxava.cardiologia.formateadores;
2
3 import java.math.*;
4
5
6
7
8
9 public class PesoFormateador implements IFormatter {
10
11     public String format(HttpServletRequest request, Object object) throws Exception {
12         if (Is.empty(object)) return "";
13         return getFormat().format(object);
14     }
15
16     public Object parse(HttpServletRequest request, String string) throws Exception {
17         if (Is.emptyString(string)) return null;
18         string = Strings.change(string, " ", ""); // In order to work with Polish
19         return new BigDecimal(getFormat().parse(string).toString()).setScale(2);
20     }
21
22     private NumberFormat getFormat() {
23         NumberFormat f = DecimalFormat.getNumberInstance(Locales.getCurrent());
24         f.setMinimumFractionDigits(2);
25         f.setMaximumFractionDigits(2);
26         return f;
27     }
28
29 }
30

```

Captura 273: Captura de pantalla de la clase *PesoFormateador*.

- Scripts de los editores:

```

1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7 symbol = "V"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 symbol = "?";
11 }
12 %>
13
14 <%=symbol%>

```

Captura 274: Captura de pantalla del script *voltiosEditor.jsp*.

```

1 <%@page import="java.util.Currency"%>
2 <%@page import="java.util.Locale"%>
3 <%@ include file="textEditor.jsp"%>
4
5 <% String symbol = null;
6 try {
7 symbol = "V*ms"; //Currency.getInstance(Locale.getDefault()).getSymbol();
8 }
9 catch (Exception ex) { // Because Locale.getDefault() may not contain the country
10 symbol = "?";
11 }
12 %>
13
14 <%=symbol%>

```

Captura 275: Captura de pantalla del script *vmsEditor.jsp*.

Por último, añadimos las etiquetas para la internacionalización de los componentes del apartado Visitas:

```

283 #Componentes de Visita
284 EXTRAORDINARIA=Extraordinaria
285 HOSPITALIZADO=Hospitalizado
286 PREALTA=Prealta
287 REGULAR=Regular
288
289 #Componentes de Diagnóstico
290 APAGADO=Apagado
291 MONITOR=Monitor
292 MONITORYTERAPIA=Monitor y Terapia
293 ECAR79=ECAR 79
294
295 #Componentes de Evento Médico
296 FAPAROXISTICA=FA Paroxística
297 FAPERMANENTE=FA Permanente
298 FAPERSISTENTE=FA Persistente
299 APROPIADAEFICAZ=Apropiada Eficaz
300 APROPIADAINEFICAZ=Apropiada Ineficaz
301 INAPROPIADAEFICAZ=Inapropiada Eficaz
302 INAPROPIADAINEFICAZ=Inapropiada Ineficaz

```

```

360 #Componentes de Diagnóstico
361 Diagnostico=Diagnóstico
362 situacionBateria=Situación batería
363 voltajeBateria=Voltaje batería
364 indicadorBateria=Indicador batería
365 amplitudMvDv=Amplitud VD
366 umbralVd=Umbral VD
367 estimulacionVd=Estimulación VD
368 estadoElectrodoVd=Estado Electrodo VD
369 tratamientoYRecomendaciones=Tratamiento y Recomendaciones
370
371 #Componentes de Evento Médico
372 EventoMedico=Evento médico
373 fibrilacionAuricular=Fibrilación auricular
374 taquicardiaVentricular=Taquicardia ventricular
375 otrasArritmias=Otras arritmias
376 comentarioDescargas=Comentario descargas
377 atp=ATP
378 comentarioAtp=Comentario ATP
379 reinicioContadores=Reinicio contadores
380 episodiosFv=Episodios FV
381 episodiosTv=Episodios TV
382 episodiosTv1=Episodios TV-1
383 episodiosTnvs=Episodios TNVS
384 cambiosRta=Cambios RTA

```

```

404 #Componentes de Programación Taquicardia
405 ProgramacionTaquicardia=Programación taquicardia
406 zonaTv1=Zona TV-1
407 zonaTv=Zona TV
408 zonaFv=Zona FV
409 atpTv1=ATP TV-1
410 atpTv=ATP TV
411 numRafagasAtp1Tv1=Núm. ráfagas ATP1 TV-1
412 numRafagasAtp1Tv=Núm. ráfagas ATP1 TV
413 atp2Tv1=ATP2 TV-1
414 atp2Tv=ATP2 TV
415 numRafagasAtp2Tv1=Núm. ráfagas ATP2 TV-1
416 numRafagasAtp2Tv=Núm. ráfagas ATP2 TV
417 descargaTv1=Descarga TV-1
418 descargaTv=Descarga TV
419 descargaFv=Descarga FV
420 descarga2Tv1=Descarga 2 TV-1
421 descarga2Tv=Descarga 2 TV
422 descarga2Fv=Descarga 2 FV
423 maxEnergiaTv1=Máx. energía TV-1
424 maxEnergiaTv=Máx. energía TV
425 maxEnergiaFv=Máx. energía FV
426 prorrogaTv1=Prórroga TV-1
427 prorrogaTv=Prórroga TV
428 prorrogaFv=Prórroga FV
429 comentarioProgramacionTaquicardia=Comentario programación taquicardia

```

Capturas 276: Capturas de pantalla de fragmento del script *EtiquetasCardiologias_es.properties*.

5.10 Roles de usuario y acceso a la aplicación.

Para personalizar la página de acceso a la aplicación web para el HUGCDN, modificamos un fichero para conseguirlo. Para ello, añadimos en el fichero MensajesCardiologia_es.properties:

```
1 # Mensajes para la aplicaciin Cardiologia
2 welcome_point1=Introduce tu usuario y contraseña.
3 welcome_point2=Accede a los datos de paciente, pruebas médicas cardiológicas, visitas médicas y demás información.
4 welcome_point3=Genera informes médicos de manera rápida y sencilla.
5 los_medicos_deben_ser_distintos=Los médicos no pueden ser iguales
6 fecha_explante_deben_ser_mayor_que_fecha_implante=La fecha de explante debe ser mayor que la fecha de implante
7 fecha_resolucion_deben_ser_mayor_que_fecha_comienzo=La fecha de resolución debe ser mayor que la fecha de comienzo
```

Captura 277: Captura de pantalla del contenido del archivo *MensajesCardiologia_es.properties*.

Con esto conseguimos lo siguiente:

Iniciar sesión

Bienvenido a KHArdiLOgía

- Introduce tu usuario y contraseña.
- Accede a los datos de paciente, pruebas médicas cardiológicas, visitas médicas y demás información.
- Genera informes médicos de manera rápida y sencilla.

Para empezar con KHArdiLOgía inicia sesión con el usuario: **admin**, contraseña: **admin**

Captura 278: Captura de pantalla de la página de acceso a la aplicación de cardiología del HUGCDN.

Las especificaciones de los requisitos piden que el usuario acceda a la aplicación autenticado y que éste tenga rol de usuario administrativo, facultativo o informático.

En este proyecto fin de carrera se configuró y desarrolló la aplicación para el usuario facultativo que puede editar y leer, ya que el servicio de informática del HUGCDN configura con otras herramientas a las que no tiene acceso el proyectante de este PFC el usuario administración que tiene sólo permisos de lectura.

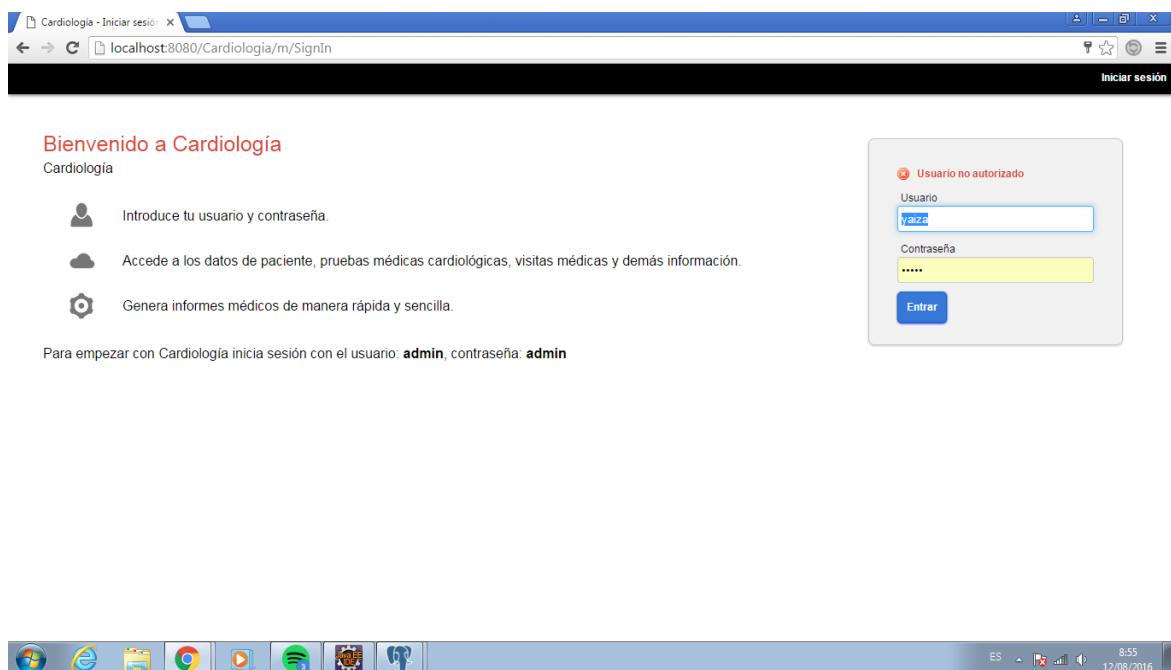
Para ello, añadir nuevos usuarios en la aplicación fue bastante sencillo, se modificó el fichero llamado *naviox-users.properties*, ubicado en la carpeta *properties* del proyecto.

En cada línea incluimos el nombre_usuario=contraseña:

```
1 # Cardiolgia - Open Java Cardiolgia - Cardiolgia - InformePrue.java form user=password
2 # If you want full management of users, roles and modules stored in database,
3 # get XavaPro from here: http://www.openjava.org/xavapro
4
5 admin=admin
6 yaiza=yaiza
7 oscar=oscar
8
```

Captura 279: Captura de pantalla del contenido del archivo en *Eclipse*.

Vemos que al introducir un usuario y/o contraseña erróneo, la aplicación no permite su acceso:



Captura 280: Captura de pantalla con usuario y/o contraseña erróneo.

Como se ha venido comentando a lo largo del Bloque 5, para que el usuario facultativo no pueda borrar información, debemos quitar la acción Borrar en modo lista (en la colección) y modo detalle (en el formulario).

Para que desaparezca la opción borrar en ambos modos, anotamos en cada colección:

@RemoveAction("")

@RemoveSelectedAction("")

@OneToMany (mappedBy="alergiasNuevasLocal", cascade=CascadeType.ALL)

private Collection<AlergiaNueva> alergiasNuevas = new ArrayList<AlergiaNueva>();

De esta manera al no especificar ninguna acción dentro de las "", se desactiva la opción de borrado (la primera desactiva en modo detalle, la segunda desactiva en modo lista).

5.11 Auditoría de los datos.

Como se pudo revisar en la documentación de Openxava, se pudo modificar el botón de guardar en las colecciones, para que además de salvar el elemento, capture datos relevantes de quién ha grabado, cuándo y el qué.

Veamos el ejemplo de cómo se guarda estos datos para *alergiaNueva.java*.

Primer paso. Añadir en *controladores.xml*:

```
48< <controlador nombre="AlergiaNueva">
49    <hereda-de controlador="Typical" />
50 </controlador>
51
52< <controlador nombre="Lopd">
53     <accion nombre="Guardar" icono="Library-plus"
54         clase="org.openxava.cardiologia.actions.GuardaAlergia" >
55     </accion>
56 </controlador>
57
```

Captura 281: Captura de pantalla del script *controladores.xml*.

Segundo paso. Añadir en la colección, la anotación de la línea 373 para la acción nueva de grabar:

```
368 //se deshabilita la opción borrar desde el formulario de alergia
369 @RemoveAction("")
370 //se deshabilita la opción borrar desde la lista de alergias
371 @RemoveSelectedAction("")
372 //ejemplo de cómo auditar los datos en la BB.DD.
373 @SaveAction("lopd.Guardar")
374 //si se borra paciente, se
375 //borrarán sus alergias
376 @OneToMany (mappedBy="alergiasNuevasLocal", cascade=CascadeType.ALL)
377 //para poder visualizar
378 //la lista de alergias
379 private Collection<AlergiaNueva> alergiaNueva = new ArrayList<AlergiaNueva>();
```

Captura 281: Captura de pantalla de fragmento de código de *Paciente.java*.

Tercer paso. Creamos la acción que graba en la tabla *GuargaAlergia.java*:

```

enlaces.txt x prueba.jsp x prueba.java x cabecera.jsp x paciente.java x generador.java x vista.java x guardaAlergia.java x
1 package org.openxava.cardiologia.actions;
2
3 import java.sql.*;
4 import java.util.*;
5
6 import javax.naming.*;
7
8 import javax.sql.*;
9
10 import org.openxava.actions.*;
11 import org.openxava.util.*;
12
13
14 public class GuardaAlergia extends SaveElementInCollectionAction{
15
16     public void execute() throws Exception {
17         super.execute();
18         GrabarTransaccion();
19     }
20
21
22     private void GrabarTransaccion() throws Exception {
23         try{
24             Context ctx = new InitialContext();
25             String recurso="java:comp/env/jdbc/CardiologiaDS";
26             DataSource dataSource = (DataSource) ctx.lookup(recurso);
27
28             Connection conn = dataSource.getConnection();
29
30             int paciente = getView().getValueInt("numeroNhc");
31             ...
32
33             //fecha
34             Calendar fecha = new GregorianCalendar();
35             //Obtenemos el valor del año, mes, día,
36             //hora, minuto y segundo del sistema
37             //usando el método get y el parámetro correspondiente
38             int año = fecha.get(Calendar.YEAR);
39             int mes = fecha.get(Calendar.MONTH);
40             int dia = fecha.get(Calendar.DAY_OF_MONTH);
41             int hora = fecha.get(Calendar.HOUR_OF_DAY);
42             int minuto = fecha.get(Calendar.MINUTE);
43             int segundo = fecha.get(Calendar.SECOND);
44             String fechaActual = ("Fecha grabación: "
45                 + dia + "/" + (mes+1) + "/" + año) + "," + " Hora grabación: "
46                 + hora + ":" + minuto + ":" + segundo;
47             //fecha
48             String accion = getClass().getSimpleName();
49             //String accion = getView().getModelName();
50             String usuario = Users.getCurrent(); // El id del usuario actualmente identificado (1)
51             String id = paciente + usuario + año + (mes+1) + dia + hora + minuto + segundo;
52
53             Statement stmtDAE= conn.createStatement();
54             String consulta="INSERT INTO lopd( id, usuario, paciente, accion, fechaactual) VALUES
55             ('"+id+"', '"+usuario+"', '"+paciente+"', '"+accion+"', '"+fechaActual+'')";
56             int res= stmtDAE.executeUpdate(consulta);
57
58             stmtDAE.close();
59             conn.close();
60
61         }
62     }

```

Captura 282: Capturas de pantalla de la acción *GuardaAlergia.java*.

Problemas solventados:

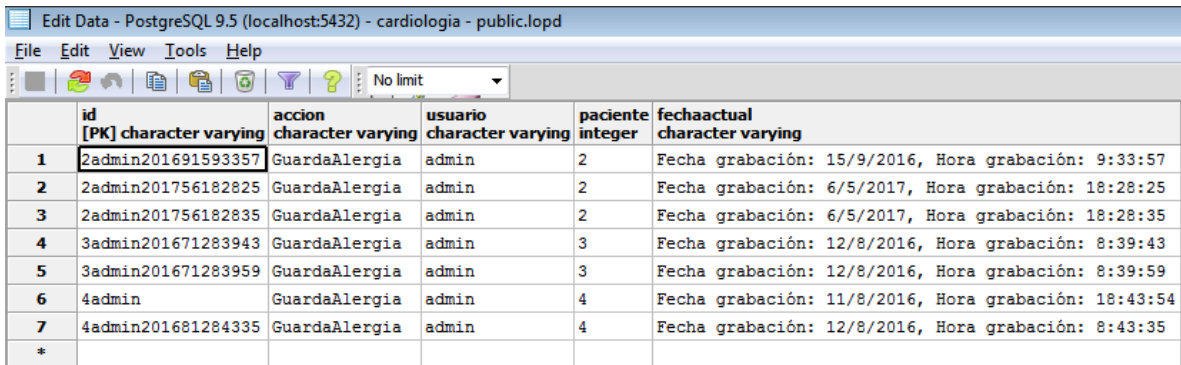
- *superexecute* en clase *GuardaAlergia* sino no funciona
- *getClass().getSimpleName()* para anotar qué clase se está creando o modificando ya que *String accion = getView().getModelName()*; coge solo *Paciente*, y no es tan

específico como señalar la clase Alergia.

Por lo tanto, cuando se realiza una acción sobre un elemento, ésta se puede almacenar y consultar en la BB.DD de cardiología. El identificador único para cada registro es el siguiente:

NHC+USUARIO+FECHA+HORA

Vemos en la siguiente tabla lopd, que se almacenan estos datos:



	id [PK] character varying	acción character varying	usuario character varying	paciente integer	fechaactual character varying
1	2admin201691593357	GuardaAlergia	admin	2	Fecha grabación: 15/9/2016, Hora grabación: 9:33:57
2	2admin201756182825	GuardaAlergia	admin	2	Fecha grabación: 6/5/2017, Hora grabación: 18:28:25
3	2admin201756182835	GuardaAlergia	admin	2	Fecha grabación: 6/5/2017, Hora grabación: 18:28:35
4	3admin201671283943	GuardaAlergia	admin	3	Fecha grabación: 12/8/2016, Hora grabación: 8:39:43
5	3admin201671283959	GuardaAlergia	admin	3	Fecha grabación: 12/8/2016, Hora grabación: 8:39:59
6	4admin	GuardaAlergia	admin	4	Fecha grabación: 11/8/2016, Hora grabación: 18:43:54
7	4admin201681284335	GuardaAlergia	admin	4	Fecha grabación: 12/8/2016, Hora grabación: 8:43:35
*					

Captura 283: de pantalla de registros de auditoría de los datos.

Como conclusión, diremos que cualquier acción, tanto de escritura, lectura, etc se puede auditar y podemos guardarla como mejor convenga.

5.12 Mayor usabilidad para la aplicación de cardiología del HUGCDN.

Además de la configuración de las etiquetas para la internacionalización de la aplicación, los estereotipos para dar formato a los datos, la carga de plantillas o el uso de la tecnología AJAX para facilitar al usuario el uso de la aplicación web de cardiología, se modificaron algunos aspectos definidos por defecto, como por ejemplo, que al grabar un objeto no se cierre el formulario y pueda continuar modificándolo, o añadir anotaciones para tener una interfaz más amigable.

5.12.1 No salir del objeto al grabar.

Por defecto *Openxava* sale del objeto paciente al grabar. Para nuestra aplicación lo ideal es que se mantenga en el paciente para que cada vez que se realice una modificación no

tengamos que volver a buscarlo. Para ello debemos modificar una variable, **resetAfter**.

Dentro del proyecto Openxava, En paquete **org.openxava.actions**, en la clase **SaveAction.java**, sólo hay que cambiar el valor por defecto en la variable;

private boolean resetAfter = false; (antes era true).



```
1 package org.openxava.actions;
2
3 import java.util.Map;
11
12 /**
13  * @author Javier Paniza
14  */
15
16 public class SaveAction extends ViewBaseAction {
17
18     private boolean resetAfter = false;
19     private boolean refreshAfter = true;
20
21     public void execute() throws Exception {
22         try {
23             Map values = null;
24             if (getView().isKeyEditable()) {
25                 // Create
26                 if (isResetAfter() || !isRefreshAfter()) {
27                     MapFacade.create(getModelName(), getValuesToSave());
28                     addMessage("entity_created", getModelName());
29                 }
30                 else {
31                     Map keyValues = MapFacade.createReturningKey(getModelName(), getValuesToSave());
32                     addMessage("entity_created", getModelName());
33                     getView().clear();
34                 }
35             }
36         }
37     }
38 }
```

Captura 284: Captura de pantalla de la clase *SaveAction.java*.

5.12.2 Minimizar apartados.

Al tener un formulario muy extenso con varios apartados, puede volverse tedioso para el usuario recorrer un largo *scroll* para llegar a la información que te interesa.

En la clase *Visita.java*, se muestran las tres secciones Evento médico, Programación Bradicardia, y Programación Taquicardia de su clase incrustada *Diagnostico.java* como apartados minimizados, que permite al usuario extender el formulario si le interesa rellenar esta información. Además, recuerda su configuración al iniciar sesión de nuevo, desplegando las que anteriormente seleccionó. Esto lo conseguimos gracias a la anotación **@Collapse** en las líneas 193, 204 y 216:

```

191 //CLASES INCRUSTABLES
192 //Diagnóstico tiene solo un EventoMedico
193 @Collapsed //Anotación para minimizar el apartado
194 @OneToOne //relación 1 a 1 entre Diagnostico y eventoMedico
195 @JoinColumn( //se define la clave foránea de la tabla diagnostico
196     name = "idevento", //columna que corresponde con el id de evento
197     nullable = true,
198     foreignKey = @ForeignKey(name = "fk_diagnostico_id"))
199     //fk_diagnostico_id es la clave foránea en la tabla diagnostico de la BB.DD.
200 @AsEmbedded //Anotación Openxava para incrustar una clase en otra.
201 private EventoMedico eventoMedico;
202
203 //Diagnóstico tiene solo una programación bradicardia
204 @Collapsed //Anotación para minimizar el apartado
205 @OneToOne //relación 1 a 1 entre Diagnostico y programacionBradicardia
206 @JoinColumn( //se define la clave foránea de la tabla diagnostico
207     name = "idpb", //columna que corresponde con el id de programacionBradicardia
208     nullable = true,
209     foreignKey = @ForeignKey(name = "fk_diagnosticipb_id"))
210     //fk_diagnosticipb_id es la clave foránea en la tabla diagnostico de la BB.DD.
211 @AsEmbedded //Anotación Openxava para incrustar una clase en otra.
212 private ProgramacionBradicardia programacionBradicardia;
213
214 //Diagnóstico tiene solo una programación taquicardia
215 @Collapsed //Anotación para minimizar el apartado
216 @OneToOne //relación 1 a 1 entre Diagnostico y programacionTaquicardia
217 @JoinColumn( //se define la clave foránea de la tabla diagnostico
218     name = "idpt", //columna que corresponde con el id de programacionTaquicardia
219     nullable = true,
220     foreignKey = @ForeignKey(name = "fk_diagnosticopt_id"))
221     //fk_diagnosticopt_id es la clave foránea en la tabla diagnostico de la BB.DD.
222 @AsEmbedded //Anotación Openxava para incrustar una clase en otra.
223 private ProgramacionTaquicardia programacionTaquicardia;

```

Captura 285: Captura de pantalla de fragmento de código de la clase *Diagnostico.java*

5.13 Búsqueda de pacientes y gráficas de información relevante.

Al usuario, tras autenticarse en la página de acceso de la aplicación web de cardiología, se le muestra la página de listado de pacientes, donde se puede acceder a cualquier paciente de la BB.DD. de Drago y aplicar criterios de búsquedas de paciente:

KHArдиоLOGía - Paciente ☆

Nuevo Generar PDF Generar Excel Mis informes

Detalle Lista Ambos

	Número de Historia	Nombre	Contacto	Otro contacto	Dirección	Peso	Estatura	BSA	IMC	Sobrepeso
	=	empieza por	=	=	empieza por	=	=	=	=	empieza por
	4	María Jesús Santana Santar	928000000	0	calle la nube	94,00	1,59	0,04	37,18	
	5	Prueba	928000000	0	calle fulantito	89,00	1,90	0,05	24,65	
	1	Mónica Santana	928000000	0	calle pascal	100,00	1,70	0,05	34,6	
	15	prueba	65	655		20,00	2,00	0,01	5	
	2	Yaiza Santana	928000000	887799	LEO	62,00	1,71	0,03	21,2	
	3	Oscar Vega	928000000	620262427	calle Piscis	92,00	2,00	0,05	23	
	Σ		Σ	Σ		Σ	Σ	Σ	Σ	

1 10 filias por página Hay 6 registros en la lista (Ocultarlos)

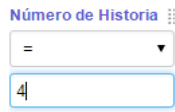
Borrar filias seleccionadas

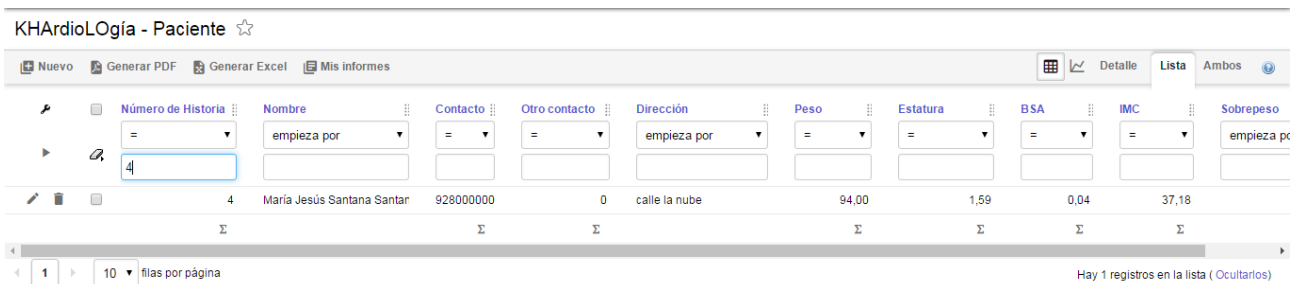
Captura 286: Captura de pantalla de la página de inicio de la aplicación web de cardiología.

Como podemos ver, *Openxava* ofrece un menú principal para la página de inicio “Búsqueda de pacientes” que consta de 9 botones:

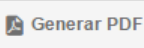
5.13.1 Botón Nuevo . Este botón queda desactivado en la versión de producción al realizar las modificaciones explicadas en el apartado de acceso a los datos de la BB.DD de Drago. Este botón no debe aparecer ya que no se puede crear pacientes nuevos.

5.13.2 Botón Generar PDF . Si pulsamos este botón, se imprimirá en un fichero extensión PDF la lista de pacientes que se esté visualizando en esta página web de inicio. Podemos manipular esta lista mediante los filtros de búsquedas. Por ejemplo, al filtrar en el filtro

de número de historia clínica con el número 4(dándole al enter) , se visualiza la lista de paciente con ese Id:



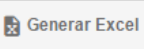
Captura 287: Captura de pantalla de la página de inicio con filtros de búsqueda.

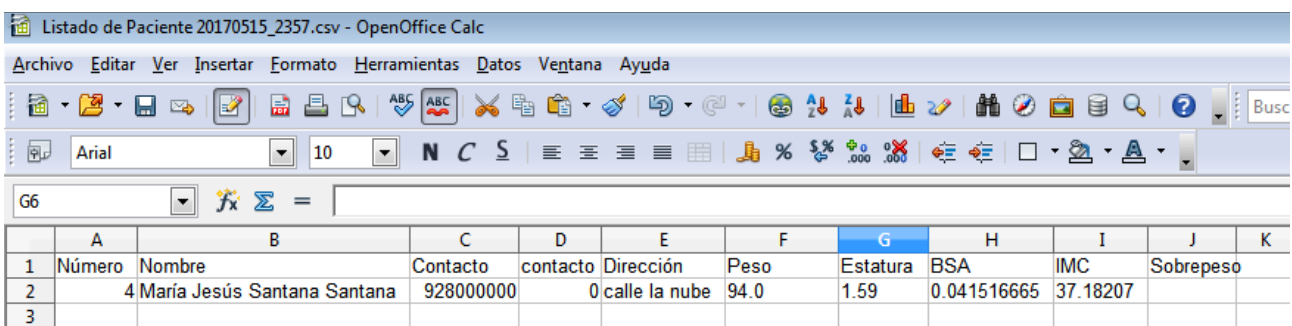
Al pulsar en el botón , se genera el siguiente PDF, permitiendo su descarga:




Captura 288: Captura de pantalla del archivo PDF que contiene la lista de pacientes deseada.


Esta funcionalidad puede ser útil para estudios, estadísticas, etc.

5.13.3 Botón Generar Excel . Si pulsamos este botón, se imprimirá en un fichero extensión csv la lista de pacientes que se esté visualizando en esta página web de inicio. Podemos manipular esta lista mediante los filtros de búsquedas como en el caso anterior. Al pulsar el botón, obtenemos el siguiente fichero:




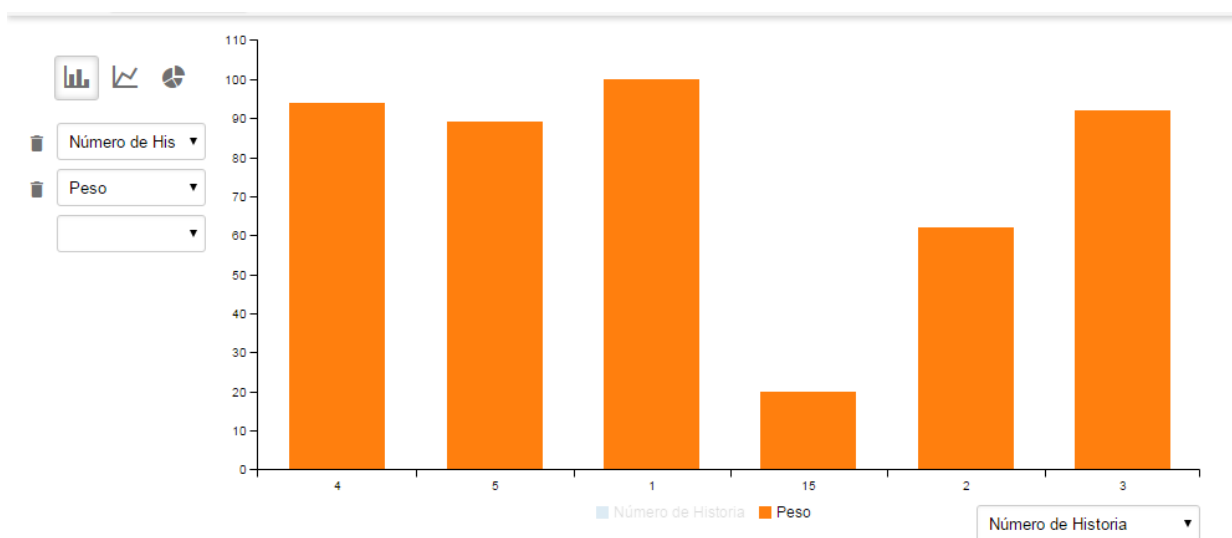
Captura 289: Captura de pantalla del archivo excel que contiene la lista de pacientes deseada.

5.13.4 Botón datos . Este botón es el que está activo por defecto en la página de inicio de Búsqueda de pacientes, el cual permite ver los datos del modo lista que venimos visualizando en este apartado.

5.13.5 Botón gráficas . Al pulsar este botón, sobre la lista de pacientes visualizada, podemos realizar cualquier gráfica de sus parámetros deseados y relacionarlos como queramos. Veamos el ejemplo de la gráfica del peso de los pacientes:

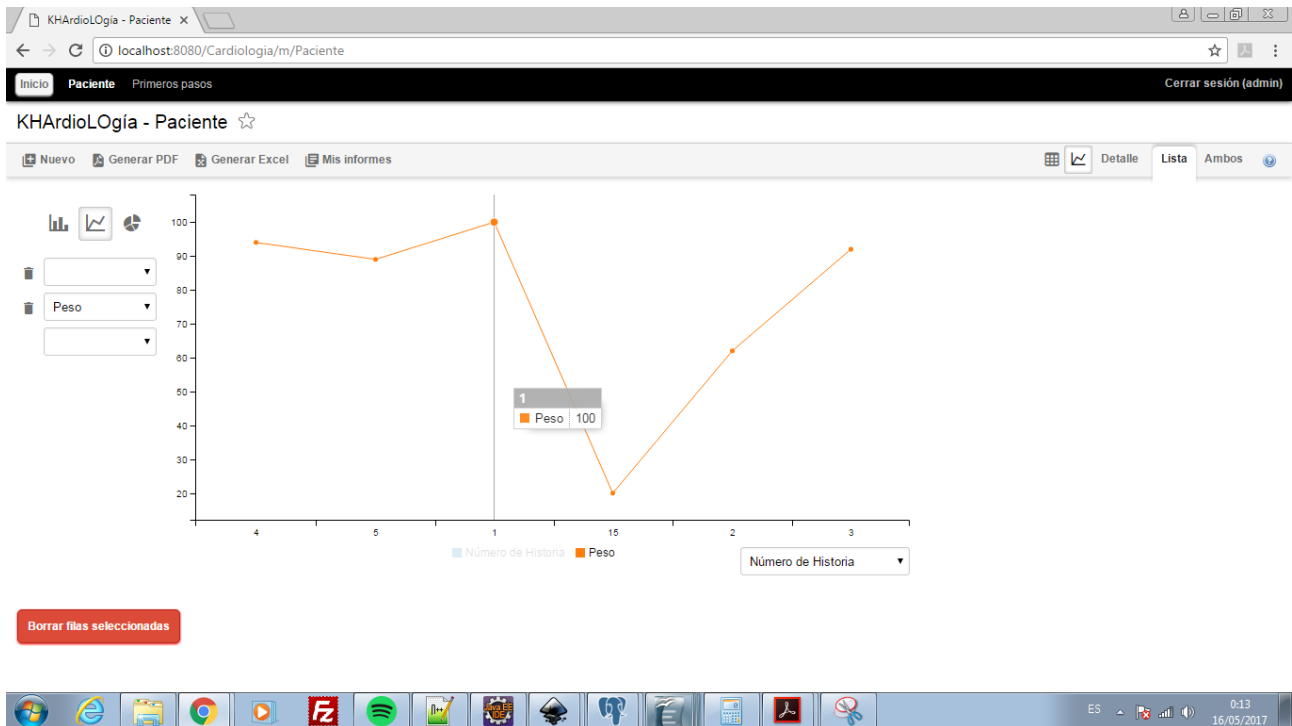
Como podemos ver a continuación, tenemos tres modos de representación.

5.13.5.1 Representación en barras : si la seleccionamos, veremos la gráfica que representa el el peso de paciente mediante barras:



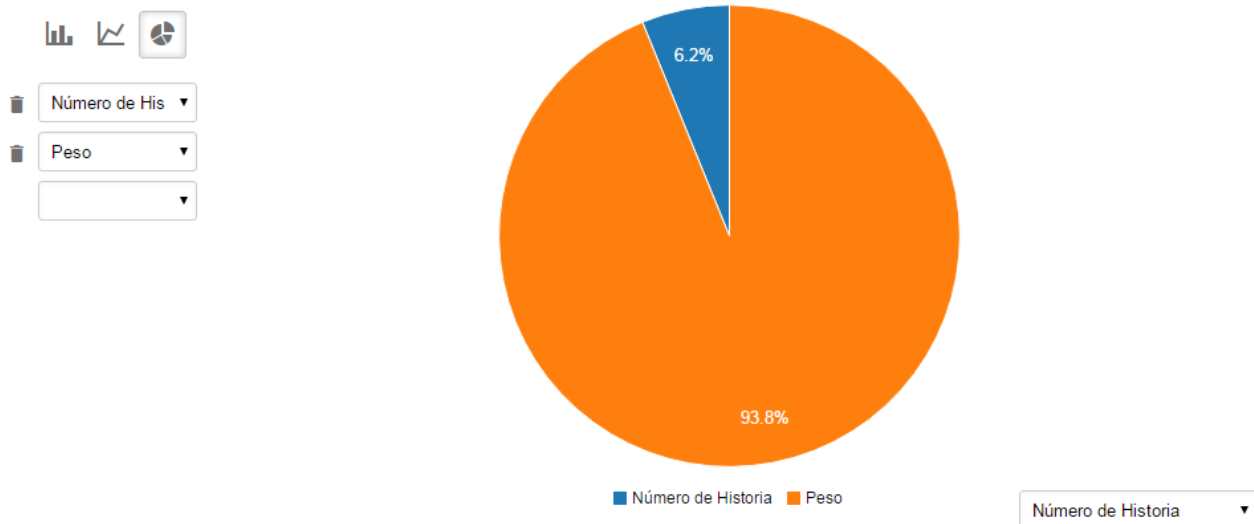
Captura 290: Captura de pantalla de la gráfica de barras de peso de paciente.

5.13.5.2 Representación lineal : si pulsamos en este botón, veremos la gráfica:



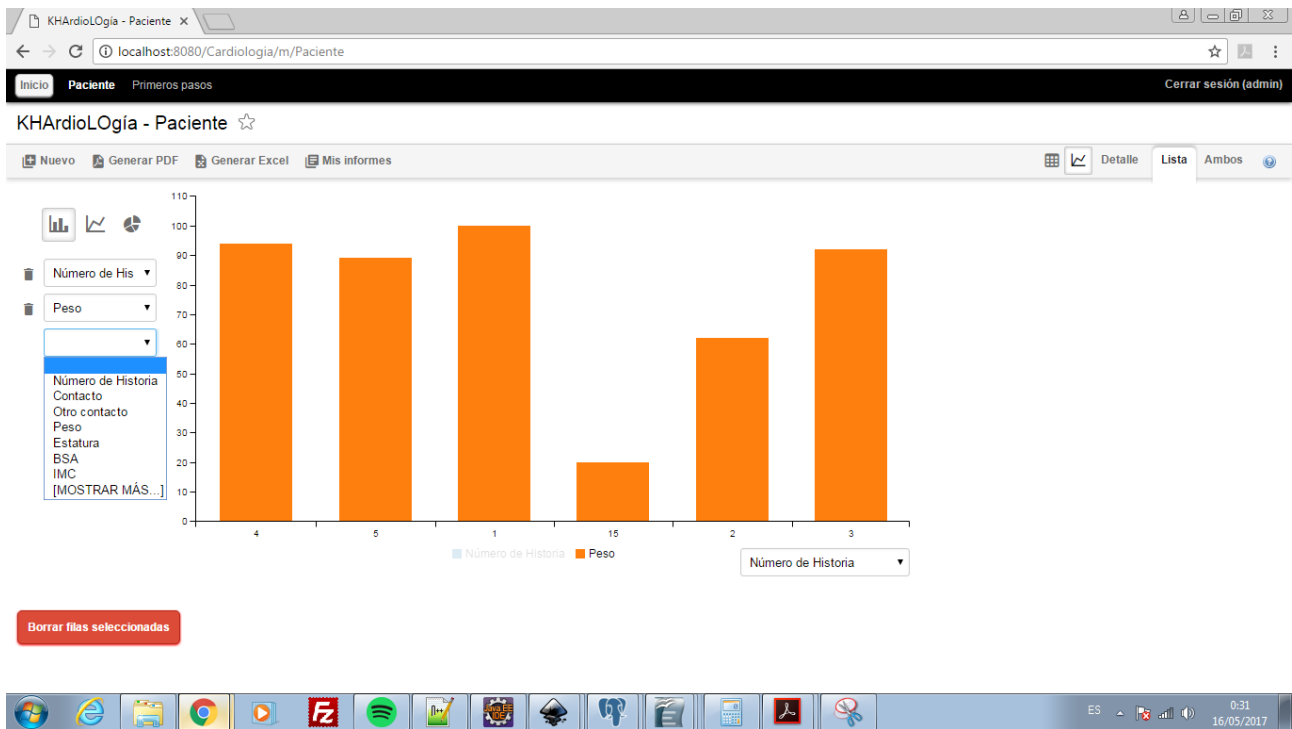
Captura 291: Captura de pantalla de la gráfica lineal de peso de paciente.

5.13.5.3 Representación en porcentajes . Aunque para este ejemplo no es significativo, vemos que seleccionando dos parámetros se realiza una comparativa entre ellos.




Captura 292: Captura de pantalla de la gráfica comparativa.

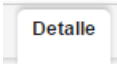
En estas gráficas podemos añadir tantos parámetros de estudio como queramos con el desplegable como vemos en la siguiente captura:




Captura 293: Captura de pantalla de los valores del desplegable añadir parámetro.

También podemos eliminar estos parámetros con el botón , el cual borrará el desplegable de su fila.

Comentar que podemos cambiar el valor del eje de las X, gracias al desplegable que encontramos en su eje

5.13.6 Botón detalle . Este botón permite ver los datos del paciente.

5.13.7 Botón lista . Al pulsar este botón, nos lleva a la página de inicio de búsqueda de pacientes que se visualiza tras autenticarse en la aplicación:

KHArдиоLOGía - Paciente ☆

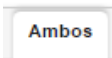
Nuevo Generar PDF Generar Excel Mis informes

Detalle Lista Ambos

Número de Historia	Nombre	Contacto	Otro contacto	Dirección	Peso	Estatura	BSA	IMC	Sobrepeso
4	María Jesús Santana Santar	928000000	0	calle la nube	94,00	1,59	0,04	37,18	
5	Prueba	928000000	0	calle fulantito	89,00	1,90	0,05	24,65	
1	Mónica Santana	928000000	0	calle pascal	100,00	1,70	0,05	34,6	
15	prueba	65	655	calle	20,00	2,00	0,01	5	
2	Yaiza Santana	928000000	887799	LEO	62,00	1,71	0,03	21,2	
3	Oscar Vega	928000000	620262427	calle Piscis	92,00	2,00	0,05	23	
Σ		Σ	Σ		Σ	Σ	Σ	Σ	

1 10 filas por página Hay 6 registros en la lista (Ocultarlos)

Captura 294: Captura de pantalla de la página de inicio de la aplicación web de cardiología.

5.13.8 Botón ambos . Pulsando este botón, se cargará en la misma página el modo lista de pacientes y el modo detalle de paciente, como podemos observar en la siguiente captura de pantalla:

KHArдиоLOGía - Paciente ☆

Nuevo Generar PDF Generar Excel Mis informes

Detalle Lista Ambos

Número de Historia	Nombre	Contacto	Otro contacto	Dirección	Peso	Estatura	BSA	IMC	Sobrepeso
4	María Jesús Santana Santar	928000000	0	calle la nube	94,00	1,59	0,04	37,18	
5	Prueba	928000000	0	calle fulantito	89,00	1,90	0,05	24,65	
1	Mónica Santana	928000000	0	calle pascal	100,00	1,70	0,05	34,6	
15	prueba	65	655	calle	20,00	2,00	0,01	5	
2	Yaiza Santana	928000000	887799	LEO	62,00	1,71	0,03	21,2	
3	Oscar Vega	928000000	620262427	calle Piscis	92,00	2,00	0,05	23	
Σ		Σ	Σ		Σ	Σ	Σ	Σ	

1 10 filas por página Hay 6 registros en la lista (Ocultarlos)

Borrar filas seleccionadas

Nuevo Grabar Borrar Buscar Refrescar

Datos demográficos

Número de Historia: 1 Nombre: Mónica Santana

Fecha de nacimiento: 15/10/1985 00:00 Sexo: Mujer

Contacto: 928000000 Otro contacto: 0

Dirección: calle pascal

Estado nutricional

Peso: 100,00 Kg

Estatura: 1,70 metros

BSA: 0,05 IMC: 34,6

Sobrepeso: Obeso: Tipo I

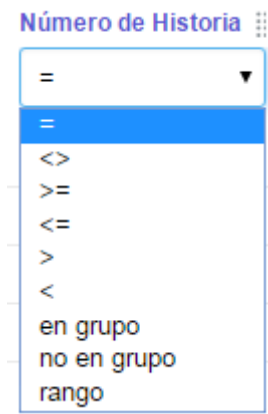
Ficha de paciente [Diagnos médicos](#) [Sistemas](#) [Visitas](#)

Captura 295: Captura de pantalla al pulsar botón Ambos.

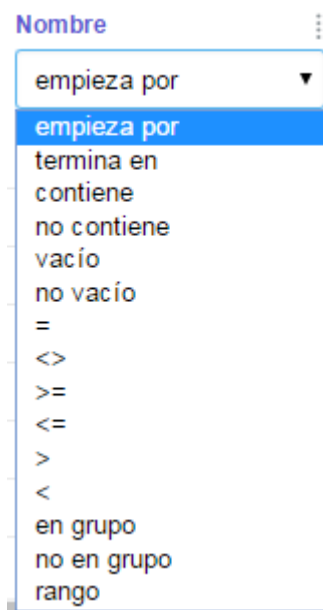
En la parte superior, tenemos la página de inicio donde se visualizan los pacientes de la BB.DD y se puede buscar al paciente deseado, y debajo los detalles del paciente seleccionado. Si queremos ver otro paciente, sólo será necesario *clickar* sobre él, y mediante *Ajax* se cargan los datos del paciente en la parte inferior.

5.13.9 Filtros.

En esta página, encontramos 10 tipos de filtros: *nhc*, nombre, contacto, otro contacto, dirección, peso, estatura, BSA, IMC y sobrepeso. Cada filtro consta de dos partes: un desplegable que ayuda a mejorar las búsquedas y que se define según el tipo de dato que sea, y la otra de una casilla donde se introducen los valores de búsqueda. Por ejemplo, para los datos de tipo numérico y de tipo *string*, el desplegable toma el siguiente valor:



Captura 296: Desplegables para filtros de tipo numérico.



Captura 297: Desplegables para filtros de tipo *string*.

Veamos cómo estos buscadores son muy potentes. Por ejemplo, nos interesa buscar a pacientes que tienen un peso corporal comprendido entre los 50 y los 70 kilos. Para ello seleccionamos la opción "rango" del desplegable tipo numérico como vemos en la siguiente captura:

Número de Historia	Nombre	Contacto	Otro contacto	Dirección	Peso	Estatura	BSA	IMC	Sobrepeso
4	María Jesús Santana Santar	928000000	0	calle la nube	94,00	1,59	0,04	37,18	
5	Prueba	928000000	0	calle fulanito	89,00	1,90	0,05	24,65	
1	Mónica Santana	928000000	0	calle pascal	100,00	1,70	0,05	34,6	
15	prueba	65	655	calle	20,00	2,00	0,01	5	
2	Yaiza Santana	928000000	887799	LEO	62,00	1,71	0,03	21,2	
3	Oscar Vega	928000000	620262427	calle Piscis	92,00	2,00	0,05	23	
Σ					Σ	Σ	Σ	Σ	Σ

Captura 289: Captura de pantalla del filtro peso.

Ahora introducimos los valores en sus correspondientes casillas:

Número de Historia	Nombre	Contacto	Otro contacto	Dirección	Peso	Estatura	BSA	IMC	Sobrepeso
4	María Jesús Santana Santar	928000000	0	calle la nube	94,00	1,59	0,04	37,18	
5	Prueba	928000000	0	calle fulanito	89,00	1,90	0,05	24,65	
1	Mónica Santana	928000000	0	calle pascal	100,00	1,70	0,05	34,6	
15	prueba	65	655	calle	20,00	2,00	0,01	5	
2	Yaiza Santana	928000000	887799	LEO	62,00	1,71	0,03	21,2	
3	Oscar Vega	928000000	620262427	calle Piscis	92,00	2,00	0,05	23	
Σ					Σ	Σ	Σ	Σ	Σ

Captura 290: Captura de pantalla del filtro peso con sus valores.

Y por último, pulsamos *enter* en la casilla con el valor 70. Vemos como obtendremos un registro, ya que es el único que cumple esta condición:

Número de Historia	Nombre	Contacto	Otro contacto	Dirección	Peso	Estatura	BSA	IMC	Sobrepeso
2	Yaiza Santana	928000000	887799	LEO	62,00	1,71	0,03	21,2	
Σ					Σ	Σ	Σ	Σ	Σ

Captura 291: Captura de pantalla del listado paciente filtrado.

Si se desea, también podremos resetear los filtros gracias al botón .

Una vez filtrado como se requiere, para acceder al paciente, podemos hacerlo picando en la fila del paciente, o pulsando sobre el botón .

Cabe destacar, que *Openxava* introduce por defecto, paginado de la entidad Paciente, donde se puede elegir cuántos elementos se mostrarán por página de la lista.



Captura 292: Valores del desplegable paginado.

6. Pruebas realizadas.

6 Test de la aplicación.

Para comprobar que la aplicación se comporta tal y como se requiere, se hicieron las siguientes pruebas. Se expone las pruebas realizadas para cada sección y otras de gran interés, como por ejemplo el acceso por usuario, y la auditoría de los datos.

6.1 Acceso a la aplicación.

Prueba: Acceso a la aplicación	
Descripción	Validaciones de diferentes usuarios de la aplicación.
Resultado esperado	Comprobar la validación del usuario y si el usuario tiene o no acceso a la aplicación.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 40: Prueba de acceso a la aplicación.

6.2 Búsqueda de paciente.

Prueba: Búsqueda de paciente	
Descripción	Se obtiene la lista de pacientes de la BB.DD. de Drago.
Resultado esperado	Al usar los filtros de búsqueda de la lista de pacientes, se accede al paciente deseado.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 41: Prueba de búsqueda de paciente.

6.3 Funcionalidad de la cabecera de la aplicación con información del paciente.

Prueba: Datos demográficos del paciente	
Descripción	Se obtienen los datos demográficos del paciente desde la BB.DD de Drago
Resultado esperado	Comprobar que los datos del paciente son reales.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 42: Prueba de datos demográficos del paciente.

Prueba: Estado nutricional del paciente	
Descripción	Se obtiene el estado nutricional del paciente desde la BB.DD de cardiología.
Resultado esperado	Comprobar que los datos del paciente son reales, y que se recalculan las variables IMC y BSA, al modificar peso y estatura del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 43: Prueba del estado nutricional del paciente.

6.4 Funcionalidad del módulo Ficha de paciente.

6.4.1 Alergias del paciente.

Prueba: Consultar alergias del paciente	
Descripción	Consulta a la BB.DD de cardiología de las alergias del paciente seleccionado.
Resultado esperado	Comprobar que se obtienen las alergias del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 44: Prueba de consulta de alergias.

Prueba: Crear alergia del paciente	
Descripción	Insertar en la BB.DD de cardiología la nueva alergia del paciente seleccionado.
Resultado esperado	Comprobar que se graba la alergia del paciente, y se incluye mediante Ajax en el listado de alergias.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 45: Prueba para crear alergias.

Prueba: Modificar alergia del paciente	
Descripción	Modificación en la BB.DD de cardiología de la alergia seleccionada de la lista.
Resultado esperado	Comprobar que se obtienen los datos de la alergia y que la modificación de ésta se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 46: Prueba de modificación de alergia de paciente.

6.4.2 Antecedentes del paciente.

Prueba: Consultar antecedentes del paciente	
Descripción	Consulta a la BB.DD de cardiología de los antecedentes del paciente seleccionado.
Resultado esperado	Comprobar que se obtienen los antecedentes del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 47: Prueba de consulta de antecedentes.

Prueba: Crear antecedente del paciente	
Descripción	Insertar en la BB.DD de cardiología el nuevo antecedente del paciente seleccionado.
Resultado esperado	Comprobar que se graba el antecedente del paciente, y se incluye mediante Ajax en el listado de antecedentes.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 48: Prueba para crear antecedentes.

Prueba: Modificar antecedente del paciente	
Descripción	Modificación en la BB.DD de cardiología del antecedente seleccionado de la lista.
Resultado esperado	Comprobar que se obtienen los datos del antecedente y que la modificación de éste se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 49: Prueba de modificación de antecedente.

6.4.3 Indicaciones del paciente.

Prueba: Consultar indicaciones del paciente	
Descripción	Consulta a la BB.DD de cardiología de las indicaciones del paciente seleccionado.
Resultado esperado	Comprobar que se obtienen las indicaciones del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 50: Prueba de consulta de indicaciones.

Prueba: Crear indicaciones del paciente	
Descripción	Insertar en la BB.DD de cardiología la nueva indicación del paciente seleccionado.
Resultado esperado	Comprobar que se graba la indicación del paciente, y se incluye mediante Ajax en el listado de indicaciones.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 51: Prueba para crear indicaciones.

Prueba: Modificar indicaciones del paciente	
Descripción	Modificación en la BB.DD de cardiología de la indicación seleccionada de la lista.
Resultado esperado	Comprobar que se obtienen los datos de la indicación y que la modificación de ésta se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 52: Prueba para modificar indicaciones.

6.4.4 Procedimientos del paciente.

Prueba: Consultar procedimientos del paciente	
Descripción	Consulta a la BB.DD de cardiología de los procedimientos del paciente seleccionado.
Resultado esperado	Comprobar que se obtienen los procedimientos del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 53: Prueba de consulta de procedimientos.

Prueba: Crear procedimientos del paciente	
Descripción	Insertar en la BB.DD de cardiología el nuevo procedimiento del paciente seleccionado.
Resultado esperado	Comprobar que se graba el procedimiento del paciente, y se incluye mediante Ajax en el listado de procedimientos.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 54: Prueba para crear procedimientos.

Prueba: Modificar procedimientos del paciente	
Descripción	Modificación en la BB.DD de cardiología del procedimiento seleccionado de la lista.
Resultado esperado	Comprobar que se obtienen los datos del procedimiento y que la modificación de éste se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 55: Prueba para modificar procedimientos.

6.4.5 Ingresos del paciente.

Prueba: Consultar ingresos del paciente	
Descripción	Consulta a la BB.DD de cardiología de los ingresos del paciente seleccionado.
Resultado esperado	Comprobar que se obtienen los ingresos del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 56: Prueba de consulta de ingresos.

Prueba: Crear ingresos del paciente	
Descripción	Insertar en la BB.DD de cardiología el nuevo ingreso del paciente seleccionado.
Resultado esperado	Comprobar que se graba el ingreso del paciente, y se incluye mediante Ajax en el listado de ingresos.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 57: Prueba para crear ingresos.

Prueba: Modificar ingresos del paciente	
Descripción	Modificación en la BB.DD de cardiología del ingreso seleccionado de la lista.
Resultado esperado	Comprobar que se obtienen los datos del ingreso y que la modificación de éste se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 58: Prueba para modificar ingresos.

6.4.6 Outcomes del paciente.

Prueba: Consultar Outcomes del paciente	
Descripción	Consulta a la BB.DD de cardiología de la información <i>Outcomes</i> del paciente seleccionado.
Resultado esperado	Comprobar que se obtiene la información <i>Outcomes</i> del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 59: Prueba de consulta de *outcomes*.

Prueba: Grabar Outcomes del paciente	
Descripción	Grabar en la BB.DD de cardiología la información <i>Outcomes</i> del paciente seleccionado.
Resultado esperado	Comprobar que se graba la información <i>Outcomes</i> del paciente, y se incluye mediante Ajax.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 60: Prueba para grabar *outcomes*.

Prueba: Modificar Outcomes del paciente	
Descripción	Modificación en la BB.DD de cardiología de la información <i>Outcomes</i> del paciente.
Resultado esperado	Comprobar que se obtiene la información <i>Outcomes</i> y que la modificación de éste se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 61: Prueba para modificar *outcomes*.

6.4.7 Comentario del paciente.

Prueba: Consultar Comentario del paciente	
Descripción	Consulta a la BB.DD de cardiología del comentario del paciente seleccionado.
Resultado esperado	Comprobar que se obtiene el comentario del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 62: Prueba de consulta de comentario.

Prueba: Grabar Comentario del paciente	
Descripción	Grabar en la BB.DD de cardiología el comentario del paciente seleccionado.
Resultado esperado	Comprobar que se graba el comentario del paciente, y se incluye mediante Ajax.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 63: Prueba para grabar comentario.

Prueba: Modificar Comentario del paciente	
Descripción	Modificación en la BB.DD de cardiología del comentario <i>del paciente</i> del paciente.
Resultado esperado	Comprobar que se obtiene el comentario del paciente y que la modificación de éste se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 64: Prueba para modificar comentario.

6.5 Funcionalidad del módulo Pruebas médicas.

Como las pruebas médicas deben comportarse igual, se incluyen las pruebas genéricas para los ocho tipos de pruebas diagnósticas que puede realizarse el paciente.

Prueba: Consultar pruebas médicas del paciente	
Descripción	Consulta a la BB.DD de cardiología de las pruebas médicas del paciente seleccionado.
Resultado esperado	Comprobar que se obtienen las pruebas médicas del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 65: Prueba de consulta de pruebas médicas.

Prueba: Crear pruebas médicas del paciente	
Descripción	Insertar en la BB.DD de cardiología la nueva prueba médica del paciente seleccionado.
Resultado esperado	Comprobar que se graba la prueba médica del paciente, y se incluye mediante Ajax en el listado de pruebas médicas.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 66: Prueba para crear pruebas médicas.

Prueba: Modificar pruebas médicas del paciente	
Descripción	Modificación en la BB.DD de cardiología de la prueba médica seleccionada de la lista.
Resultado esperado	Comprobar que se obtienen los datos de la prueba médica y que la modificación de ésta se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 67: Prueba para modificar una prueba médica.

Prueba: Generación automática de informes de pruebas médicas del paciente	
Descripción	Generación automática de informe médico de la prueba médica seleccionada a través de un botón.
Resultado esperado	Comprobar que se obtienen los datos de la prueba médica y que éstos se vuelcan correctamente en el informe.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 68: Prueba para generar informe de manera automática.

6.6 Funcionalidad del módulo Sistemas.

Como los sistemas deben comportarse igual, se incluyen las pruebas genéricas para los dos tipos de sistemas que puede tener implantado el paciente, y las pruebas para la visualización de los problemas activos del paciente.

Prueba: Consultar sistemas del paciente	
Descripción	Consulta a la BB.DD de cardiología de los sistemas del paciente seleccionado.
Resultado esperado	Comprobar que se obtienen los sistemas del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 69: Prueba de consulta de los sistemas de paciente.

Prueba: Crear sistemas del paciente	
Descripción	Insertar en la BB.DD de cardiología el nuevo sistema implantado en el paciente seleccionado.
Resultado esperado	Comprobar que se graba el sistema del paciente, y se incluye mediante Ajax en el listado de sistemas.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 70: Prueba para crear un sistema implantado.

Prueba: Modificar sistemas del paciente	
Descripción	Modificación en la BB.DD de cardiología del sistema seleccionado de la lista.
Resultado esperado	Comprobar que se obtienen los datos del sistema y que la modificación de éste se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 71: Prueba para modificar un sistema implantado.

Prueba: Consultar problemas activos del paciente	
Descripción	Consulta a la BB.DD de cardiología de los problemas activos de los sistemas del paciente seleccionado.
Resultado esperado	Comprobar que se obtienen los problemas activos del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 72: Prueba para consultar problemas activos.

6.7 Funcionalidad del módulo Visitas.

Prueba: Consultar visitas del paciente	
Descripción	Consulta a la BB.DD de cardiología de las visitas del paciente seleccionado.
Resultado esperado	Comprobar que se obtienen las visitas del paciente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 73: Prueba de consulta de las visitas del paciente.

Prueba: Crear visitas del paciente	
Descripción	Insertar en la BB.DD de cardiología la nueva visita del paciente seleccionado.
Resultado esperado	Comprobar que se graba la visita del paciente, y se incluye mediante Ajax en el listado de visitas.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 74: Prueba para crear una visita del paciente.

Prueba: Modificar visitas del paciente	
Descripción	Modificación en la BB.DD de cardiología de la visita seleccionada de la lista.
Resultado esperado	Comprobar que se obtienen los datos de la visita y que la modificación de ésta se realiza correctamente.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 75: Prueba para modificar una visita del paciente.

6.8 Auditoría de los datos.

Prueba: Auditoría de los datos	
Descripción	Auditar los datos de la aplicación insertando en la BB.DD de cardiología esta información.
Resultado esperado	Comprobar que se guarda esta auditoría.
Resultado obtenido	Se cumple el resultado esperado.

Tabla 76: Prueba para auditar los datos de las BB.DD.

7. Conclusiones.

7. Conclusiones.

7.1 Introducción.

El objetivo principal de este Proyecto Fin de Carrera fue desarrollar un paquete software para simplificar el trabajo a los especialistas en cardiología, creando una aplicación de fácil uso que unifica todos los servicios que ofrecen ahora las aplicaciones cardiológicas del Hospital Universitario de Gran Canaria Dr. Negrín, *Cardiostim* y *Cardioweb*. Para ello además se añadieron nuevas funcionalidades y tecnologías en su desarrollo, ayudando así a la toma de datos y almacenamiento del paciente y de los parámetros de generadores y electrodos implantados, facilitando el seguimiento de las visitas de los pacientes, y mejorando las prestaciones actuales para el registro y acceso de/a las pruebas diagnósticas así como la generación de informes de éstos últimos de forma automática.

7.2 Conclusiones.

Para lograr los objetivos de este PFC, se desarrolló una aplicación web capaz de guardar la información de un generador o de un electrodo en el momento en el que se implanta, para así, poder revisar dicha información con posterioridad, si fuese necesario. Además, una vez almacenada la información del implante, la aplicación permite llevar un seguimiento del generador o del electrodo cada vez que el paciente asista a la consulta médica, recogiendo una serie de datos que sirven como histórico de la información relevante recopilada por los dispositivos que lleva implantado el paciente, y permite almacenar otra información de interés del paciente.

Así mismo, tras cada realización de prueba diagnóstica al paciente, se puede entregar un informe a éste con los parámetros más relevantes de estas pruebas y una serie de recomendaciones y anotaciones adjuntas al informe.

Por último, también se implementó el acceso a la aplicación a través de usuario, y la auditoría de los datos para salvaguardar la integridad de la información implicada.

7.3 Líneas futuras.

Como mejoras a esta nueva aplicación para el servicio de cardiología, se propone realizar

el diseño *responsive* de la misma, para adaptarla a los dispositivos móviles como *tablets* o *smartphones*.

Además, suponiendo que podamos acceder a la BB. DD. que guarda la agenda de paciente de visitas programadas, se podría implementar con *Openxava* el envío de *sms* o correo electrónico como recordatorio de citas, pruebas médicas.

8. Presupuesto.

8. Presupuesto

8.1 Introducción

Hasta hace unos años para la realización de los presupuestos los honorarios de los trabajos estaba basado en el documento “Baremos de Honorarios Orientativos para Trabajos Profesionales” publicado por el Colegio Oficial de Ingenieros de Telecomunicación (COIT) y la Asociación Española de Ingenieros de Telecomunicación (AEIT). Con la entrada en vigor de la llamada **Ley Ómnibus** [105], el Ministerio de Economía y Hacienda, siguiendo directivas europeas modificó las normativas de los Colegios Profesionales, eliminando los baremos orientativos de honorarios que tradicionalmente se publicaban.

El presupuesto del presente Trabajo Fin de Título se ha dividido en los siguientes apartados:

- Coste de los Recursos Humanos.
- Coste de los Recursos *Hardware*.
- Coste de los Recursos *Software*.
- Coste de la Redacción.
- Coste Total.

8.2 Coste de los Recursos Humanos.

El coste de recursos humanos está asociado al tiempo empleado por un ingeniero en la realización del trabajo. Para el presente trabajo, se estima que se ha trabajado durante 4 meses, 4 horas diarias, en días laborables.

De acuerdo a los Baremos de Honorarios Orientativos para Trabajos Profesionales del Colegio Oficial de Ingenieros de Telecomunicación (COIT) los gastos de mano de obra de un ingeniero según el salario correspondiente por hora de trabajo se calculan a partir de:

$$H=81 * H_n + 103 H_e (\text{€})$$

Donde:

H: Honorarios.

Hn: Horas dentro de la jornada laboral.

He: Horas en jornada especial.

El valor final al que ascienden estos honorarios se obtiene aplicando un coeficiente de reducción en función del número de horas. El valor de este coeficiente de reducción se obtiene de una tabla especificada por el COIT:

Horas	Factor de corrección
Hasta 36	1
Exceso de 36 hasta 72	0,9
Exceso de 72 hasta 108	0,8
Exceso de 108 hasta 360	0,7
Exceso de 144 hasta 180	0,65
Exceso de 180 hasta 360	0,6
Exceso de 360 hasta 510	0,55
Exceso de 510 hasta 720	0,5
Exceso de 720 hasta 1080	0,45
Exceso de 1080	0,4

Tabla 77: coeficiente de reducción establecido por el COIT

Coste Total Recursos Humanos $H=81*320*0.6+103*20=17.612€$

Los recursos humanos tienen un coste total libre de impuestos de **diecisiete mil seiscientos doce euros**.

8.3 Coste de los Recursos *Hardware*.

El coste total de los recursos *hardware* incluye la compra de un ordenador de sobremesa para poder realizar el proyecto fin de carrera, puesto que el equipo que poseía el proyectante no reunía las condiciones suficientes para garantizar el desarrollo de la aplicación, y el uso de servicio de internet.

Concepto	Coste Adquisición	Coste mensual	Período empleado	Importe
Componentes Ordenador Sobremesa	250,00 €	-	4 meses	250,00 €
Servicio internet	-	13,00 €	4 meses	52,00 €
Coste total de los recursos <i>Hardware</i>				302,00 €

Tabla 78: Tabla de costes de los recursos *Hardware*.

Los Recursos *Hardware* tienen un coste total libre de impuestos de **trescientos dos euros**.

8.4 Coste de los Recursos *Software*.

Como ya se comentó en esta memoria, el HUGCDN apuesta por los recursos *software* de licencia libre, así que para el desarrollo de este trabajo no supone **ningún coste añadido**, a excepción de la licencia del S.O. utilizado.

Concepto	Tiempo empleado	Coste adquisición	Importe
Gestor de BB.DD. <i>PostgreSQL</i> <i>pgAdminIII</i>	4 meses	0,00 €	0,00 €
Marco de trabajo <i>Openxava</i>	4 meses	0,00 €	0,00 €
Herramienta IDE <i>Eclipse</i>	4 meses	0,00 €	0,00 €
<i>Ireport 5.1</i>	4 meses	0,00 €	0,00 €
<i>OpenOffice Writer</i>	4 meses	0,00 €	0,00 €
Sistema Operativo <i>Window</i>	4 meses	0,00 €	3,00 €
Coste Total Recursos <i>Software</i>			3,00 €

Tabla 79: Tabla de costes de los recursos *Software*.

Los Recursos *Software* asociados tienen un coste de **tres euros**.

8.5 Coste de la Redacción.

Concepto	Importe
Desarrollo de la documentación	0,00 €
Encuadernación y enmarcado	130,00 €
Coste de Redacción e Impresión	130,00 €

Tabla 80: Tabla de costes de redacción.

Los Gastos de Redacción e Impresión ascienden a **ciento treinta euros**.

8.6 Coste total.

Concepto	Importe
Coste Total Recursos Humanos	17.612,00 €
Coste Total Recursos <i>Hardware</i>	302,00 €
Coste Total Recursos <i>Software</i>	3,00 €
Coste de Redacción e Impresión	130,00 €
Total	18.047,00 €
Impuestos IGIC 7%	1.263,29 €
TOTAL	19.307,29 €

Tabla 81: Tabla de costes totales.

El trabajo “Aplicación para el control de la estimulación cardíaca, protocolos de pruebas médicas diagnósticas y seguimiento de pacientes de cardiología del Hospital Universitario de Gran Canaria DR. Negrín”, asciende a un coste total de **diecinueve mil trescientos siete euros con veintinueve céntimos. (19.307,29 €)**.

Bibliografía.

Bibliografía.

[1] Terapia de resincronización cardíaca:

<http://www.medtronic.es/su-salud/insuficiencia-cardiaca/dispositivo/que-es/>

[Última visita: 6 de Marzo de 2017].

[2] Marco de trabajo Ajax para desarrollo rápido de aplicaciones web empresariales
Openxava:

<http://www.openxava.org/es/>

[Última visita: 6 de Marzo de 2017].

[3] ISO/IEC 12207:2008, *Systems and software engineering -- Software life cycle processes*:

http://www.iso.org/iso/catalogue_detail?csnumber=43447

[Última visita: 6 de Marzo de 2017].

[4] ISO/IEC TR 15.504/SPICE:

<http://www.aec.es/web/guest/centro-conocimiento/spice>

[Última visita: 6 de Marzo de 2017].

[5] Sistemas de gestión de la calidad. Requisitos. (ISO 9001:2000):

<http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0023966#.VkMtLXohtz0>

[Última visita: 6 de Marzo de 2017].

[6] Sistemas de gestión de la calidad. Fundamentos y vocabulario. (ISO 9000:2000):

<http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0023965#.VkMtoHohtz0>

[Última visita: 6 de Marzo de 2017].

[7] IEEE 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*:

<https://standards.ieee.org/findstds/standard/610.12-1990.html>

[Última visita: 6 de Marzo de 2017].

[8] *Structured systems analysis and design method* (SSADM):

https://en.wikipedia.org/wiki/Structured_systems_analysis_and_design_method

[Última visita: 6 de Marzo de 2017].

[9] Metodología *Merise*:

<https://en.wikipedia.org/wiki/Merise>

[Última visita: 6 de Marzo de 2017].

[10] "*Information engineering*," part 3, part 4, part 5, Part 6" by Clive Finkelstein. In *Computerworld*, In depths, appendix. May 25 - June 15, 1981.

[Última visita: 6 de Marzo de 2017].

[11] MAGERIT v.3 : Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información:

http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Magerit.html#.VkMxr3ohtz0

[Última visita: 6 de Marzo de 2017].

[12] EUROMETODO:

<http://tienda.boe.es/detail.html?id=9788434010475>

[Última visita: 6 de Marzo de 2017].

[13] *Health Level Seven* (HL7 Internacional):

<http://www.hl7.org/implement/standards/>

[Última visita: 6 de Marzo de 2017].

[14] ISO/IEC 27000:2014:

<https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-3:v1:en>

[Última visita: 6 de Marzo de 2017].

[15] *Information Technology Infrastructure Library* (ITIL):

<http://www.itilcertification.org/>

[Última visita: 6 de Marzo de 2017].

[16] Ley Oficial de Protección de Datos:

https://www.agpd.es/portalwebAGPD/canaldocumentacion/legislacion/estatal/common/pdfs/2014/Ley_Organica_15-

[1999_de_13_de_diciembre_de_Proteccion_de_Datos_Consolidado.pdf](https://www.agpd.es/portalwebAGPD/canaldocumentacion/legislacion/estatal/common/pdfs/2014/Ley_Organica_15-1999_de_13_de_diciembre_de_Proteccion_de_Datos_Consolidado.pdf)

[Última visita: 6 de Marzo de 2017].

[17] Servicios web:

https://es.wikipedia.org/wiki/Servicio_web

[Última visita: 20 de Marzo de 2017].

[18] *Hypertext Transfer Protocol* (HTTP):

<https://tools.ietf.org/html/rfc2616>

[Última visita: 20 de Marzo de 2017].

[19] *Transmission Control Protocol* (TCP):

<https://tools.ietf.org/html/rfc793>

[Última visita: 20 de Marzo de 2017].

[20] *Firewall*:

<http://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html>

[Última visita: 20 de Marzo de 2017].

[21] Cliente-Servidor:

<https://es.wikipedia.org/wiki/Cliente-servidor>

[Última visita: 20 de Marzo de 2017].

[22] Cliente ligero:

https://es.wikipedia.org/wiki/Cliente_liviano

[Última visita: 20 de Marzo de 2017].

[23] *WorldWideWeb*:

<https://www.w3.org/>

[Última visita: 20 de Marzo de 2017].

[24] ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection -- Basic Reference Model: The Basic Model*.

<https://www.iso.org/standard/20269.html>

[Última visita: 20 de Marzo de 2017].

[25] Manifiesto por el desarrollo ágil de software:

<http://agilemanifesto.org/iso/es/manifesto.html>

[Última visita: 01 de Abril de 2017].

[26] Ciclo del desarrollo ágil de aplicaciones:

<http://www.northware.mx/desarrollo-en-cascada-waterfall-vs-desarrollo-agile-scrum/>

[Última visita: 01 de Abril de 2017].

[27] *Middleware*:

<https://es.wikipedia.org/wiki/Middleware>

[Última visita: 03 de Abril de 2017].

[28] Entorno de desarrollo web:

<http://lml.ls.fi.upm.es/ep/entornos.html#toc3>

[Última visita: 08 de Abril de 2017].

[29] Herramientas IDE:

https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

[Última visita: 08 de Abril de 2017].

[30] Sistema de información:

https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n

[Última visita: 6 de Marzo de 2017].

[31] Factores críticos de éxito:

http://itilv3.osiatis.es/proceso_mejora_continua_servicios_TI/metricas.php

[Última visita: 7 de Marzo de 2017].

[32] Objetivos generales, estratégicos y específicos:

<http://es.workmeter.com/blog/bid/273265/7-diferencias-entre-Objetivos-estrat-gicos-y-Objetivos-espec-ficos>

[Última visita: 7 de Marzo de 2017].

[33] Hipertensión Pulmonar:

<http://www.archbronconeumol.org/index.php?p=watermark&idApp=UINPBA00003Z&piiltem=S0300289611700539&origen=bronco&web=bronco&urlApp=http://www.archbronconeumol.org&estadoItem=S300&idiomaltem=es>

[Última visita: 7 de Marzo de 2017].

[34] Resincronización Cardíaca:

http://www.heartfailurematters.org/es_ES/%C2%BFQu%C3%A9-puede-hacer-su-m%C3%A9dico%3F/Tratamiento-de-resincronizacion-cardiaca-TRC

[Última visita: 7 de Marzo de 2017].

[35] Ecocardiografía Transtorácica:

https://es.wikipedia.org/wiki/Ecocardiograf%C3%ADa#Ecocardiograf.C3.ADa_transtor.C3.A1cica

[Última visita: 7 de Marzo de 2017].

[36] Ergometría:

<https://es.wikipedia.org/wiki/Ergometr%C3%ADa>

[Última visita: 7 de Marzo de 2017].

[37] *Holter*:

<https://www.nlm.nih.gov/medlineplus/spanish/ency/article/003877.htm>

[Última visita: 7 de Marzo de 2017].

[38] Cateterismo cardíaco:

<http://www.fundaciondelcorazon.com/informacion-para-pacientes/metodos-diagnosticos/cateterismo-cardiaco.html>

[Última visita: 7 de Marzo de 2017].

[39] Estudio electrofisiológico:

<http://www.fundaciondelcorazon.com/informacion-para-pacientes/metodos-diagnosticos/estudio-electrofisiologico.html>

[Última visita: 7 de Marzo de 2017].

[40] Desfibrilador:

<http://www.fundaciondelcorazon.com/informacion-para-pacientes/tratamientos/desfibrilador.html>

[Última visita: 7 de Marzo de 2017].

[41] Marcapasos:

<http://www.fundaciondelcorazon.com/informacion-para-pacientes/tratamientos/marcapasos.html>

[Última visita: 7 de Marzo de 2017].

[42] *Holter*:

<http://www.madrid.org/cs/Satellite?blobcol=urldata&blobheader=application%2Fpdf&blobheadername1=Content-disposition&blobheadername2=cadena&blobheadervalue1=filename%3DHolter+Implantable.pdf&blobheadervalue2=language%3Des%26site%3DHospitalGregorioMaranon&blobkey=id&blobtable=MungoBlobs&blobwhere=1352809018721&ssbinary=true>

[Última visita: 7 de Marzo de 2017].

[43] *Electrodos*:

<http://www.revespcardiol.org/es/implante-electrodos-epicardicos-el-ventriculo/articulo/13059723/>

[Última visita: 7 de Marzo de 2017].

[44] *Electrodo ventricular de fijación pasiva*:

http://www.enfermeriaencardiologia.com/grupos/electrofisiologia/investiga/electrodo_01/03.htm

[Última visita: 7 de Marzo de 2017].

[45] *Modelado de los procesos de la organización*:

<https://manuel.cillero.es/doc/metrica-3/tecnicas/modelado-de-procesos-de-la-organizacion/>

[Última visita: 13 de Marzo de 2017].

[46] *Drago web*:

<https://www.gobiernodecanarias.org/dragoweb/index.htm#>

[Última visita: 13 de Marzo de 2017].

[47] *Active Directory de Windows Server 2008*:

https://es.wikipedia.org/wiki/Active_Directory

[Última visita: 13 de Marzo de 2017].

[48] *Descarga StarUML*:

<http://staruml.io/download>

[Última visita: 13 de Marzo de 2017].

[49] Especificación de requisitos de software:

https://es.wikipedia.org/wiki/Especificaci%C3%B3n_de_requisitos_de_software

[Última visita: 13 de Marzo de 2017].

[50] Estándar IEEE 830-1998:

<https://standards.ieee.org/findstds/standard/830-1998.html>

[Última visita: 13 de Marzo de 2017].

[51] Identificación de los riesgos:

Ingeniería del Software. Un enfoque práctico. Séptima edición. Roger S. Pressman. McGraw-Hill. Año 2010, página 642.

<http://cotana.informatica.edu.bo/downloads/Id->

[Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF](http://cotana.informatica.edu.bo/downloads/Id-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF)

[Última visita: 18 de Marzo de 2017].

[52] Identificación de los riesgos:

Ingeniería del Software. Un enfoque práctico. Séptima edición. Roger S. Pressman. McGraw-Hill. Año 2010, página 642, apartado 28.3.

<http://cotana.informatica.edu.bo/downloads/Id->

[Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF](http://cotana.informatica.edu.bo/downloads/Id-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF)

[Última visita: 18 de Marzo de 2017].

[53] Estimación del riesgo:

Ingeniería del Software. Un enfoque práctico. Séptima edición. Roger S. Pressman. McGraw-Hill. Año 2010, página 644, apartado 28.4.

<http://cotana.informatica.edu.bo/downloads/Id->

[Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF](http://cotana.informatica.edu.bo/downloads/Id-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF)

[Última visita: 18 de Marzo de 2017].

[54] Valoración de impacto:

Boehm, B. W., Software Risk Management, IEEE Computer Society Press, 1989

[Última visita: 18 de Marzo de 2017].

[55] Planificación de los riesgos:

Ingeniería del Software. Un enfoque práctico. Séptima edición. Roger S. Pressman. McGraw-Hill. Año 2010, página 649, apartado 28.6.

<http://cotana.informatica.edu.bo/downloads/ld->

[Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF](http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF)

[Última visita: 18 de Marzo de 2017].

[56] Supervisión de los riesgos:

Ingeniería del Software. Un enfoque práctico. Séptima edición. Roger S. Pressman. McGraw-Hill. Año 2010

<http://cotana.informatica.edu.bo/downloads/ld->

[Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF](http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF)

[Última visita: 18 de Marzo de 2017].

[57] *Openxava*:

<http://openxava.org/es>

[Última visita: 13 de Marzo de 2017].

[58] *Framework*:

<https://es.wikipedia.org/wiki/Framework>

[Última visita: 13 de Marzo de 2017].

[59] *PostgreSQL*:

<http://www.postgresql.org.es/>

[Última visita: 13 de Marzo de 2017].

[60] *Data source*:

<http://www.um.es/atica/acceso-a-base-de-datos-desde-aplicaciones-web-en-java-4>

[Última visita: 10 de Abril de 2017].

[61] Clave primaria:

https://www.ibm.com/support/knowledgecenter/es/SSGU8G_11.50.0/com.ibm.ddi.doc/ids_ddi_182.htm

[Última visita: 10 de Abril de 2017].

[62] Clave foránea:

https://www.ibm.com/support/knowledgecenter/es/SSGU8G_11.50.0/com.ibm.ddi.doc/ids_ddi_183.htm

[Última visita: 10 de Abril de 2017].

[63] Sistema *PostgreSQL*:

http://www.postgresql.org.es/sobre_postgresql

[Última visita: 10 de Abril de 2017].

[64] Web oficial *Hibernate*:

<http://hibernate.org/validator/>

[Última visita: 13 de Abril de 2017].

[65] Tipos de anotaciones *Hibernate*:

http://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/#chapter-bean-constraints

[Última visita: 16 de Abril de 2017].

[66] Tipos de anotaciones *Hibernate*:

http://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/#chapter-bean-constraints

[Última visita: 16 de Abril de 2017].

[67] *Java Databases Connectivity* (JDBC):

https://es.wikipedia.org/wiki/Java_Database_Connectivity

[Última visita: 13 de Abril de 2017].

[68] SQL:

<http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt>

[Última visita: 13 de Abril de 2017].

[69] IDE's de escritorio ECLIPSE:

<http://www.eclipse.org/ide/>

[Última visita: 09 de Abril de 2017].

[70] Representación de la estructura de Eclipse:

<http://help.eclipse.org/neon/index.jsp?nav=%2F0>

[Última visita: 09 de Abril de 2017].

[71] Arquitectura del directorio:

<http://tomcat.apache.org/tomcat-6.0-doc/introduction.html>

[Última visita: 16 de Abril de 2017].

[72] Definición de tipo de documento (DTD`s):

http://www.mclibre.org/consultar/xml/lecciones/xml_dtd.html

[Última visita: 17 de Abril de 2017].

[73] JDBC:

<http://docs.oracle.com/javase/tutorial/jdbc/overview/index.html>

[Última visita: 16 de Abril de 2017].

[74] *JavaServer Pages™ Specification*:

<http://www.oracle.com/technetwork/java/index-141364.html#specs>

[http://download.oracle.com/otn-pub/jcp/jsp-2_3-mrel2-spec/JSP2.3MR.pdf?](http://download.oracle.com/otn-pub/jcp/jsp-2_3-mrel2-spec/JSP2.3MR.pdf?AuthParam=1492536912_5d8258073aeacceccc35f92bf121bff5)

[AuthParam=1492536912_5d8258073aeacceccc35f92bf121bff5](http://download.oracle.com/otn-pub/jcp/jsp-2_3-mrel2-spec/JSP2.3MR.pdf?AuthParam=1492536912_5d8258073aeacceccc35f92bf121bff5)

[Última visita: 18 de Abril de 2017].

[75] PHP:

<https://secure.php.net/>

[Última visita: 18 de Abril de 2017].

[76] Arquitectura *JavaServer Pages*:

https://en.wikipedia.org/wiki/JavaServer_Pages

[Última visita: 18 de Abril de 2017].

[77] Variables implícitas *JavaServer Pages*:

https://es.wikipedia.org/wiki/JavaServer_Pages

[Última visita: 18 de Abril de 2017].

[78] *Model-Driven Software Development:*

Technology, Engineering, Management, Thomas Stahl, Markus Voelter, pag 5.

<http://www.voelter.de/data/books/mdsd-en.pdf>

[Última visita: 02 de Abril de 2017].

[79] *Java Server Pages:*

<http://java.sun.com/products/jsp/>

[Última visita: 02 de Abril de 2017].

[80] *Java Server Faces:*

<https://www.jcp.org/en/jsr/detail?id=314>

[Última visita: 02 de Abril de 2017].

[81] Librería gráfica para *Java*:

<https://docs.oracle.com/javase/7/docs/api/javafx/swing/package-summary.html>

[Última visita: 02 de Abril de 2017].

[82] Estereotipos de OX:

https://openxava.wikispaces.com/model_es#Modelo-Propiedades-Estereotipo

[Última visita: 19 de Abril de 2017].

[83] Ventajas de *Openxava* como marco de trabajo:

<http://openxava.org/es/>

[Última visita: 20 de Abril de 2017].

[84] JSR 317: *Java Persistence API (JPA)*

<https://www.jcp.org/en/jsr/detail?id=317>

[Última visita: 21 de Abril de 2017].

[85] JSR 303: *Bean Validation.*

<https://www.jcp.org/en/jsr/detail?id=303>

[Última visita: 21 de Abril de 2017].

[86] JSR 330: *Dependency Injection for Java.*

<https://www.jcp.org/en/jsr/detail?id=330>

[Última visita: 21 de Abril de 2017].

[87] JSR-220: *Enterprise JavaBean 3.0*.

<https://www.jcp.org/en/jsr/detail?id=220>

[Última visita: 21 de Abril de 2017].

[88] JSR-153: *Enterprise JavaBean 2.1*.

<https://www.jcp.org/en/jsr/detail?id=153>

[Última visita: 21 de Abril de 2017].

[89] JSR-168: *Portlet Specification*.

<https://www.jcp.org/en/jsr/detail?id=168>

[Última visita: 21 de Abril de 2017].

[90] JSR-286: *Portlet Specification 2.0*.

<https://www.jcp.org/en/jsr/detail?id=286>

[Última visita: 21 de Abril de 2017].

[91] *Hibernate* como proveedor JPA:

<http://hibernate.org/orm/>

[Última visita: 21 de Abril de 2017].

[92] Comparativa aplicación web clásica y aplicación web con tecnología *Ajax*:

<http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion01-apuntes.html>

[Última visita: 22 de Abril de 2017].

[93] XHR:

<https://xhr.spec.whatwg.org/>

[Última visita: 22 de Abril de 2017].

[94] Árbol DOM *Javascript*:

https://www.w3schools.com/js/js_htmlDOM.asp

[Última visita: 22 de Abril de 2017].

[95] *Ireport*:

<https://sourceforge.net/projects/ireport/files/iReport/iReport-5.5.1/>

[Última visita: 22 de Abril de 2017].

[96] *JasperReport*:

<http://community.jaspersoft.com/project/jasperreports-library>

[Última visita: 23 de Abril de 2017].

[97] jrxml, JasperReports Library Ultimate Guide, página 40:

<http://community.jaspersoft.com/documentation/jasperreports-library-ultimate-guide>

[Última visita: 24 de Abril de 2017].

[98] *iText*:

<http://itextpdf.com/>

[Última visita: 24 de Abril de 2017].

[99] Aplicación *cports*:

<http://www.nirsoft.net/utills/cports.html>

[Última visita: 17 de Abril de 2017].

[100] Descarga del controlador de *PostgreSQL* – JDBC:

<https://jdbc.postgresql.org/download.html>

[Última visita: 16 de Abril de 2017].

[101] *Javax.persistence.EntityManager*:

<http://docs.oracle.com/javaee/6/api/javax/persistence/EntityManager.html>

[Última visita: 21 de Abril de 2017].

[102] Interface *EntityManager*:

[http://docs.oracle.com/javaee/6/api/javax/persistence/EntityManager.html#setProperty\(java.lang.String, java.lang.Object\)](http://docs.oracle.com/javaee/6/api/javax/persistence/EntityManager.html#setProperty(java.lang.String,java.lang.Object))

[Última visita: 21 de Abril de 2017].

[103] *Java SE Development Kit 7u79*:

<http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html>

[Última visita: 24 de Abril de 2017].

[104] Descarga del fichero *ojdbc14.jar*:

<http://www.java2s.com/Code/Jar/o/Downloadojdbc14jar.htm>

Ultima visita [02 de Mayo de 2017].

[105] Ley 25/2009, de 22 de diciembre, de modificación de diversas leyes para su adaptación a la Ley sobre el libre acceso a las actividades de servicios y su ejercicio

<https://www.boe.es/buscar/doc.php?id=BOE-A-2009-20725>

Ultima visita [02 de Mayo de 2017].