UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

SISTEMA DE GESTIÓN DE CAMPEONATOS Y GENERACIÓN DE CALENDARIO DE PÁDEL

Josué Caballero López
Las Palmas de Gran Canaria, 06 de Junio de 2017

Proyecto fin de carrera de la Facultad de Informática de la Universidad de Las Palmas de Gran Canaria presentado por el alumno:

JOSUÉ CABALLERO LÓPEZ

Título del Proyecto: Desarrollo de un sistema de gestión de campeonatos y generación de calendario de pádel.

Tutor: D. Alexis Quesada Arencibia

Índice de Contenidos

Índice de Contenidos 2
Índice de ilustraciones 5
Índice de tablas 6
1. Introducción 7
2. Objetivos 7
2.1. De la aplicación 7
2.2. Del desarrollo 8
3. Estructura del documento 8
4. Estado del arte 9
4.1. Complejidad del problema 9
4.2. Técnicas de resolución empleadas 10
4.3. Estudio de herramientas 10
4.3.1. TodoTorneos.com 10
4.3.2. Xporty 11
4.3.3. Pádel manager 13
4.4. Conclusiones del estudio de herramientas 14
5. Metodología 14
5.1. Modelo de proceso de software 14
5.2. Lenguaje de modelado 18
5.3. Metodología aplicada 18
5.4. Planificación 18
6. Recursos necesarios 19
6.1. Recursos hardware 19
6.1.1. Servidor web 19
6.2. Recursos software 19
6.2.1. Servicio Hypnotoad 19
6.2.2. Servicio PostgreSQL 9 o superior 19
6.2.3. Mojolicious 19
6.2.4. Putty 19
6.2.5. Git 20
6.2.6. Vim 20
6.2.7. Psql 20
6.2.8. pgAdmin 20
6.2.9. Sistema operativo Windows 10 20

6.2.10. Microsoft Word 2010 21
6.2.11. StarUML 21
6.3. Tecnologías utilizadas 21
6.3.1. Perl 21
6.3.2. PostgreSQL 21
6.3.3. HTML 22
6.3.4. JavaScript 22
6.3.5. jQuery 22
6.3.6. CSS 22
6.3.7. AJAX 23
6.3.8. Bootstrap 23
6.3.9. DataTables 23
6.3.10. ApiRest 24
6.3.11. Curl 24
6.3.12. CPAN 25
7. Análisis 25
7.1. Introducción 25
7.2. Requisitos del software 25
7.2.1. Introducción 25
7.2.2. Modelo conceptual 26
7.2.3. Identificación de actores 27
7.2.4. Diccionario de conceptos 27
7.2.5. Actores/Objetivos 28
7.2.6. Listado resumen de los casos de uso 29
7.2.7. Diagramas de casos de uso 30
7.2.8. Casos de uso completos 31
7.3. Prototipos de validación 34
8. Diseño 38
8.1. Introducción 38
8.2. Arquitectura Cliente-Servidor 38
8.3. Patrón MVC (Modelo Vista Controlador) 39
8.4. Diseño arquitectónico 39
8.5. Base de datos 41
8.5.1. Tablas 42
8.6. Estructura de la aplicación 45
8.7. Interfaz 46

8.7.1. Importancia de la interfaz 46
8.7.2. Estructura y diseño 46
8.8. Diagrama de clases 47
8.9. Diagramas de secuencia 49
8.9.1. Sistema de gestión 49
9. Desarrollo 51
9.1. Detalles de implementación 51
9.1.1. Introducción 51
9.1.2. Estructura interna del sistema 52
9.1.3. Generar competiciones 52
9.1.4. Heurística para la asignación de pistas 53
9.2. Librerías externas 54
9.2.1. DataTables 54
9.2.2. Bootstrap 55
10. Pruebas del sistema 55
11. Resultados y conclusiones 57
11.1. Resultados 57
11.1.1. Interfaz 57
11.1.2. Gestión de torneos 58
11.1.3. Generación de torneos 58
11.1.4. Asignación de pistas 58
11.1.5. Base de Datos 58
11.1.6. Accesibilidad y optimización 59
11.1.7. Aportaciones en el contexto de herramientas de gestión de competiciones de pádel 59
11.2. Conclusiones 59
12. Trabajo futuro 60
13. Bibliografía 61
13.1. Libros. 61
13.2. Web 61
14. Anexos 62
14.1. ANEXO A: Manual de usuario 62
14.2. Anexo B: Guía de instalación 64
14.3. Anexo C: Plantilla de casos de uso 65
14.4. Anexo D: Base de datos 73

Índice de ilustraciones

Ilustración 1. Paradigma de construcción de prototipos 15
Ilustración 2. Ciclo del paradigma de construcción de prototipos 16
Ilustración 3. Modelo en espiral 17
Ilustración 4. Casos de uso del usuario anónimo 30
Ilustración 5. Casos de uso del usuario registrado 31
Ilustración 6. Casos de uso del usuario administrador 31
Ilustración 7. Gestión de horarios de los jugadores 35
Ilustración 8. Listado de jugadores 35
Ilustración 9. Configurador de competiciones 36
Ilustración 10. Calendario de asignación de pistas 37
Ilustración 11. Edición de pistas 37
Ilustración 12. Modelo vista controlador 39
Ilustración 13. Diseño arquitectónico 40
Ilustración 14. Ejemplo de registro de un usuario 41
Ilustración 15. Esquema de la base de datos 42
Ilustración 16. Detalle de almacenamiento de equipos 42
Ilustración 17. Detalle de almacenamiento de pistas 43
Ilustración 18. Detalle de competiciones, categorías y partidos 43
Ilustración 19. Tabla de almacenamiento de disponibilidad 44
Ilustración 20. Estructura de la aplicación 45
Ilustración 21.Ejemplo de diseño de la aplicación 47
Ilustración 22. Modelo vista controlador 48
Ilustración 23. Estructura de Mojolicious 48
Ilustración 24. Diagrama de clases 49
Ilustración 25. Diagrama del sistema de gestión 50
Ilustración 26. Menú superior de la aplicación 52

Índice de tablas

Tabla 1. Identificación de actores 27
Tabla 2. Tabla de jerarquías 27
Tabla 3. Lista actor-objetivo 29
Tabla 4. Resumen de los casos de uso 30
Tabla 5. Protocolo RestFull 50

1.Introducción

A través de la experiencia en la organización de diversos torneos de pádel, se ha podido observar que la organización de estos eventos conlleva un gran gasto de tiempo, tendiendo éste a crecer en gran medida si se hace un especial énfasis en la organización del horario de forma que este se adecúe a las necesidades de todos los participantes del mismo.

Con este proyecto se propone desarrollar una herramienta capaz de gestionar cualquier competición de pádel, así como la generación de las jornadas dependiendo de la disponibilidad de cada jugador. También abarca la gestión de los cambios de las jornadas y la gestión de los resultados obtenidos.

El proyecto se plantea como un sistema integral de gestión para torneos de pádel. Pretende ser un lugar de encuentro entre jugadores, organizadores de torneos y clubes de pádel, facilitando en gran medida la gestión de cada uno de los actores.

Este proyecto sirve igualmente para conocer todo el proceso necesario para aprender cómo gestionar y desarrollar un proyecto, desde la idea hasta su ejecución final, superando los obstáculos que se presentan durante el trayecto.

2.Objetivos

Dentro de los objetivos del proyecto fin de carrera diferenciaremos dos. Los objetivos que queremos conseguir con el software y los objetivos relacionados con la adquisición de competencias con algunas herramientas que se emplean durante el desarrollo.

2.1.De la aplicación

El objetivo principal de este proyecto es el desarrollo de una aplicación web que permita la gestión de torneos de pádel. El proceso facilitará la gestión de cada uno de los roles participantes en el torneo.

Por un lado a los organizadores del torneo se les facilita la generación de competiciones, permitiendo añadir las diferentes categorías y el formato de competición de cada una de ellas. Así mismo se facilita la selección de fechas de inscripción y del número máximo de inscritos.

Por otro lado los jugadores tendrán una herramienta donde poder indicar su disponibilidad horaria para facilitar la generación de jornadas. Así mismo tendrán un lugar donde poder inscribirse en el torneo, introducir los resultados de los partidos y seguir la clasificación de las competiciones donde estén inscritos.

Los objetivos del proyecto serían, por lo tanto:

- Facilitar la inscripción de las parejas integrantes al torneo.

- Permitir a las parejas seleccionar su horario de disponibilidad para disputar los partidos.
- Generar las jornadas basándose en los horarios definidos por las parejas y la disponibilidad del recinto de juego.
- Permitir gestionar los cambios en las jornadas, pudiendo cambiar los horarios de los partidos.
- Permitir añadir y validar los resultados de los partidos.
- Generar automáticamente la clasificación.

2.2.Del desarrollo

Existen una serie de objetivos para el desarrollo de la aplicación, entre los que destacan los siguientes

- Uso de software libre.
- Instalación y uso de un Servidor Web Hypnotoad.
- Instalación y uso de servicios Rest sobre Hypnotoad.
- Uso de frameworks de desarrollo basados en Perl (mojolicious).
- Sistema de gestión de BBDD postgresql.
- Uso de dbdesigner.
- Plataformas de modelado, StarUml y ArgoUml.
- Plataformas de desarrollo, Vim.

Con el uso de software libre se persiguen dos objetivos. Disminuir los costes, debido al ahorro de uso de licencias de propietario y sacar ventaja de la comunidad de desarrolladores de proyectos de software libre. También hay que destacar que las herramientas que su uso están bastante contrastadas por su larga trayectoria como método de desarrollo de aplicaciones Web.

3.Estructura del documento

Con el objetivo de facilitar la lectura del presente documento, se presenta en este apartado la estructura general del mismo.

Tras haber realizado la introducción del proyecto y después de haber definido los objetivos del mismo, se explicarán cada una de las fases por las que ha pasado el proyecto.

En primer lugar se presenta el estudio sobre el Estado del Arte y las herramientas existentes antes de comenzar el proyecto. A continuación, se explica la metodología utilizada durante la elaboración de la aplicación así como los recursos utilizados.

En los siguientes apartados se detallan las fases de análisis, diseño y desarrollo del proyecto.

Seguidamente, se presentan los resultados obtenidos, las conclusiones y el trabajo futuro que podría realizarse para ampliar este proyecto.

Por último, se muestra la bibliografía utilizada y se presentan los anexos del documento, donde puede encontrarse el manual de usuario y la guía de instalación.

En cada una de las fases presentadas en este documento se podrán encontrar diferentes apartados. El objetivo es facilitar la lectura y comprensión. A lo largo del documento también se citarán las referencias bibliográficas consultadas. El formato de cada referencia es el código correspondiente a cada libro o sitio web entre corchetes, por ejemplo: [PRE03] hace referencia al libro escrito por Roger S. Pressman sobre "Ingeniería del Software. Un enfoque práctico".

El "Anexo A – Manual de usuario" contiene toda la información sobre las funcionalidades de la aplicación. Además, se acompaña cada explicación con imágenes para facilitar la comprensión.

Junto a este documento se anexa un CD con el código fuente, éste documento, una guía de instalación y la imagen de la máquina virtual donde se ha llevado a cabo el proyecto.

4.Estado del arte

En este apartado se pretende conocer en qué punto se encuentra el tema que se está estudiando así como las tendencias que se están siguiendo en nuevos desarrollos.

Revisar lo que se ha trabajado en torno al problema que se está abordando, las herramientas metodológicas utilizadas, las conclusiones alcanzadas y las sugerencias propuestas, permite dar un paso adelante en la acumulación de conocimiento para poder innovar con lo que se pretende desarrollar.

4.1.Complejidad del problema

Para poder elaborar una herramienta que permita dar solución al problema de gestión y generación de calendarios de pádel es necesario conocer las investigaciones que se han realizado hasta el momento.

Las técnicas de generación de torneos se clasifican según el deporte al que estén designados, siendo la liga y el sistema por eliminación los más comunes en el pádel. Los torneos tipo americano están siendo cada vez más habituales por los aficionados.

El sistema de generación de ligas responde al sistema round-robin, por el que todos los equipos juegan contra todos a un número de vueltas determinados. El ganador es el que más partidos gana al finalizar las vueltas designadas.

El sistema por eliminatoria difiere ya que se deben determinar los cruces entre los equipos desde el comienzo del torneo.

El tercer sistema, denominado americano, es aquel en el que juegan todos contra todos, siendo similar al round-robin, la diferencia radica en que en algunos casos no solo rotan los equipos, sino también los jugadores.

Una vez definidas las jornadas, la otra complejidad encontrada, y que es el objetivo principal de este proyecto, es la asignación de horarios y pistas a los partidos designados.

Esta asignación de pistas puede definirse como "la asignación, sujeta a restricciones, de los recursos otorgados con el propósito de ser establecidos en un espacio de tiempo, de tal manera que satisfaga lo más cercanamente posible el conjunto de objetivos deseados".

El desafío en la asignación de pistas radica en proponer una jornada que cumpla con los requisitos de disponibilidad de horas de los jugadores y de las pistas disponibles por los clubes. Esta asignación debe poder asignar pista al mayor número de partidos posibles.

El desafío en la asignación de pistas radica en proponer una planificación que cumpla con todas las restricciones impuestas y obtenga una solución de calidad aceptable en un tiempo razonable. Conseguir este objetivo no es una tarea sencilla. La asignación de pistas pertenece a la categoría de problemas NP-Completos, en los cuales existe una proporción directa entre el tiempo y la calidad de la solución. La clase NP-Completo es una manera de demostrar que cierto tipo de problemas no son resolubles en tiempo real.

4.2.Técnicas de resolución empleadas

Las técnicas para asignar horarios se basaban en la forma humana de resolver el problema. El horario se va extendiendo hasta que todos los partidos queden asignados, o bien acercarse a la solución más cercana, donde el número de partidos sin asignar sean mínimos.

Para ello se debe buscar una solución en la que aquellos partidos donde la disponibilidad horaria sea más restrictiva se pueda asignar a los momentos dónde la disponibilidad de recursos (pistas) sea máxima.

4.3.Estudio de herramientas

En este apartado se realiza una investigación de las herramientas existentes en el mercado que puedan responder a los requisitos del proyecto.

Para la elección de las aplicaciones que se estudian a continuación se han tenido en cuenta los siguientes aspectos importantes:

- Ser de uso extendido.
- Ser de fácil utilización por parte de los administradores.
- Asignación de pistas a partidos.

4.3.1.TodoTorneos.com

Todotorneos es un sitio web fundado en 2007 que permite organizar, gestionar y publicar un torneo o competición. Todo se hace desde el entorno web por lo que no

tendrá que descargarse ningún software adicional y es posible acceder a dicha gestión desde cualquier ordenador.

Características

- Completamente gratis: sin necesidad de tener conocimientos técnicos
- Área independiente: Cada torneo tiene su área independiente, o lo que es lo mismo su propia web, tanto para gestionarlo como para publicarlo.
- Diferentes formatos y fondos: Se puede elegir entre diferentes formatos de fondos y colores para la web del torneo
- Diferentes tipos de competiciones: Se pueden crear ligas, rankings, eliminatorias, torneos combinados (clasificación tipo liguilla y Fase Final Eliminatoria, como el mundial de fútbol, la Liga ACB...)
- Publicar toda la información: Se publica la clasificación y resultados por jornada, historial de partidos jugados y pendientes por equipo, página de noticias, normativa, galería de fotos, etc.
- Generación de cruces: Generación automática de cruces de ligas, eliminatorias, sorteo automático de grupos...
- Clasificación: Clasificación personalizada. Permite personalizar los puntos por victoria, por derrota, por empate, clasificaciones por partidos ganados, por sets.... en base a los resultados la clasificación se calculará automáticamente según la configuración.
- Utilidades: Galería de fotos, página de contacto propia, número de visitas de la página,...
- Calendario: Establecer y publicar el calendario para los partidos a disputar.
- Mapa: Poner un mapa con el callejero de Google maps incorporado señalando la ubicación del torneo.
- Gestión de Inscripción: Permite que los equipos puedan inscribirse a través de la propia web, abrir o cerrar la inscripción a voluntad, establecer cupos de inscripción y gestionar listas de reservas.
- Gratuito. Todotorneos se limita a poner publicidad.

4.3.2.Xporty

Xporty es un sitio web fundado en 2010 orientado a deportistas. Incluye red social, completo software online para organización y gestión de torneos, gestión de clubes, equipos y eventos deportivos, ayuda a encontrar gente para practicar deportes, permite gestionar los resultados y llevar un seguimiento estadístico.

Características

- Sistemas de competición: Liga, rankings, copas o sistemas personalizados.
- Página del torneo: Publicación de noticias, documentos, normativas, fotos.
- Inscripciones y participantes: Inscripción online o manualmente.
- Formas de pago: Pago en mano o por transferencia.
- Fases y cuadros: Permite añadir un número ilimitado de fases.
- Comunicación: Permite la comunicación entre los participantes.

- Calendario: Genera de forma manual o automática los emparejamientos y el número de jornadas.
- Clasificación: Genera las clasificaciones automáticamente.
- Administradores: Se puede añadir más administradores o gestores de resultados.
- Árbitros: Asignación de árbitros para la carga de resultados.
- Patrocinadores: Ofrece un espacio para los patrocinadores.
- Privacidad: Torneos públicos o privados.
- Requiere de un ordenador conectado a internet.
- Aplicación para móviles iOs 6.0+ y Android 4.0.3+.

Precio

Precio			
	Básico	Profesional	Premium
	0 €	299€/año	499 €/año
Número de competiciones	Ilimitadas	Ilimitadas	Ilimitadas
Número de participantes	Ilimitadas	Ilimitadas	Ilimitadas
Sistemas de competición	SÍ	SÍ	SÍ
Categorías y divisiones	Ilimitadas	Ilimitadas	Ilimitadas
Múltiples administradores	NO	SÍ	SÍ
Sanciones	NO	SÍ	SÍ
Árbitros	NO	SÍ	SÍ
Circuitos	NO	NO	SÍ
Patrocinadores	NO	NO	SÍ
Envio de emails a los participantes	Hasta 200 participantes	Ilimitados	Ilimitados
Inscripciones			
Inscripciones online	SÍ	SÍ	SÍ
Campos personalizados en el formulario de inscripción	SÍ	SÍ	SÍ
Preferencias horarias al inscribirse	NO	NO	SÍ
Bolsa de jugadores	NO	SÍ	SÍ
Importar participantes de otras competiciones.	SÍ	SÍ	SÍ
Bloqueo de equipos	NO	SÍ	SÍ
Pago en mano	SÍ	SÍ	SÍ
Transferencia bancaria	SÍ	SÍ	SÍ
Control de cobros	SÍ	SÍ	SÍ
Fases y cuadros			
Gestión de pistas	NO	NO	SÍ
Fases ilimitadas	SÍ	SÍ	SÍ
Generación automática de grupos.	SÍ	SÍ	SÍ
Selección de cabezas de serie	SÍ	SÍ	SÍ
Generación automática de cuadros por sorteo	SÍ	SÍ	SÍ
Generación automática de cuadros con horario y pista	NO	NO	SÍ
Generación automática de cuadros de	SÍ	SÍ	SÍ

consolación			
Eliminatorias a doble KO	NO	SÍ	SÍ
Permite eliminatorias a ida, ida y vuelta, 3 o 4 partidos.	SÍ	SÍ	SÍ
Los participantes pueden añadir sus partidos y cargar sus resultados	SÍ	SÍ	SÍ
Clasificaciones			
Generación automática de clasificaciones	SÍ	SÍ	SÍ
Personalización de los campos mostrados en la clasificación.	NO	SÍ	SÍ
Añadir bonificaciones y penalizaciones a los participantes en las clasificaciones	SÍ	SÍ	SÍ
Condiciones para entrar en la clasificación	SÍ	SÍ	SÍ
Informes para imprimir			
Clasificación	SÍ	SÍ	SÍ
Calendario de partidos	SÍ	SÍ	SÍ
Calendario de partidos por pista	NO	NO	SÍ
Página web del torneo			
Dirección web única del torneo	SÍ	SÍ	SÍ
Personalizar imagen de cabecera	SÍ	SÍ	SÍ
Torneos privados	NO	SÍ	SÍ
Sección de noticias	SÍ	SÍ	SÍ
Galería de fotos	SÍ	SÍ	SÍ
Compartir archivos	SÍ	SÍ	SÍ
Más...			
Redes sociales	SÍ	SÍ	SÍ
App móvil iOS y Android para los jugadores	SÍ	SÍ	SÍ
Soporte	Ordinario	Prioritario	Preferente

4.3.3. Pádel manager

Pádel Manager es una plataforma online que ofrece un servicio integral de gestión deportiva, organización y regulación, dando solución a todas las demandas de clubes y jugadores amateur.

Pádel Manager proporciona al club una plataforma que da respuesta a las demandas actuales dando soluciones y ofreciendo un servicio integral de gestión deportiva, organización y regulación. Los factores determinantes del éxito para un club de pádel se basan en cuatro aspectos fundamentales:

- Optimización: Organización y gestión.
- Mejora: Marketing y publicidad.
- Incremento: Calidad del servicio.
- Aumento: Satisfacción del cliente.

Dentro de sus características principales podemos destacar las siguientes:

- Registro sencillo a través de Facebook.
- Perfiles de jugadores con estadísticas de juego.
- Inscripción online a competiciones: torneos, ligas, circuitos.
- Consulta en tiempo real de cuadros de competiciones, horarios, resultados, perfiles de rivales, fotos e imágenes de los partidos.
- Creación de un Ranking de jugadores amateur.
- Búsqueda de clubes con toda la información de los mismos.

Posee además una aplicación para Android y para Iphone.

4.4.Conclusiones del estudio de herramientas

Una vez analizadas las diferentes herramientas disponibles en el mercado, existe una alternativa que se podría asemejar a nuestra solución, como es Xporty. Parece una herramienta muy completa para la organización y gestión de torneos de diferentes deportes.

No existen numerosas herramientas que den respuesta a los objetivos del proyecto, y la única que parece que puede adecuarse, lo hace con un precio que en algunas ocasiones no es asumible para organizadores de torneos pequeños.

5.Metodología

En este apartado se explica la metodología empleada para el desarrollo del proyecto

5.1.Modelo de proceso de software

Un proceso de software se puede caracterizar como un marco común del proceso definiendo un pequeño número de actividades de trabajo que son aplicables a todos los proyectos software, con independencia de su tamaño o complejidad.

La metodología utilizada en el desarrollo del software de este proyecto parte del modelo de proceso "construcción de prototipos" que se muestra en la ilustración 1. Esta técnica pertenece a los modelos de desarrollo evolutivo y, por tanto, en momentos del desarrollo se ha podido salir eventualmente del modelo propuesto para seguir modelos alternativos evolutivos. A continuación, se explica y se justifica la elección de estas metodologías.

El paradigma de construcción de prototipos, que podemos ver en la ilustración 1, comienza con la recolección de requisitos. El desarrollador y el cliente se encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es necesaria más definición. Entonces, aparece un "diseño rápido". El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente (por ejemplo: enfoques de entrada y formatos de salida). El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

Lo ideal sería que el prototipo sirva como un mecanismo para identificar los requisitos del software.

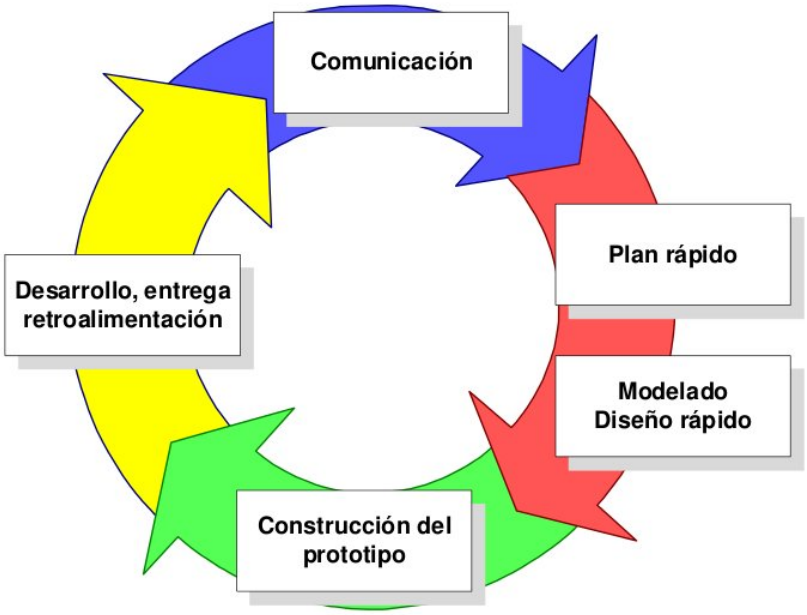


Ilustración 1. Paradigma de construcción de prototipos

La construcción de prototipos nos aporta ciertas ventajas al tipo de proyecto que se está desarrollando:

- El cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida.
- También ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina.

Aunque pueden surgir problemas, la construcción de prototipos ha demostrado ser un paradigma efectivo para la ingeniería del software. La clave es definir las reglas al comienzo, es decir, cliente y desarrollador se deben poner de acuerdo en las fases a seguir. La ilustración 2 muestra un ciclo normal para este paradigma.

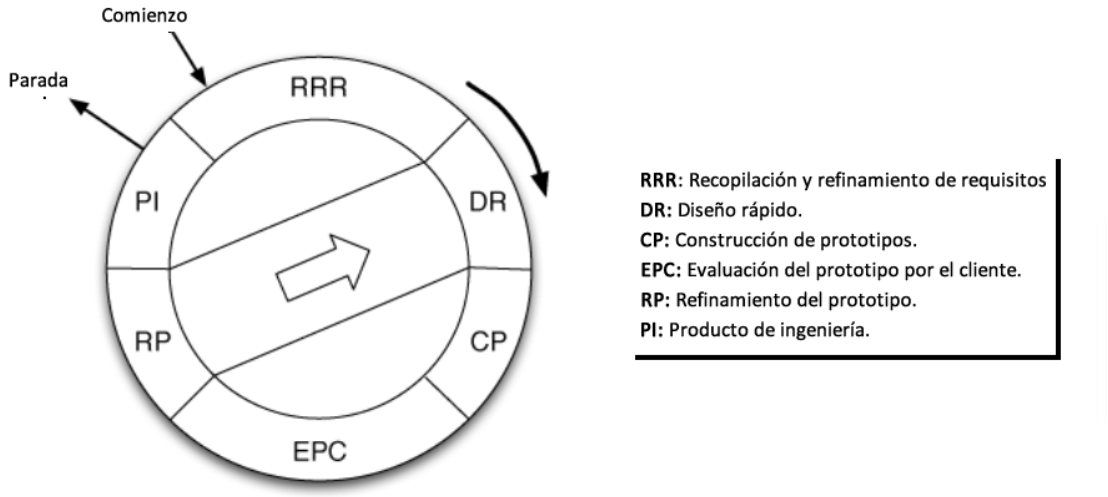


Ilustración 2. Ciclo del paradigma de construcción de prototipos

Dentro de cada fase se recogen los siguientes aspectos:

- Recolección y refinamiento de requisitos: requerimientos del usuario y software.
- Diseño rápido: usa un modelo de desarrollo lineal y secuencial.
- Construcción de prototipos: se implementa un prototipo y se hacen baterías de pruebas para comprobar el funcionamiento.
- Evaluación del prototipo por el cliente: el cliente comprueba la evolución del proyecto para asegurarse que toma el camino adecuado.
- Refinamiento del prototipo: con la información obtenida en las reuniones con el cliente realizamos los cambios pertinentes.
- Producto de ingeniería: es el producto final, resultado del primer prototipo, del segundo, y así sucesivamente.

El modelo en espiral, propuesto originalmente por Boehm [BOE88], es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.

La ilustración 3 muestra un modelo en espiral que contiene seis regiones de tareas:

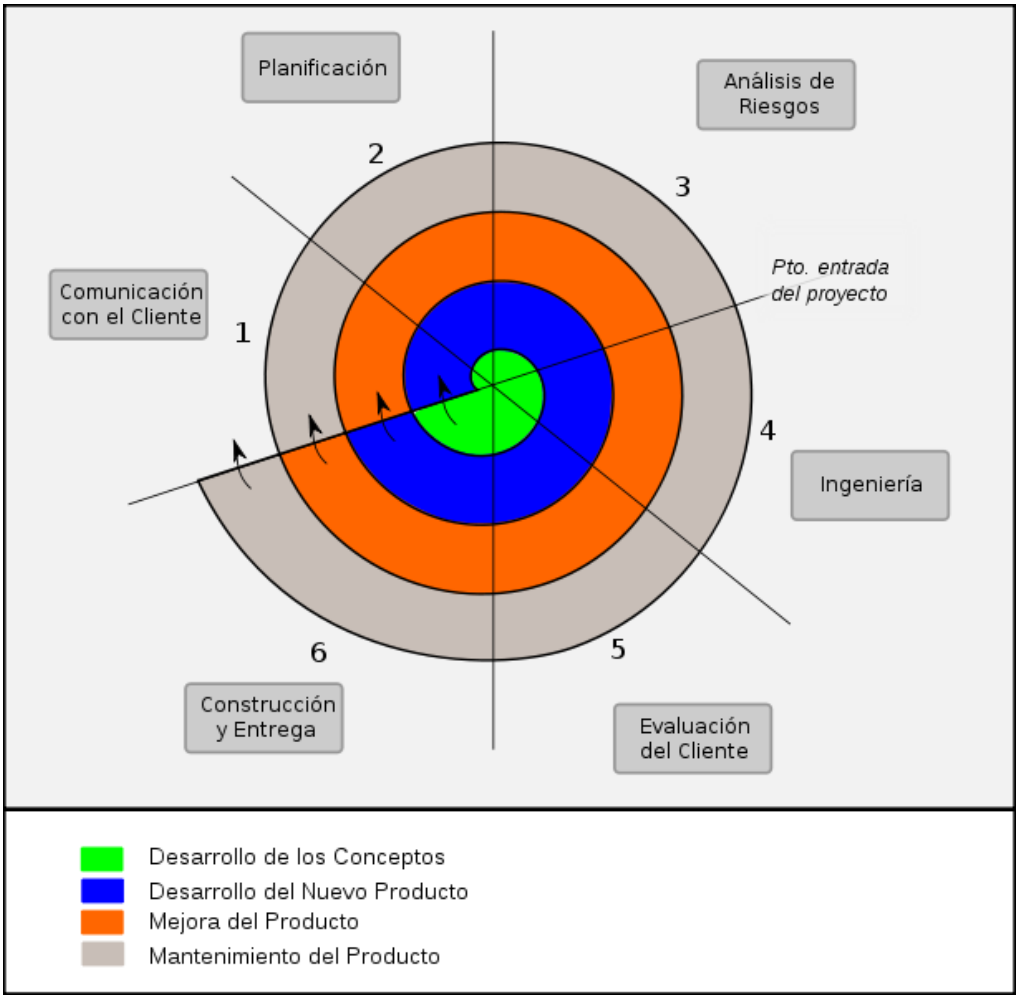


Ilustración 3. Modelo en espiral

- Comunicación con el cliente: se trata de las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
- Planificación: las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.
- Análisis de riesgos: evaluación de riesgos técnicos y de gestión.
- Ingeniería: las tareas requeridas para construir una o más representaciones de la aplicación.
- Construcción y acción: las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario, como documentación y práctica.
- Evaluación del cliente: las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

El modelo en espiral utiliza la construcción de prototipos como mecanismo de reducción de riesgos, pero, lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del proyecto. Mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, pero lo incorpora al marco de trabajo iterativo que refleja de forma más realista el mundo real [PRE03].

El modelo en espiral es un enfoque realista del desarrollo de sistemas y de software a gran escala. Como el software evoluciona, a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor antes los riesgos en cada uno de los niveles evolutivos.

5.2.Lenguaje de modelado

El lenguaje de modelado utilizado a lo largo del proyecto será UML o Lenguaje Unificado de Modelado. Permite expresar un modelo de análisis utilizando una notación de modelado con unas reglas sintácticas, semánticas y prácticas.

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, además de aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Se emplea para definir un sistema, para detallar los artefactos en el mismo, para documentar y construir.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, sólo se diagrama la realidad de una utilización en un requerimiento. La programación estructurada es una forma de programar como lo es la orientación a objetos, sin embargo, la programación orientada a objetos es un complemento de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

5.3.Metodología aplicada

El proyecto descrito en este documento se desarrollará en base a los modelos de proceso de software y el lenguaje de modelo indicado en los apartados precedentes.

Para analizar y diseñar el sistema, se utilizarán los diagramas estándar establecidos por el Lenguaje Unificado de Modelado (UML). Entre estos diagramas destacan la identificación de actores, casos de uso, diagramas de clase y de secuencia.

El desarrollo del software se llevará a cabo utilizando un Modelo de Proceso en Espiral apoyado en prototipos, es decir, éstos se desarrollarán como método de validación después de cada una de las fases descritas en el modelo, especialmente en análisis, diseño e implementación.

5.4.Planificación

La planificación que se realizó al comienzo del proyecto se muestra en la siguiente tabla:

Fase	Horas
Análisis de requisitos	120
Análisis	200
Diseño e Implementación	490
Pruebas	135

TOTAL	900

Se puede observar que en dicha planificación se estimó que aproximadamente un 35% del tiempo se emplearía en el análisis, más de la mitad en el desarrollo y aproximadamente un 15% en las pruebas.

Sin embargo, el compatibilizar el desarrollo del proyecto con la jornada laboral del autor del trabajo, ha complicado la medición exacta de las horas dedicadas a cada una de las fases. A pesar de ello podemos afirmar que, en general, la estimación realizada se ha ajustado bastante a la realidad con dos salvedades: el tiempo dedicado a la fase de análisis ha sido superior al indicado mientras que el tiempo dedicado al desarrollo, debido a la experiencia del autor con el *framework* de desarrollo, ha sido menor.

6.Recursos necesarios

6.1.Recursos hardware

6.1.1.Servidor web

Un servidor web no es más que un programa que se ejecuta de forma continua en un ordenador (también se utiliza el término para referirse al ordenador que lo ejecuta), manteniéndose a la espera de peticiones por parte de un cliente (un navegador de internet) y que contesta a estas peticiones de forma adecuada.

Al tratarse de un servicio basado en Web, será primordial el tiempo de respuesta de la máquina a la hora de responder a las múltiples peticiones por parte de los usuarios del sistema. El servidor indicado deberá poseer al menos las siguientes características:

- Sistema de virtualización, deseable VirtualBox 5.1.14+.
- Acceso a internet.

6.2.Recursos software

6.2.1. Servicio Hypnotoad

El servidor Hypnotoad es un servidor HTTP y WebSocket, preparado para la ejecución multihilos no bloqueantes, optimizado para equipos UNIX. Es el servidor de viene integrado en mojolicious y optimizado para Perl.

6.2.2. Servicio PostgreSQL 9 o superior

Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo licencia PostgreSQL, similar a las BSD o la MIT.

6.2.3.Mojolicious

Framework web en tiempo real basado en PERL que permite desarrollar aplicaciones web con estructura MVC de una manera sencilla.

6.2.4.Putty

Cliente Telnet SSH. Es el cliente para la conexión desde Windows al servidor donde está alojada la aplicación.

6.2.5.Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o front end como Cogito o StGIT. 3 Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena.

6.2.6.Vim

Vi es un programa que entra en la categoría de los editores de texto, pues a diferencia de un procesador de texto no ofrece herramientas para determinar visualmente cómo quedará el documento impreso. Por esto carece de opciones como centrado o justificación de párrafos, pero permite mover, copiar, eliminar o insertar caracteres con mucha versatilidad. Este tipo de programas es frecuentemente utilizado por programadores para escribir código fuente de software.

Vim es una versión mejorada del editor de texto vi, presente en todos los sistemas UNIX.

Su autor, Bram Moolenaar, presentó la primera versión en 1991, fecha desde la que ha experimentado muchas mejoras. La principal característica tanto de Vim como de Vi consiste en que disponen de diferentes modos entre los que se alterna para realizar ciertas operaciones, lo que los diferencia de la mayoría de editores comunes, que tienen un solo modo en el que se introducen las órdenes mediante combinaciones de teclas o interfaces gráficas.

6.2.7.Psql

Es un programa de línea de órdenes para la administración de la base de datos PostgreSQL. Permite introducir queries SQL directamente, o ejecutarlas desde un fichero. Posee una serie de metaórdenes y varias características tipo Shell para facilitar escribir scripts y automatizar tareas.

6.2.8.pgAdmin

PgAdmin es una herramienta multilenguaje de administración PostgreSQL con interfaz gráfica. PgAdmin está escrito en C++ y puede ser ejecutado en los principales sistemas operativos. La herramienta incluye un lenguaje llamado pgScript para soporte administrativo y herramientas de desarrollo.

6.2.9.Sistema operativo Windows 10

Debido a la naturaleza del proyecto – aplicación web – se puede optar por cualquier sistema operativo. Sin embargo, debido a las aplicaciones y editores que se van a utilizar en el desarrollo se ha decidido utilizar un sistema operativo Windows, concretamente Windows 10 Professional.

Windows 10 es la versión más reciente de Microsoft Windows, línea de sistemas operativos producida por Microsoft Corporation. Esta versión está diseñada para uso en PC, incluyendo equipos de escritorio en hogares y oficinas, equipos portátiles, tablet PC,

netbooks y equipos media center. El desarrollo de Windows 10 se completó el 29 de julio de 2015.

Uno de los aspectos más importantes de Windows 10 es el enfoque en la armonización de experiencias de usuario y funcionalidad entre diferentes tipos de dispositivos, además de abordar las deficiencias en la interfaz de usuario de Windows que se introdujo por primera vez en Windows 8.

6.2.10.Microsoft Word 2010

Procesador de texto para la escritura de la memoria. Para la realización de la documentación se ha preferido el Microsoft Word 2010 en detrimento del procesador Latex, ya que éste se hace especialmente eficiente en la creación de libros técnicos donde abunden las expresiones matemáticas. No es el caso que nos ocupa y por tanto se ha seleccionado la opción de Microsoft.

Microsoft Office 2010 es una versión de la suite ofimática Microsoft Office de Microsoft y sucesora de Microsoft Office 2007. Office 2010 incluye compatibilidad extendida para diversos formatos de archivos, actualizaciones de la interfaz de usuario y una experiencia de usuario refinada. Por primera vez y con la introducción de Office 2010, la suite está disponible en una compilación para arquitecturas de 64 bits.

6.2.11.StarUML

La gran mayoría de los diagramas elaborados a lo largo de las diferentes fases del proyecto serán elaborados en el Lenguaje Unificado de Modelado UML, por lo que será necesario utilizar alguna herramienta de edición que permita trabajar de forma sencilla y rápida con este lenguaje.

En concreto, se ha utilizado el software StarUML. Se trata de una herramienta de software libre de edición UML, bajo licencia modificada de GNU GPL. StarUML es compatible con la mayoría de los tipos especificados en el diagrama de UML 2.0 y fue desarrollada en Delphi.

6.3.Tecnologías utilizadas

6.3.1.Perl

Perl es un lenguaje de programación diseñado por Larry Wall en 1987. Perl toma características del lenguaje C, del lenguaje interpretado bourne shell (sh), AWK, sed, Lisp y, en un grado inferior, de muchos otros lenguajes de programación.

Estructuralmente, Perl está basado en un estilo de bloques como los del C o AWK, y fue ampliamente adoptado por su destreza en el procesado de texto y no tener ninguna de las limitaciones de los otros lenguajes de script.

Perl es un lenguaje imperativo, con variables, expresiones, asignaciones, bloques de código delimitados por llaves, estructuras de control y subrutinas.

6.3.2.PostgreSQL

Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo licencia PostgreSQL, similar a las BSD (*Berkeley Software Distribution*) o la MIT (*Massachusetts Institute of Technology*).

Sus principales características son: alta concurrencia, amplia variedad de tipos nativos, posibilidad de crear tipos de datos propios, claves ajenas, triggers, vistas, integridad transaccional, herencia de tablas, tipos de datos y operaciones geométricas, soporte para transacciones distribuidas, funciones.

Sus principales ventajas son: seguridad en términos generales, integridad en Base de datos: restricciones en el dominio, integridad referencial, afirmaciones (Assertions), disparadores (Triggers), autorizaciones, conexión a DBMS (*DataBase Management System*), transacciones y respaldos.

6.3.3.HTML

HTML (*HyperText Markup Language*) es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido de un sitio web en forma de texto, así como para complementar el texto con objetos tales como imágenes.

Además, puede describir la apariencia de un documento e incluir scripts como Javascript o elementos de maquetado como CSS.

Este lenguaje será el utilizado para presentar los contenidos y herramientas al usuario en cada uno de los apartados que componen el sistema desarrollado.

6.3.4.JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (*client-side*).

En la fase de análisis y diseño, así como en los objetivos planteados, se ha recalcado la importancia de una interfaz tecnológicamente puntera que facilite al usuario trabajar con gran cantidad de datos en pantalla. Por tanto, es necesaria la utilización de un lenguaje de programación que se ejecutará en el cliente para que éstos tengan la responsabilidad de gestionar su propia interfaz.

Por esta razón, se ha utilizado JavaScript para modificar "en caliente" el aspecto de la interfaz de usuario.

6.3.5.jQuery

jQuery es una biblioteca o framework de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Se han utilizado algunas librerías Jquery que han permitido conseguir grandes efectos (incluidos algunos fundamentales en la consecución de los objetivos) y que serán comentadas debidamente en el apartado de desarrollo de la memoria.

6.3.6.CSS

CSS, del inglés *Cascading Style Sheets*, es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML, XML o XHTML. La idea principal que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

De esta manera, se ha utilizado CSS para dar formato a cada uno de los documentos HTML desarrollados a través de sus etiquetas. Mediante esta tecnología, se facilita la presentación de contenidos y se ofrece una interfaz de usuario más accesible y clara.

6.3.7.AJAX

AJAX, acrónimo de *Asynchronous JavaScript And XML*, es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

En la aplicación que se ha desarrollado, se presenta en numerosas ocasiones la posibilidad de mostrar contenidos al usuario sin la necesidad de recargar la página al completo. De esta manera, se ha utilizado la tecnología AJAX para ofrecer servicios más interactivos y presentar información al usuario sin interrupciones.

6.3.8.Bootstrap

Bootstrap es un framework o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

Bootstrap tiene un soporte relativamente completo para HTML5 y CSS 3, y es compatible con la mayoría de los navegadores web. La información básica de compatibilidad de sitios web o aplicaciones está disponible para todos los dispositivos y navegadores. Existe un concepto de compatibilidad parcial que hace disponible la información básica de un sitio web para todos los dispositivos y navegadores. Por ejemplo, las propiedades introducidas en CSS3 para las esquinas redondeadas, gradientes y sombras son usadas por Bootstrap a pesar de la falta de soporte de navegadores antiguos. Esto extiende la funcionalidad de la herramienta, pero no es requerida para su uso.

Desde la versión 2.0 también soporta diseños sensibles. Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado (computadoras, tabletas, teléfonos móviles).

Bootstrap es de código abierto y está disponible en GitHub.

6.3.9.DataTables

DataTables es un librería de la biblioteca jQuery – JavaScript. Es una herramienta muy flexible para el control de tablas HTML. De todas las características que ofrece, se destacan las siguientes:

- Paginación de longitud variable.
- Ordenación por múltiples columnas.
- Manejo inteligente de los anchos de columnas.
- Totalmente personalizable.
- Buscador dinámico.

6.3.10.ApiRest

ApiRest es un estilo de arquitectura del software para sistemas WWW. Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura —descritos más abajo—, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP. Es posible diseñar sistemas de servicios web de acuerdo con el estilo arquitectónico REST de Fielding y también es posible diseñar interfaces XMLHTTP de acuerdo con el estilo de llamada a procedimiento remoto (RPC), pero sin usar SOAP. Estos dos usos diferentes del término REST causan cierta confusión en las discusiones técnicas, aunque RPC no es un ejemplo de REST.

Los sistemas que siguen los principios REST se llaman con frecuencia RESTful. REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST)
- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones CRUD en bases de datos (CLAB en castellano: crear, leer, actualizar, borrar) que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.
- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.
- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

6.3.11.Curl

cURL es un proyecto de software consistente en una biblioteca (libcurl) y un intérprete de órdenes (curl) orientado a la transferencia de archivos. Soporta los protocolos FTP, FTPS, HTTP, HTTPS, TFTP, SCP, SFTP, Telnet, DICT, FILE y LDAP, entre otros. La primera versión se publicó en 1997.

cURL soporta certificados HTTPS, HTTP POST, HTTP PUT, subidas FTP, Kerberos, subidas mediante formulario HTTP, proxies, cookies, autenticación mediante usuario y contraseña (Basic, DIgest, NTLM y Negotiate para HTTP y kerberos 4 para FTP), continuación de transferencia de archivos, tunneling de proxy HTTP y otras prestaciones. cURL es Open Source, software libre distribuido bajo la Licencia MIT.

El principal propósito y uso para cURL es automatizar transferencias de archivos o secuencias de operaciones no supervisadas. Es por ejemplo, una herramienta válida para simular las acciones de usuarios en un navegador web.

6.3.12.CPAN

CPAN es el acrónimo de *Comprehensive Perl Archive Network*. Es un enorme archivo de software escrito en Perl, así como de documentación sobre el mismo. Tiene presencia en la Web a través de su sitio www.cpan.org y sus 267 espejos distribuidos por todo el mundo.

Actualmente hay más de 119 118 módulos empaquetados en más de 27 018 paquetes, aportados por más de 10 467 autores.

Se dispone de módulos para una amplia variedad de tareas, incluyendo matemáticas avanzadas, conectividad de bases de datos y conexión de redes. Esencialmente, todo lo que hay en CPAN está disponible de forma libre; La mayor parte del software está licenciado bajo la Licencia Artística, la GPL o ambas. Cualquiera puede subir software a CPAN vía PAUSE, el *Perl Authors Upload Server* (servidor de subidas de autores Perl).

Los módulos en CPAN pueden ser descargados e instalados a mano. Sin embargo, es muy común que los módulos dependan de otros módulos y seguir las dependencias a mano puede ser tedioso. Tanto el módulo CPAN.pm (incluido en la distribución Perl) como el módulo mejorado CPANPLUS ofrecen instaladores en línea de comandos que entienden las dependencias; pueden configurarse para descargar automáticamente e instalar un módulo y, recursivamente, todos los módulos de los que dependa.

7.Análisis

7.1.Introducción

Tras realizar el estudio de las aplicaciones de gestión de torneos de pádel, se puede tener una idea de la línea de trabajo necesaria para desarrollar el proyecto:

- Interfaz web.
- Heurísticas de cálculo de horarios.
- Sistema de comunicación entre los distintos usuarios.

No obstante, la fase de análisis va a permitir identificar y definir formalmente los requerimientos de la aplicación antes de pasar a su diseño e implementación.

Dichos requerimientos son, según Pressman, la descripción de los servicios proporcionados por el sistema que se pretende desarrollar y sus restricciones operativas.

7.2.Requisitos del software

7.2.1.Introducción

Roger S. Pressman expresa que para que un esfuerzo de desarrollo de software tenga éxito, es esencial comprender perfectamente los requisitos del software.

Independientemente de lo bien diseñado o codificado que esté un programa, si se ha analizado y especificado pobremente, decepcionará al usuario y desprestigiará al que lo ha desarrollado. La parte más difícil en la construcción de sistemas software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan ardua como establecer los requisitos técnicos detallados, incluyendo todas las interfaces con humanos, máquinas y otros sistemas. Ninguna otra parte del trabajo puede perjudicar tanto el resultado final si se realiza de forma errónea. Ninguna otra parte es tan difícil de rectificar posteriormente.

Por todo lo explicado en el párrafo anterior, se ha puesto especial interés en esta sección, pues una documentación bien estructurada y detallada puede facilitar tanto el trabajo que se pretende desarrollar actualmente como el futuro.

7.2.2.Modelo conceptual

Aquí se muestran las distintas funcionalidades que se pueden encontrar en un sistema de generación y gestión de campeonatos y calendarios de pádel. El número de funcionalidades que se pueden encontrar se verá en aumento dependiendo de la funcionalidad que le desee dar al software nuestro cliente. Hay que destacar que la funcionalidad del sistema no se basa únicamente en la gestión de los torneos. El sistema está provisto de ciertas acciones basadas en eventos. Así se enviará un aviso a los jugadores recordándole partidos, informando fecha límite de cambios o informando de incidencias en el torneo.

Anónimo

Se entiende por anónimo a todo usuario del sistema que no esté registrado. Sus funcionalidades son:

- Ver noticias relacionadas con campeonatos.
- Listar campeonatos y filtrarlos.
- Ver resultados por jornadas de un campeonato.
- Ver calendario de un campeonato.
- Registrarse en el sistema.
- Apuntarse a un campeonato.

Usuario Registrado

Se entiende por usuario registrado a cualquier usuario que se ha dado de alta en el sistema. Son jugadores que desean apuntarse en un campeonato, o ya se han apuntado. Sus funcionalidades son todas las del usuario anónimo más:

- Ver los campeonatos en los que está participando.
- Generar incidencias con el administrador.
- Insertar y editar sus preferencias de horario.
- Darse de baja de un campeonato.
- Gestionar cambios en el calendario.
- Introducir resultados.
- Validar resultados.

Administrador

Se entiende por administrador a todo usuario con capacidad de crear campeonatos. Posee todas las características del usuario anónimo más:

- Crear campeonatos.
- Dar de alta clubes.
- Insertar recursos (pistas).
- Generar horarios.

- Modificar horarios.
- Gestionar incidencias.
- Administrar los resultados de la jornada.
- Generar demanda de pistas de pádel.

7.2.3.Identificación de actores

Definición de los actores identificados en el sistema. En la tabla 1 aparecen los distintos actores, el tipo al que pertenecen y su definición.

Actor/rol	Tipo	Definición
Administrador	Avanzado	Tareas enfocadas a la generación y supervisión de los campeonatos y jornadas
Usuario registrado	Avanzado	Jugador de pádel que desea apuntarse a uno o varios campeonatos
Anónimo	Básico	Usuario que desea introducirse en el mundo del pádel o aficionado al mismo como espectador

Tabla 1. Identificación de actores

En la tabla 1 se especifican los distintos roles que un usuario del sistema puede ejercer.

Por lo tanto, a la hora de crear un Usuario o en la modificación de los datos del mismo, se podrá especificar que roles podrá realizar. Un usuario al ejercer un rol podrá realizar todas los tareas definidas para este. Además pueden existir usuarios ejerciendo varios roles a la vez. Dentro de los roles, existe una cierta jerarquía donde al ejercer un rol, se ejercen automáticamente otros, según se muestra en la tabla 2.

Rol	Roles incluidos
Anónimo	Ninguno
Usuario registrado	Anónimo
Administrador	Anónimo

Tabla 2. Tabla de jerarquías

7.2.4.Diccionario de conceptos

Pádel: Deporte de raqueta que se juega normalmente por parejas en una pista diseñada para su práctica.

Club: Instalación con 1 o más pistas para la práctica de pádel.

Pista: Área de juego rectangular de diez metros de ancho por veinte de largo y cerrada en su totalidad con una red que la divide en dos.

Jugador: Cada uno de los participantes en un partido de pádel.

Equipo: El pádel se juega normalmente por parejas, aunque existen modalidades individuales. En algunos torneos o ligas se permite que el equipo esté compuesto por más de dos jugadores, siendo seleccionados dos de ellos para cada partido. No se admiten cambios durante un partido de Pádel.

Disponibilidad: Se entiende por disponibilidad los momentos en el que cada uno de los jugadores puede participar en un partido de pádel. Este mismo concepto se aplica a la oferta de pistas de un club.

Competición: En el pádel existen diferentes modalidades de juego: americano, torneo por parejas, torneo individual, liga, etc.

Liga: Modalidad de competición en la que todos los equipos se enfrentan entre sí y en la que el equipo con más victorias es el ganador. La liga puede realizarse a varias vueltas y se puede añadir un sistema de playoffs al finalizar.

Torneo: Sistema de competición en la que el equipo perdedor queda eliminado, realizando cruces entre los ganadores.

Americano: sistema de competición en la que se suman los puntos conseguidos por cada participante o equipo en cada partido. El ganador es aquel que haya conseguido ganar más juegos durante la competición.

7.2.5. Actores/Objetivos

En la tabla 3 se muestran los actores y sus objetivos en una lista actor-objetivo.

Actor/rol	Objetivos	Breve resumen
Anónimo	Ver noticias relacionadas con campeonatos	Ve un listado de noticias genéricas sobre la aplicación, campeonatos, etc.
	Listar campeonatos y filtrarlos	Ve los campeonatos que han sido creados en la aplicación. Puede filtrarlos según el estado, ubicación, categoría, etc.
	Ver resultados por jornadas de un campeonato	Sobre un campeonato específico, muestra la clasificación y resultados.
	Ver calendario de un campeonato	Muestra las jornadas de un campeonato.
	Registrarse en el sistema	Permite registrarse en el sistema con todos sus datos personales y extras como horario preferido y categoría.
	Apuntarse a un campeonato	En algunos campeonatos se permite apuntarse directamente sin haberse registrado previamente, especificando los datos mínimos para ese campeonato.
	Darse de baja de un campeonato	Antes del comienzo de un campeonato el usuario registrado puede abandonarlo.
	Gestionar cambios en el calendario	Sobre un calendario propuesto y previo a la fecha límite de cambios el usuario registrado puede realizar cambios y propuestas a la organización y a la otra pareja.
	Introducir resultados	Una vez terminado el partido cualquier usuario que haya jugado el partido puede introducir el resultado.
	Validar resultados	Una vez validado el resultado por la otra

		pareja el resultado se da por válido. Se dará por válido igualmente una vez finalizada la fecha de introducción de resultados.
Usuario registrado	Ver los campeonatos en los que está participando	Muestra un listado de campeonatos en los que el usuario participa.
	Gestionar incidencias con el administrador	Permite insertar incidencias y la respuesta a ésta.
	Insertar y editar sus preferencia de horario	Permite gestionar su disponibilidad.
Administrador	Dar de alta clubes	El administrado puede dar de alta clubes de juego informando de su localización. Sobre los clubes se insertarán las pistas disponibles en cada uno.
	Crear campeonatos	Permite la creación de un campeonato, con sus características correspondientes.
	Insertar recursos (pistas)	Permite insertar el número de pistas de un club.
	Generar horarios	Realiza la generación de horarios del campeonato.
	Modificar horarios	Permite modificar los horarios una vez generado por el sistema.
	Gestionar incidencias	Realiza la gestión de incidencias de los partidos.
	Introducir los resultados de la jornada	Introduce los resultados de los partidos.

Tabla 3. Lista actor-objetivo

7.2.6. Listado resumen de los casos de uso

La tabla 4 muestra un resumen de los casos de uso de cada rol del sistema, con el identificador correspondiente.

Actor Principal	Casos de uso	Nº CU
Anónimo	Ver noticias relacionadas con campeonatos	19
	Listar campeonatos y filtrarlos	13
	Ver resultados por jornadas de un campeonato	20
	Ver calendario de un campeonato	17
	Registrarse en el sistema	15
	Apuntarse a un campeonato	1
Usuario registrado	Ver los campeonatos en los que está participando	18
	Generar incidencias	6
	Insertar y editar sus preferencia de horario	10
	Darse de baja de un campeonato	4
	Gestionar cambios en el calendario	7

	Introducir resultados	12
	Validar resultados	16
Administrador	Dar de alta clubes	3
	Crear campeonatos	2
	Insertar recursos (pistas)	9
	Generar horarios	5
	Modificar horarios	14
	Gestionar incidencias	8
	Introducir los resultados de la jornada	11

7.2.7. Diagramas de casos de uso

Los diagramas de casos de uso proporcionan una visión rápida y global de las relaciones entre los casos de uso. En la presente sección se muestran los diagramas de casos de uso asociados a cada actor del sistema.

Tabla 4. Resumen de los casos de uso

Anónimo

La ilustración 4 muestra los casos de uso del usuario anónimo:

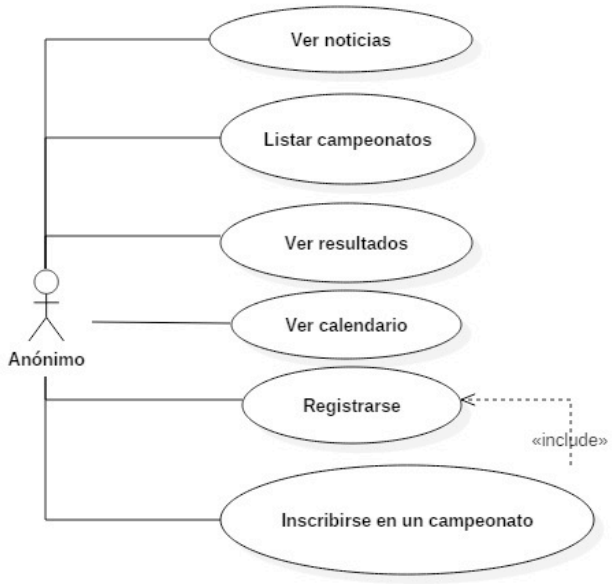


Ilustración 4. Casos de uso del usuario anónimo

Usuario Registrado

La ilustración 5 muestra los casos de uso del usuario registrado:

Ilustración 5. Casos de uso del usuario registrado

Administrador

La ilustración 6 muestra las opciones del sistema para el usuario administrador:

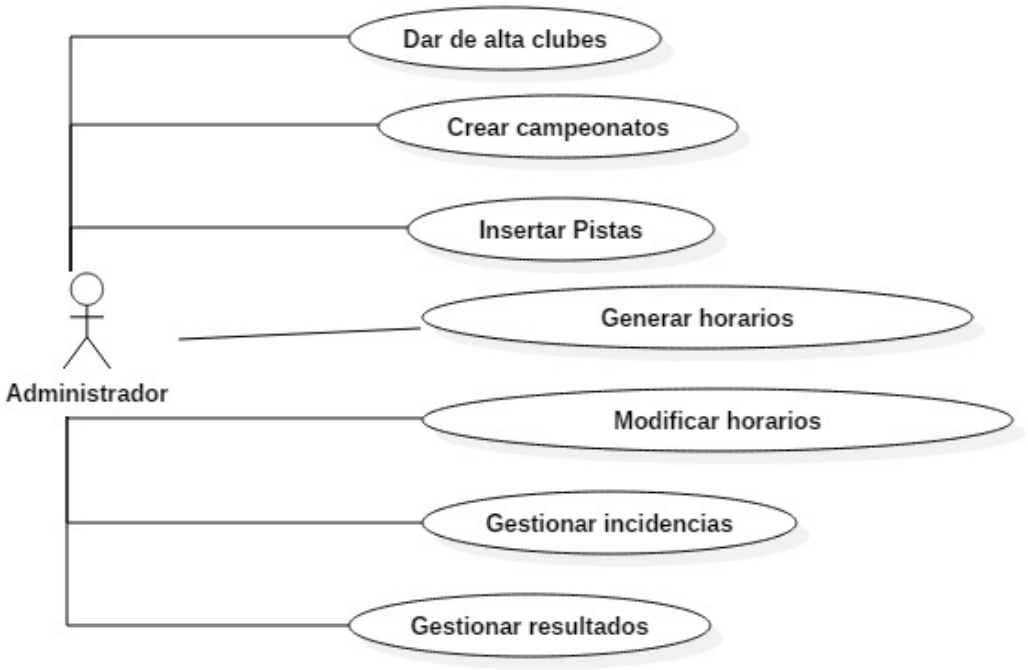


Ilustración 6. Casos de uso del usuario administrador

7.2.8.Casos de uso completos

Ya se han identificado los actores y definido mediante diagramas los casos de uso. Los diagramas dan una representación rápida del sistema, pero no aportan toda la

información sobre el caso de uso. Jacobson [JAC92] sugiere varias preguntas que se deberían contestar mediante un caso de uso. Estas preguntas se han extendido para proporcionar una visión más completa del contenido del caso de uso:

- ¿Quién(es) es (son) el(los) actor(es) primario(s)?
- ¿Cuáles son las metas del actor?
- ¿Cuáles son las condiciones previas que deben existir antes de comenzar la historia?
- ¿Cuáles son las tareas o funciones principales que realiza el actor?
- ¿Qué excepciones podrían considerarse mientras se describe la historia?

Para dar respuesta a todas estas cuestiones relativas a los casos de uso se utilizará una plantilla que se explica a continuación.

Nombre	APUNTARSE A UN CAMPEONATO	ID	01
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	ANÓNIMO
Personal involucrado	Anónimo: Selecciona un campeonato y se apunta a él Administrador: Ve un nuevo jugador apuntado a su campeonato
Descripción	Un nuevo jugador se apunta a un campeonato
Trigger	
Precondición	
Postcondición	El jugador aparece en la lista de inscritos
Flujo Normal	1. Rellena el formulario de los datos personales 2. El jugador se añade a la lista de inscritos
Flujo alternativo	
Excepción	El jugador ya está inscrito Se ha superado el número máximo de inscritos
Includes	
Requisitos especiales	
Notas	

ID

Dar a cada caso de uso un entero secuencial único identificativo. Alternativamente, se puede usar la forma jerárquica X.Y. Casos relacionados pueden agruparse jerárquicamente.

Nombre

Selecciona un nombre que sea lo más explicativo posible. Este ha de reflejar por sí mismo la tarea que el usuario necesita realizar.

Creado por

Identifica a la persona que ha creado el caso de uso.

Fecha

Muestra la fecha de creación del caso de uso.

Modificado por

Identifica a la persona que ha modificado el caso de uso.

Fecha modificación

Muestra la fecha de modificación del caso de uso.

Actor principal

Especifica el actor principal que recurre a los servicios del sistema para cumplir un objetivo.

Personal involucrado o intereses

Esta lista es más importante y práctica de lo que podría parecer a primera vista.
Sugiere y delimita qué es lo que debe hacer el sistema.

Descripción

Especifica una descripción resumida de las razones y el resultado del caso de uso.

Trigger

Identifica al evento que inicializo el caso de uso. Esto puede ser un evento externo o un evento del generado por el propio sistema, también puede ser el primer paso del flujo normal.

Precondiciones

Las precondiciones establecen lo que siempre debe cumplirse antes de comenzar un escenario de caso de uso. Las precondiciones no se prueban en el caso de uso, sino que son condiciones que se asumen que son verdad. Normalmente, una precondición implica un escenario de otro caso de uso que se ha completado con éxito. Por lo tanto, hay que listar cualquier actividad que debe tener lugar, o cualquier condición que debe ser cierta, antes de que el caso de uso pueda comenzar.

Postcondiciones

Las postcondiciones o garantías de éxito establecen qué debe cumplirse cuando el caso de uso se completa con éxito. La garantía debería satisfacer a todo el personal involucrado. Por lo tanto, las postcondiciones describen el estado del sistema tras la conclusión del caso de uso.

Flujo normal

Describe el camino de éxito típico que satisface los intereses del personal involucrado. Provee una descripción detallada de las acciones de usuario y las respuestas del sistema que tendrán lugar durante la ejecución normal del caso de uso. Esta secuencia llevará a la

consecución del caso de uso, alcanzando el objetivo deseado. La descripción se puede escribir como una respuesta a la hipotética pregunta, "¿Cómo hago para conseguir la tarea especificada en el caso de uso en cuestión?" Esto se consigue mejor mediante una lista de acciones realizadas por el actor, alternativamente con las respuestas ofrecidas por el sistema.

Flujo alternativo o extensiones

Las extensiones son muy importantes. Indican todos los otros escenarios o bifurcaciones, tanto de éxito como de fracaso. Por lo tanto, la combinación del flujo normal y del flujo alternativo deberían satisfacer "casi" todos los intereses del personal involucrado (de los usuarios).

Excepciones

Describe cualquier condición de error que pueda ocurrir durante la ejecución del caso de uso, y define como el sistema responde en estas situaciones. También describe como el sistema responde si la ejecución del caso de uso falla por alguna situación no controlada. Se ha de especificar si tras un error de este tipo se ha de realizar una vuelta atrás de las modificaciones que se estaban realizando, si finaliza parcialmente con un estado conocido, o si se deja en un estado indeterminado como resultado de la excepción.

Includes

Lista cualquier otro caso de uso que este incluido ("llamado") por este caso de uso. Si aparece una funcionalidad común en múltiples casos de uso, esta puede convertirse en un caso de uso el cual pueda ser incluido por aquellos casos de uso que necesiten esa funcionalidad.

Requisitos especiales

Si un requisito no funcional, atributo de calidad o restricción se relaciona de manera específica con un caso de uso, se recoge en el caso de uso. Esto incluye cualidades tales como rendimiento, fiabilidad y facilidad de uso, y restricciones de diseño (a menudo, en dispositivos de entrada/salida) que son obligados o se consideran probables.

Notas

Lista cualquier comentario adicional sobre el caso de uso.

Los casos de uso completos se encuentran de forma detallada en el anexo C de éste documento.

7.3.Prototipos de validación

La realización de prototipos ayuda a identificar los requisitos y objetivos globales del software. Además, es la primera representación de la interfaz, lo que facilita la tarea de diseño. Por otra parte, ayuda al desarrollador a entender la interacción hombre-máquina, obteniendo así un mejor enfoque. El prototipo se pone a punto para satisfacer las

necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

Con el fin de validar los actores y casos de uso, se han realizado una serie de prototipos de la aplicación antes de continuar con la siguiente fase del proyecto.

En la ilustración 7 se muestra la gestión de horarios de los jugadores. Es similar en cuanto a su diseño a la gestión de horarios de pistas de pádel.

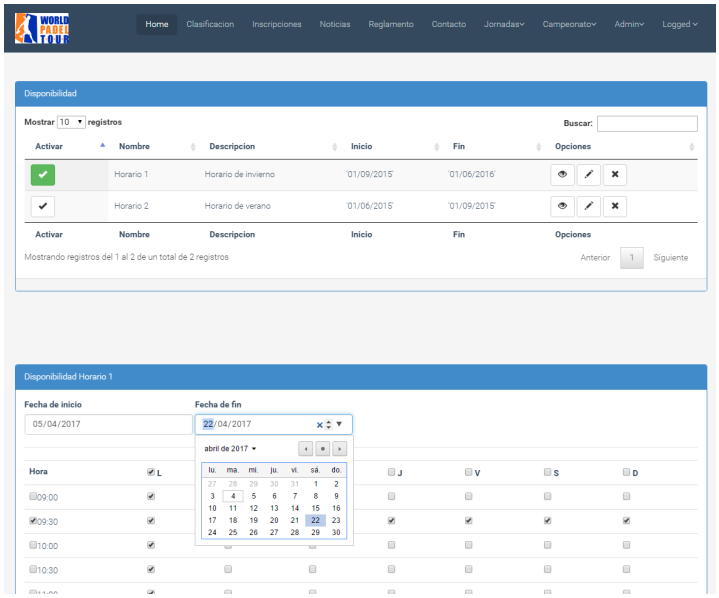


Ilustración 7. Gestión de horarios de los jugadores

La ilustración 8 muestra el listado de jugadores/usuarios registrados. Es la base común para mostrar listas de datos.

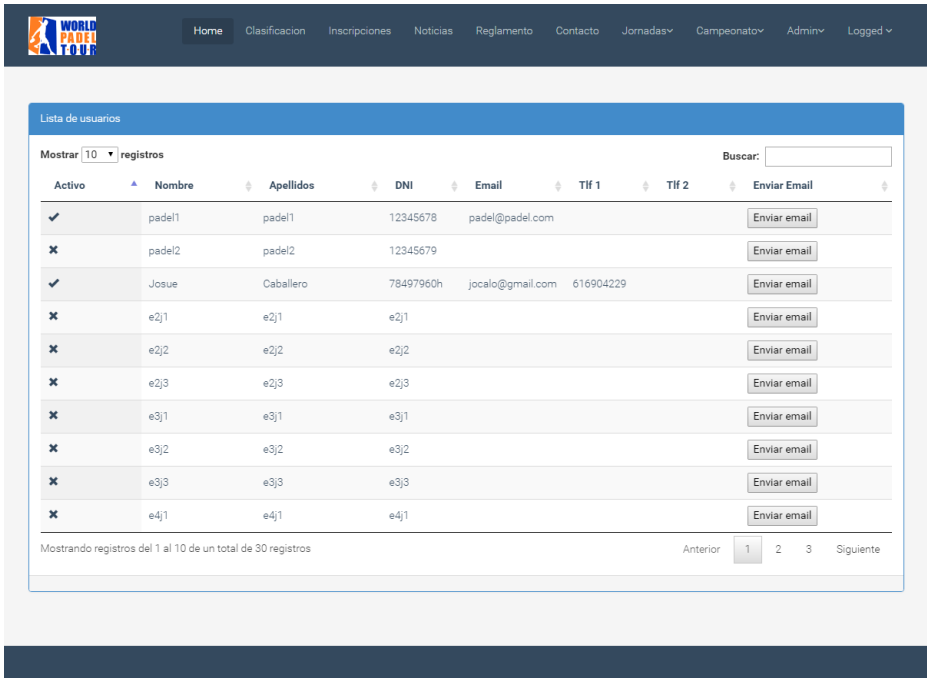


Ilustración 8. Listado de jugadores

La ilustración 9 muestra la configuración de competiciones. Es el formato definido para insertar datos en el sistema mediante formularios.

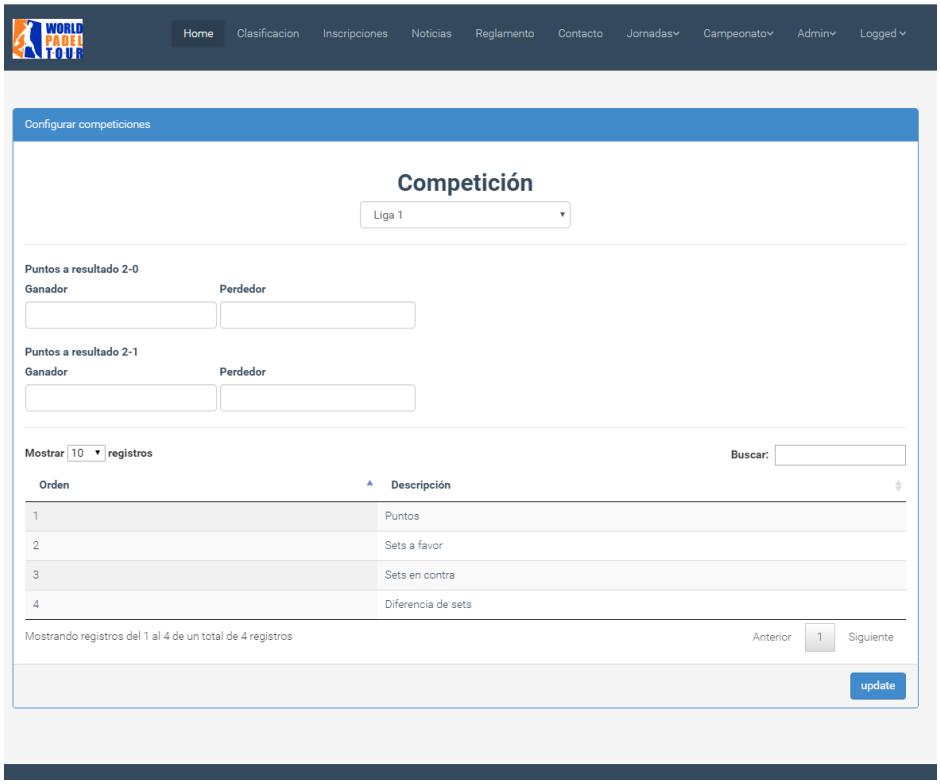


Ilustración 9. Configurador de competiciones

La ilustración 10 muestra el calendario para la asignación de pistas.

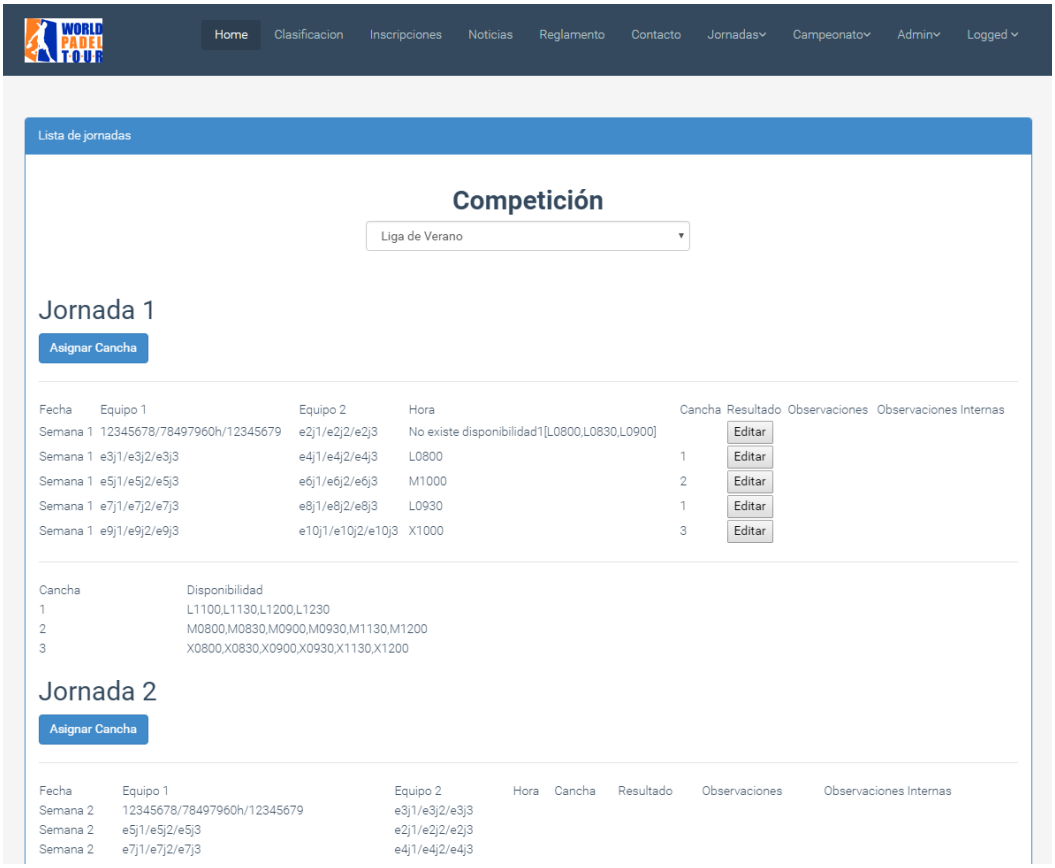


Ilustración 10. Calendario de asignación de pistas

La ilustración 11 muestra el formulario de edición de pistas.

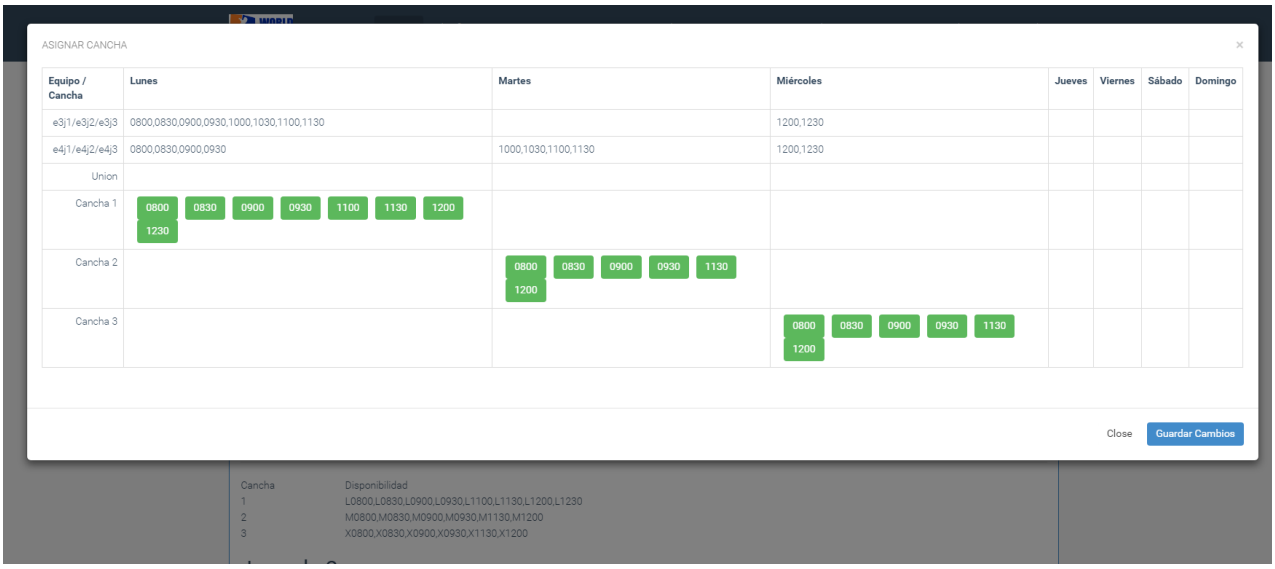


Ilustración 11. Edición de pistas

8.Diseño

8.1.Introducción

El diseño del software se encuentra en el núcleo técnico de la ingeniería del software y se aplica independientemente del modelo de diseño del software que se utilice. Una vez que se analizan y especifican los requisitos del software, el diseño es la primera de las tres actividades técnicas – diseño, generación de código y pruebas – que se requieren para construir y verificar el software. Cada actividad transforma la información de manera que dé lugar, por último, a un software validado. [PRE03].

En general, la actividad del diseño se refiere al establecimiento de las estructuras de datos, la arquitectura general del software, representaciones de la interfaz y algoritmos. Por tanto, el diseño debe contemplar todos los requisitos explícitos obtenidos en la fase de análisis, debe ser una guía que puedan leer y entender los que construyen el código y los que prueban y mantienen el software, debe proporcionar una idea completa de lo que es el software.

El sistema desarrollado es una aplicación tipo web y la arquitectura será cliente-servidor.

Además, para diseñar dicha arquitectura se ha seleccionado el patrón MVC (Modelo, Vista, Controlador).

8.2.Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes.

En este modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios y en el cliente permanece sólo lo particular de cada usuario.

Los clientes realizan funciones como:

- Manejar la interfaz de usuario.
- Capturar y validar datos de entrada.
- Generar consultas e informes sobre la base de datos.

El servidor realiza, entre otras, las siguientes funciones:

- Controlar accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.

Entre las principales características de esta arquitectura se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.

- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

8.3.Patrón MVC (Modelo Vista Controlador)

Modelo Vista Controlador (MVC)- ilustración 12- es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos:

- Modelo: datos.
- Vista: muestra la información del modelo al usuario.
- Controlador: gestiona las entradas del usuario e implementa la lógica de la aplicación.

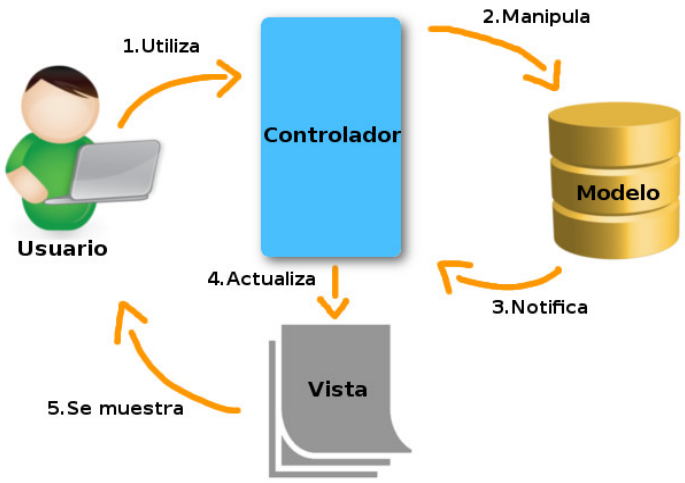


Ilustración 12. Modelo vista controlador

La Vista se presentará en un formato adecuado para que el usuario pueda interactuar con la misma. La interfaz de usuario, principalmente desarrollada en HTML, se encargará de presentar los contenidos de manera clara y accesible, obtenidos del Modelo.

Cuando el usuario interactúa con la Vista haciendo clic en algún botón o realizando alguna acción, el Controlador recibirá la petición solicitada. Tras procesar la acción requerida, éste será el encargado de invocar peticiones al Modelo e incluso a la Vista. El Controlador está desarrollado en Perl, con lo que le permite comunicarse con la Vista y el Modelo, para modificarlos o actualizarlos.

El Modelo es la representación específica de la información con la que opera el sistema. Consiste, principalmente, en las bases de datos en PostgreSQL que almacena toda la información de la aplicación. A ellas accede el Controlador para realizar modificaciones y actualizaciones y la Vista para obtener la información a presentar.

8.4.Diseño arquitectónico

El diseño arquitectónico consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

Para la realización del diseño arquitectónico se ha seguido el patrón Modelo-Vista-Controlador, junto con la arquitectura Cliente-Servidor que nos proporciona Mojolicious, ambas explicadas anteriormente. La ilustración 13 muestra el diseño arquitectónico general del proyecto.

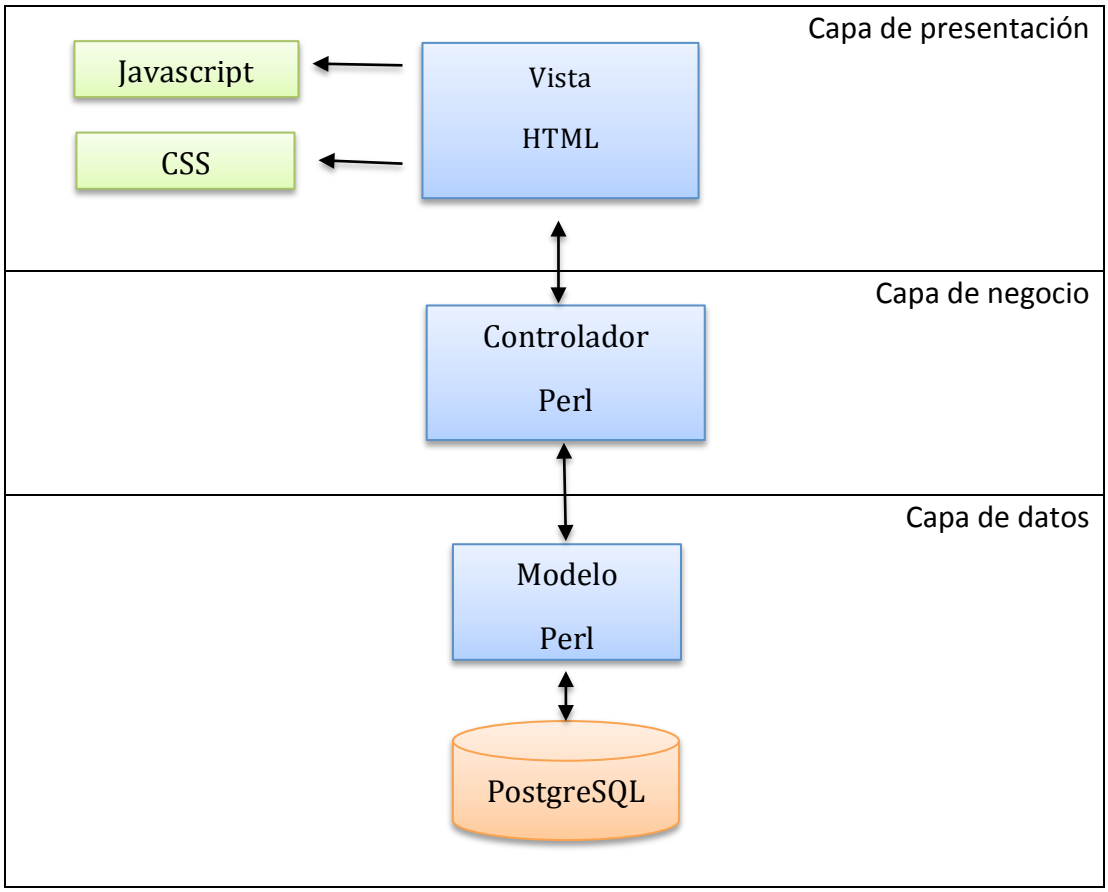


Ilustración 13. Diseño arquitectónico

La capa de presentación representa la interfaz de usuario, a través de la cual se interactúa con la aplicación. Contiene el código HTML de la página y se encuentra enlazado a las librerías JavaScript y a las hojas de estilo (CSS). Esta capa presenta el sistema al usuario, le comunica información y también la captura para poder transmitirla a la capa de negocio.

La capa de negocio es donde residen los programas que se ejecutan. Se reciben peticiones del usuario y se envían respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse.

Esta capa se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados; y con la capa de datos para ejecutar programas y solicitar al gestor de la base de datos almacenar o recuperar información.

La capa de datos es donde residen los datos. Está formada por el módulo Modelo que contiene las clases, métodos y funciones externas al sistema, las cuales son llamadas en la capa de negocio. Además, consta de un gestor de base de datos en PostgreSQL que realiza gestiones como almacenar o recuperar información.

El siguiente diagrama, ilustración 14, muestra un ejemplo de funcionamiento de todo lo explicado en las secciones anteriores para el caso de registro de un usuario.

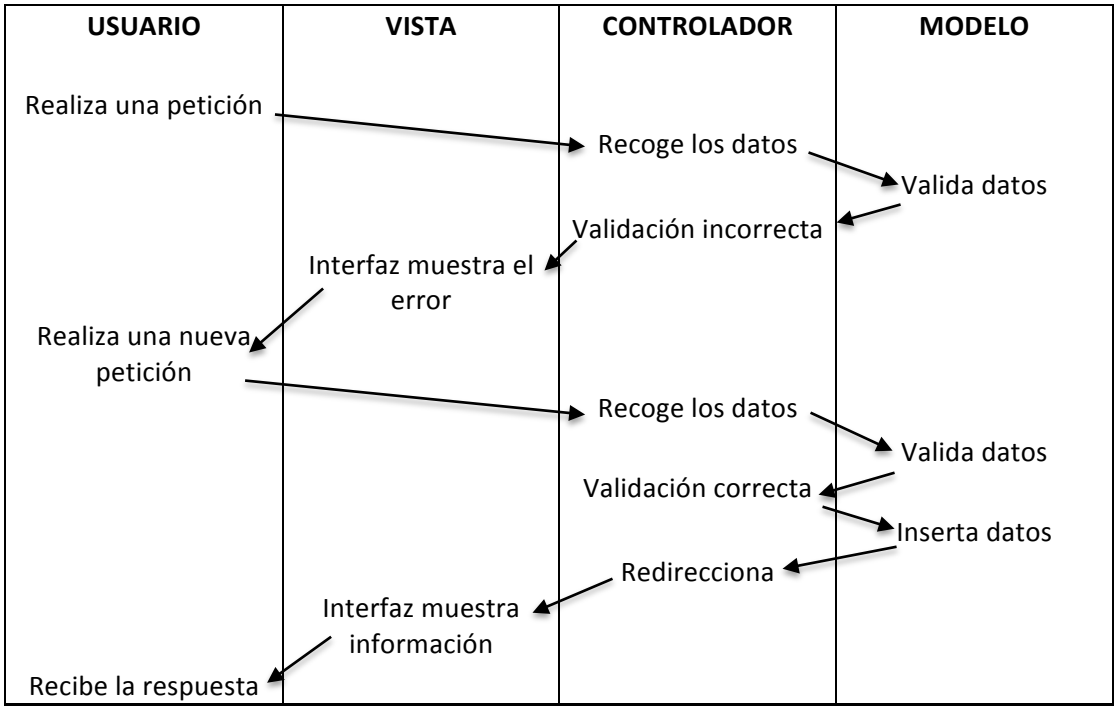


Ilustración 14. Ejemplo de registro de un usuario

8.5.Base de datos

El tratamiento automático de la información en un computador ha experimentado una profunda evolución desde los pioneros sistemas de gestión de ficheros hasta los actuales sistemas de gestión de bases de datos. En la actualidad, las técnicas de bases de datos representan la tecnología informática disponible para la organización y gestión de grandes volúmenes de datos, ocupando un lugar relevante en todas las áreas en las que son utilizados los computadores. [MAT03].

El modelo de datos representa un lenguaje orientado a describir la base de datos. Para realizar el diagrama que se muestra a continuación, se ha utilizado el programa PgAdmin.

La ilustración 15 muestra el esquema general de la base de datos del sistema.

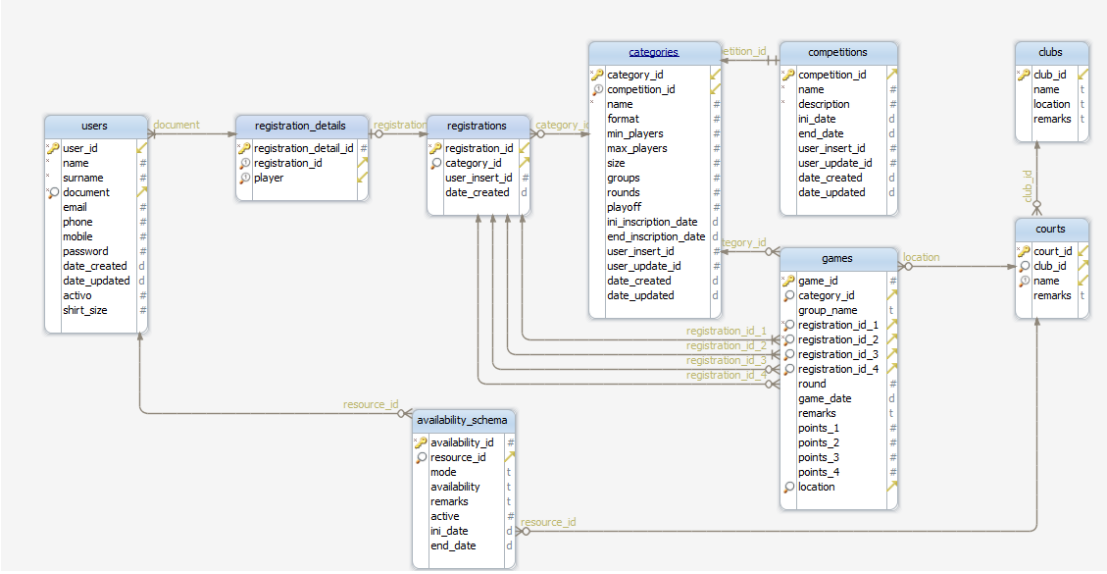


Ilustración 15. Esquema de la base de datos

8.5.1. Tablas

En este apartado se explicará brevemente las características de las tablas más importantes así como las relaciones entre ellas. Se detallarán aquellas tablas que componen el núcleo de la aplicación.

Ilustración 16. Detalle de almacenamiento de equipos

La relación que vemos en la ilustración 16 almacena la información de los equipos de la aplicación. La tabla *registrations* almacena cada uno de los equipos enlazado con la categoría en la que se ha inscrito para jugar. La tabla *registration_details* almacena cada uno de los jugadores que pertenecen a dicho equipo. La tabla *users* almacena toda la información de los jugadores. Cada vez que se crea un equipo, se inserta tantos usuarios en la tabla *users* como jugadores. No todos los registros de la tabla *users* son usuarios registrados.

Un jugador puede estar en varios equipos a la vez, y un equipo puede tener a uno o varios jugadores, que no se pueden repetir. Esta relación de muchos a muchos se gestiona con la tabla *registration_details*, ya que un jugador (*user*) solo puede estar una vez por equipo y puede estar en varios equipos a la vez.

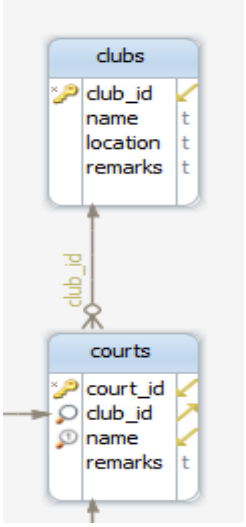


Ilustración 17. Detalle de almacenamiento de pistas

La aplicación gestiona la información de clubes y las pistas disponibles. Cada club tiene una o más pistas disponibles para la competición. Esta información permite al gestor informar al club automáticamente de la necesidad de pistas una vez que los partidos han quedado fijados. El esquema se muestra en la ilustración 17.

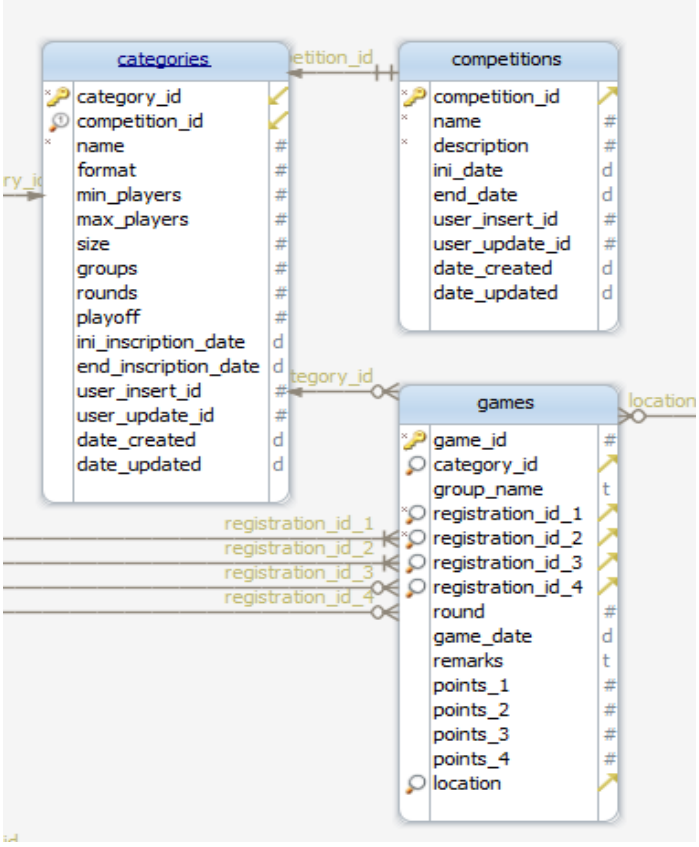


Ilustración 18. Detalle de competiciones, categorías y partidos

Una de las relaciones claves en la aplicación es la de competiciones, categorías y partidos como se muestra en la ilustración 18. La aplicación permite gestionar varias competiciones simultáneamente, y de diferente naturaleza. La aplicación puede gestionar

competiciones tipo round-robin, conocida popularmente como liga, eliminatoria o mezcla de ambas.

Cada competición posee una fecha de inicio y una de fin y varias categorías, según el nivel de los participantes, y cada categoría debe pertenecer a una única competición. Existe una fecha de registro para cada categoría. Cada categoría de la competición puede tener un formato de competición diferente, un número máximo y mínimo de jugadores y uno o varios grupos. Según esta información se generan los partidos para la categoría de la competición.

Los partidos son jugados por un mínimo de dos y hasta cuatro equipos. El caso de cuatro equipos es el caso de una competición de participantes individuales, en los que todos los jugadores juegan con y contra todos. La aplicación almacena el número de puntos o sets de cada participante en el juego para realizar la posterior clasificación.

Cada juego tiene una fecha y una localización que es calculada por el sistema automáticamente según la disponibilidad de los jugadores y las pistas de los clubes. El esquema de la tabla de disponibilidad se muestra en la ilustración 19.

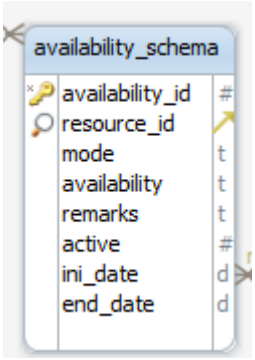


Ilustración 19. Tabla de almacenamiento de disponibilidad

La tabla *availability_schema* almacena la información de disponibilidad de las pistas y de los usuarios/jugadores. El registro *Mode* almacena si estamos hablando de pistas o de jugadores. Una pista o jugador puede tener varios calendarios disponibles simultáneamente, siendo el campo active del registro el que indica cuál de ellos es el que se debe tener en cuenta para la asignación de partidos.

8.6.Estructura de la aplicación

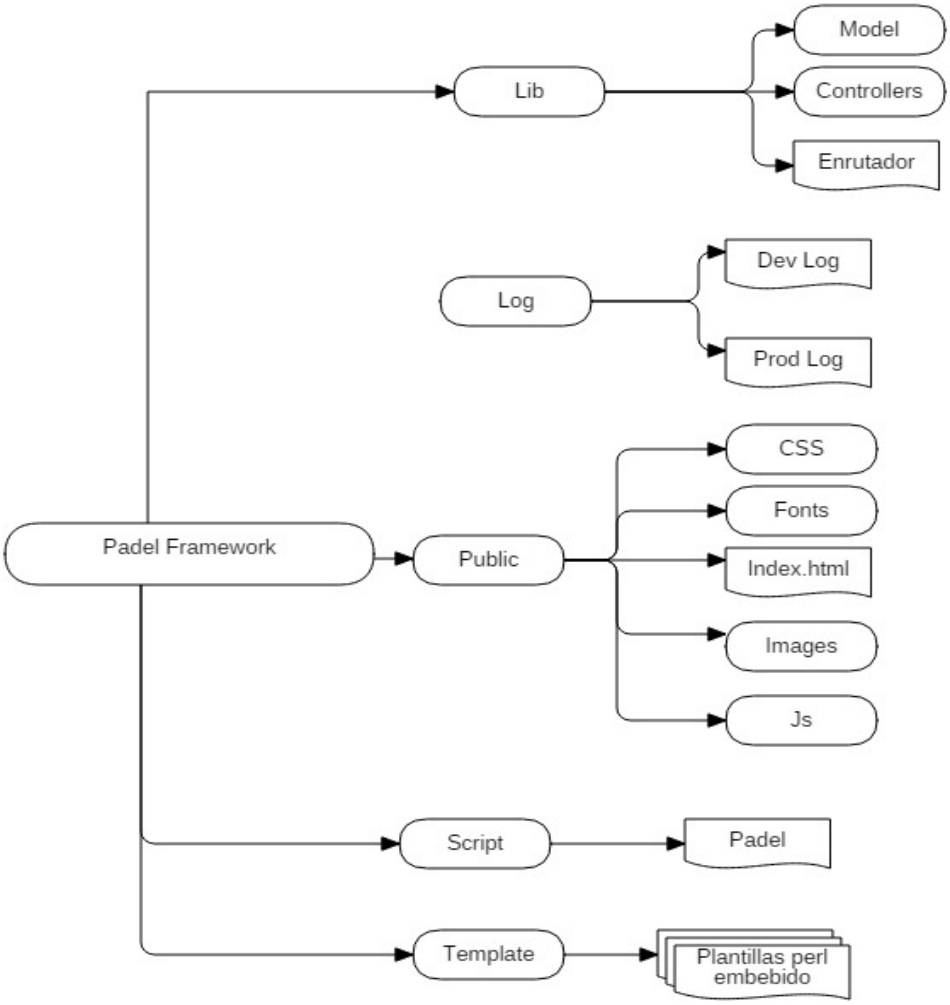


Ilustración 20. Estructura de la aplicación

El framework de Mojolicious viene estructurado para soportar el diseño controlador-vista-modelo. Para ayudarnos en esta tarea al crear un nuevo proyecto, se genera un árbol de directorios similar al que se ve en la imagen superior – ilustración 20 -.

La carpeta *Lib* almacena el archivo que soporta las rutas al sistema. En este proyecto además se ha intentado que todas las rutas respondan al protocolo REST. Dentro se almacena el árbol de directorios de cada uno de los módulos de la aplicación, donde se definirán cada uno de los métodos de los controladores y su enlace con el modelo de la aplicación. También guarda todos los archivos del modelo de la aplicación.

La carpeta Log genera automáticamente los logs del sistema. El entorno Mojolicious genera automáticamente dos entornos, uno de desarrollo y uno de producción.

La carpeta *public* guarda todos los archivos estáticos que son servidos automáticamente. Aquí se ha generado varios archivos:

- JS: Guarda todos los archivos javascript.
- CSS: Hojas de estilo.

- Fonts: Fuentes necesarias.
- Images: Imágenes.
- Index html. Página estática de bienvenida generada automáticamente.

La carpeta *Script* guarda el script de arranque de la aplicación.

La carpeta *template* almacena las plantillas que son renderizadas al ser enviadas al navegador. Para cada uno de los controladores se debe crear una carpeta y dentro de las carpetas un fichero para cada acción del controlador.

8.7.Interfaz

El diseño de la interfaz se centra en tres áreas de interés: El diseño de la interfaz entre los componentes del software, el diseño de las interfaces entre el software y los otros productores y consumidores de información no humanos (esto es, otras entidades externas) y el diseño de la interfaz entre el usuario y la computadora. Esta sección se centrará exclusivamente en la tercera categoría del diseño de interfaz – diseño de interfaz de usuario.

8.7.1.Importancia de la interfaz

Se ha intentado poner especial cuidado en el diseño de las interfaces de usuario. Si una aplicación es difícil de utilizar causa frustración para conseguir los objetivos y no será de agrado independientemente de la potencia informática que pueda ofrecer.

La aplicación trabaja con gran cantidad de datos y si éstos no son fácilmente accesibles por el usuario, la aplicación no conseguiría uno de sus objetivos principales.

Es cierto que las interfaces gráficas, ventanas, iconos y selecciones mediante ratón han eliminado muchos problemas, pero incluso en un "mundo de ventanas" podemos encontrar interfaces difíciles de aprender, de utilizar, confusas y frustrantes.

8.7.2.Estructura y diseño

Theo Mantel, en su libro [MAN97] crea tres reglas de oro para el diseño de la interfaz que se han intentado respetar en el proyecto:

- Dar el control al usuario.
- Reducir la carga de memoria del usuario.
- Construir una interfaz consecuente.

Una vez analizados los aspectos concretos que deberá poseer la aplicación, definida la base de datos en la que almacenar los contenidos y establecida la organización general, se define un diseño genérico para cada una de las funcionalidades del software. El diseño general de la aplicación sigue la siguiente estructura, con una serie de características comunes que se muestran remarcadas para su posterior explicación, cómo se muestra en la ilustración 21.

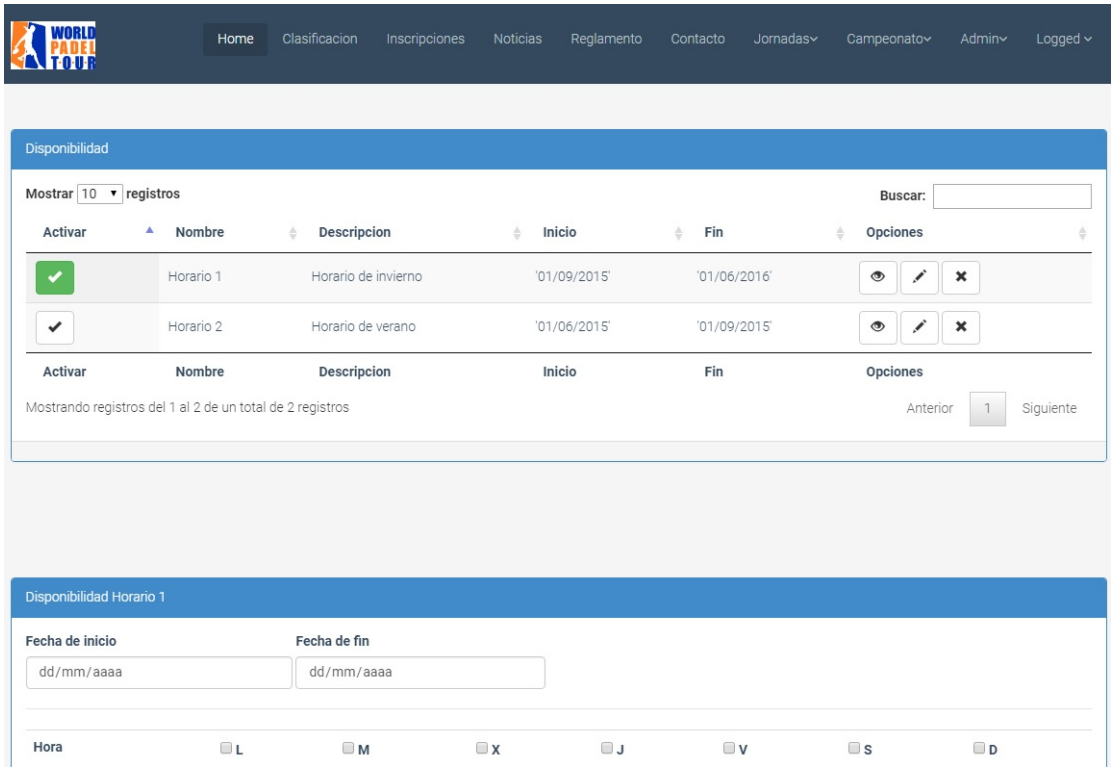


Ilustración 21.Ejemplo de diseño de la aplicación

En la parte superior se ha creado un menú con diferentes opciones según los roles del usuario. En función de los permisos podrá acceder a diferentes pantallas de la aplicación.

Al ser una aplicación web, se ha añadido al menú la opción home, que será la pantalla principal de la aplicación para cualquier usuario que acceda a la web.

Se ha categorizado los diferentes apartados de la aplicación en grupos y cada una de las opciones dentro de los grupos con un menú desplegable en pestañas.

El diseño que se ha elegido permite mucha flexibilidad, ya que añadir nueva funcionalidad a la aplicación se realiza de una manera sencilla, insertando nuevos elementos en el menú superior.

8.8.Diagrama de clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

Tal y como se ha indicado en anteriores apartados, se utilizará el patrón Modelo Vista Controlador. Para poder soportar la separación de la interfaz de usuario (Vista) de la lógica del programa (Controlador), se ha diseñado un diagrama de clases que permita abstraer la programación interna de Perl de los documentos de maquetación HTML. Además, deberá existir una serie de librerías separadas por módulos que permitan realizar acciones sobre la base de datos PostgreSQL (modelo). Éste esquema se muestra en la ilustración 22

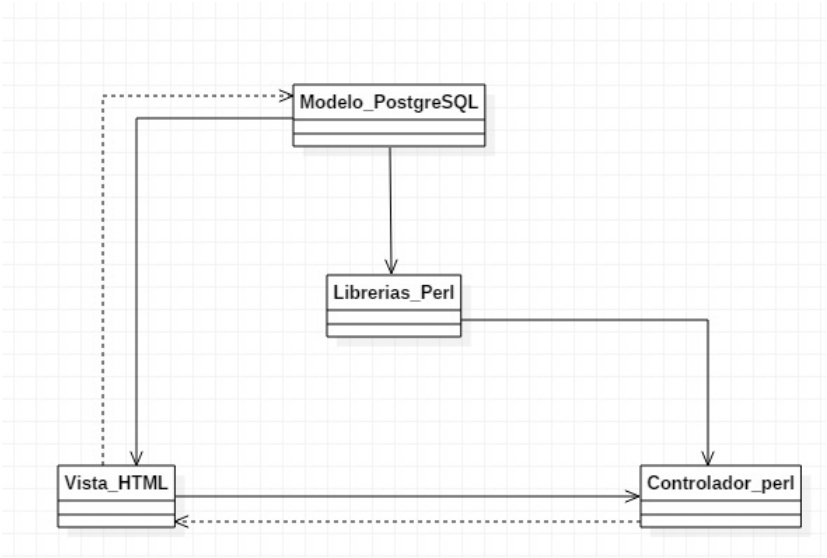


Ilustración 22. Modelo vista controlador

Mojolicious posee una estructura de clases como se puede observar en la siguiente imagen:

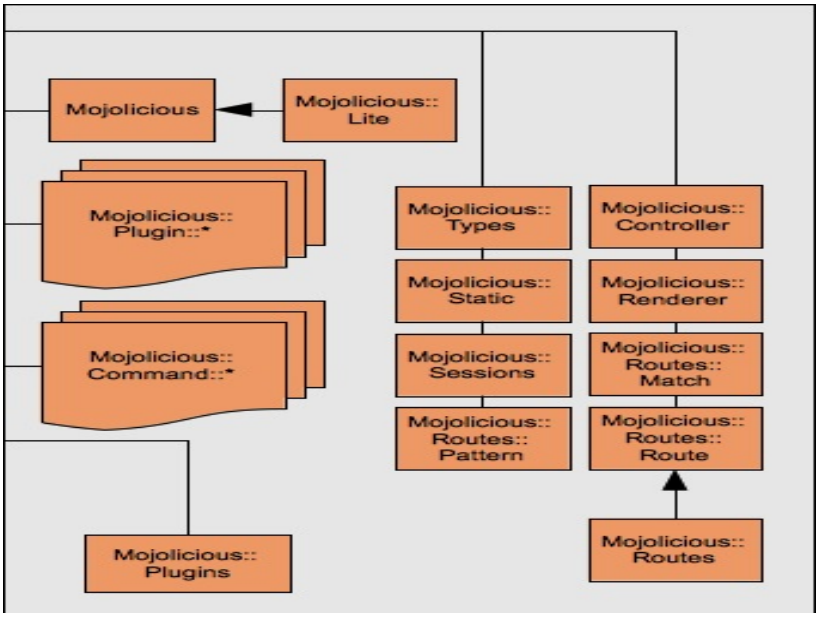


Ilustración 23. Estructura de Mojolicious

Al crear una aplicación Mojolicious estos módulos son instalados automáticamente. La aplicación de pádel usa los varios plugins de la librería CPAN de Mojolicious. Entre ellos destacan el plugin de autorización y el de autenticación de la aplicación.

También usa el módulo de acceso a la base de datos, de escritura en el Log y de rutas. El diagrama de clases queda como se ve a continuación:

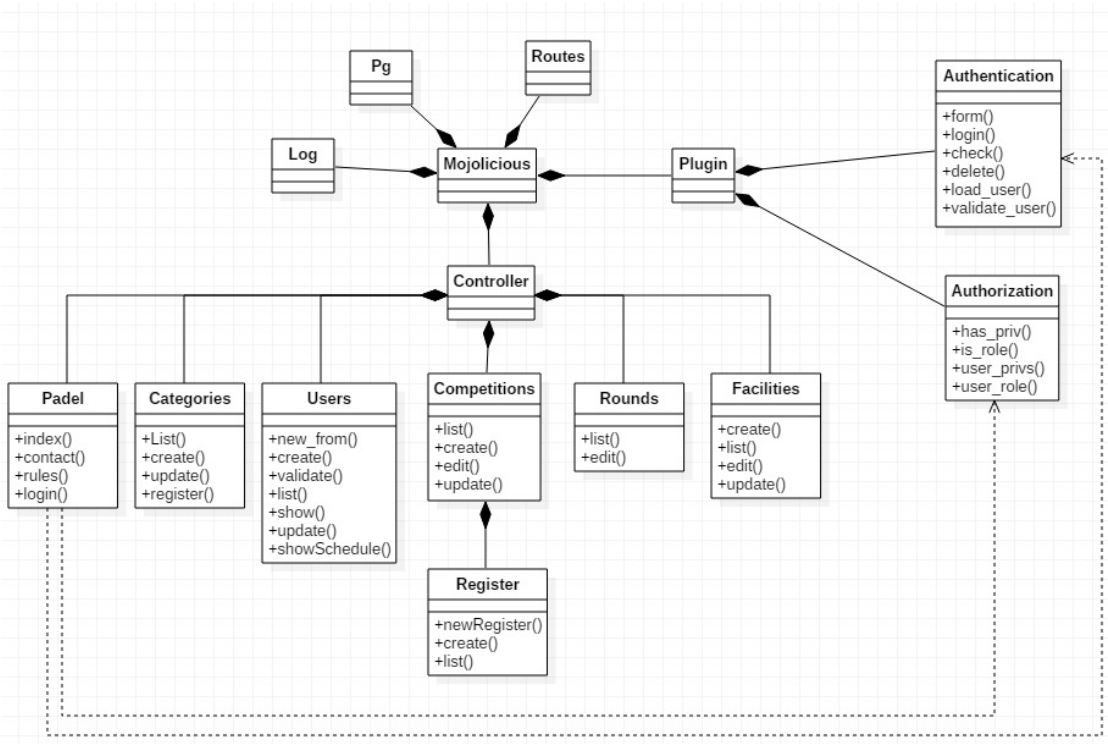


Ilustración 24. Diagrama de clases

Existen algunas clases que son usadas directamente por la aplicación para su uso base y que son heredadas por las demás clases. Un ejemplo claro es la librería PG, que es la que realiza la conexión a la base de datos. Esa clase es usada por la clase base Mojolicious, y por todas las demás que heredan de la clase base.

La clase *Routes* es la encargada de determinar a qué controlador apunta cada una de las rutas del sistema.

Dentro de la clase *plugins* se han utilizado el de autorización y el de autenticación. Autorización permite acceder a ciertas rutas según el rol del usuario, el de autenticación permite entrar o no a la aplicación.

La aplicación define las clases propias dentro de la clase *controller*. Cuando se realiza la llamada al controlador, éste determina a que método de que clase está llamando.

8.9.Diagramas de secuencia

Los diagramas de secuencia UML ofrecen una representación abreviada de la forma en la cual las acciones del usuario (los elementos dinámicos de un sistema que definen los casos de uso) colaboran con las clases de análisis (los elementos estructurales de un sistema que definen los diagramas de clases). [PRE03].

Para facilitar la comprensión del sistema, se muestra a continuación un ejemplo de diagrama de secuencia de la aplicación.

8.9.1.Sistema de gestión

La base de datos ha sido diseñada de forma que la gestión de los diferentes elementos se realice de manera similar. La gestión de usuarios, pistas, horarios, categorías y torneos sigue una estructura y lógica interna similar. Por esta razón, se presenta a continuación un diagrama de secuencia que representa el sistema de gestión.

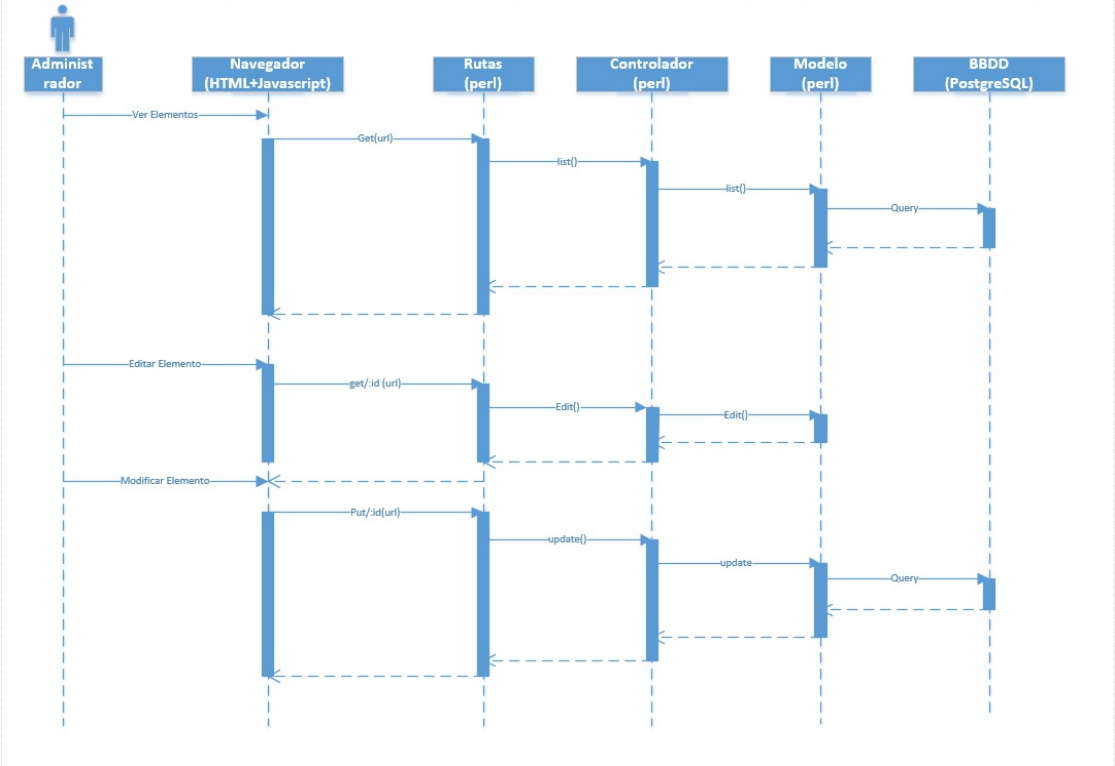


Ilustración 25. Diagrama del sistema de gestión

El usuario principal de la aplicación es el encargado de gestionar el torneo. El usuario interactúa con el sistema mediante el navegador web, aunque el framework permite la abstracción del cliente que use el usuario, permitiendo así que la aplicación sea multiplataforma.

El navegador (FrontEnd) que está desarrollado en HTML, Bootstrap y Jquery solicita al backend mediante peticiones AJAX la información para poder mostrarla al usuario. Para las rutas se ha optado por un protocolo RestFull, cómo se muestra en la tabla 5.

HTTP Verb	Path	Action	Used for
GET	/admin/posts	index	admin_posts_path
GET	/admin/posts/new	new	new_admin_post_path
POST	/admin/posts	create	admin_posts_path
GET	/admin/posts/:id	show	admin_post_path(:id)
GET	/admin/posts/:id/edit	edit	edit_admin_post_path(:id)
PATCH/PUT	/admin/posts/:id	update	admin_post_path(:id)
DELETE	/admin/posts/:id	destroy	admin_post_path(:id)

Tabla 5. Protocolo RestFull

Así al realizar una llamada a la URL del objeto al que queremos acceder y según el verbo HTTP que usemos, el enrutador llamará al método del controlador que corresponda. En el caso concreto que mostramos en la ilustración 25 al llamar a la ruta con el verbo GET, llamaremos al método list, que es similar al index que se muestra en la tabla 5. El controlador hace la llamada al modelo, que realiza la consulta a la base de datos para obtener los elementos, estos elementos son devueltos al modelo, que después

de tratarlos, los entrega al controlador, que los envía al navegador en el formato solicitado.

En la edición de los datos se pueden dar varios casos. Puede ocurrir que el navegador (cliente) tenga todos los datos para poder editar. En éste caso no hará falta realizar una llamada para mostrar el formulario al backend, será el código del frontend el que se encargue de realizar esa función. En otros casos es necesario obtener información del modelo de datos, por lo que se realizará la llamada AJAX correspondiente. Según como se ha definido en la tabla superior con el API REST, se realiza un llamada GET con el id del elemento que deseamos editar y añadimos /edit al final de la url.

Para la actualización de datos se sigue el mismo procedimiento, pero en este caso con el verbo PUT y el id del elemento que se va a actualizar.

El proceso para cada una de las acciones es bastante directo. Cuándo se desea añadir una acción sobre la aplicación, se debe tener en cuenta a que ruta de que controlador llamar, y respetar el protocolo RESt que se ha diseñado.

9.Desarrollo

En esta sección se procede a explicar la fase de implementación realizada para el proyecto. Se detallarán los aspectos más importantes de la interfaz, las clases externas utilizadas así como aquellas partes más importantes de las clases propias, tecnologías utilizadas, etc.

La opción elegida para desarrollar la interfaz descrita en la fase de Diseño ha sido la utilización de una plantilla de bootstrap, a la que se ha añadido la funcionalidad necesaria para cubrir los requisitos expuestos en la fase de análisis. A esto se le ha añadido la librería jquery ya que facilita la implementación de la funcionalidad del DOM, y las llamadas al backend de la aplicación.

La aplicación a desarrollar posee numerosa funcionalidad, y mucha dependencia de los requisitos de los usuarios. La aplicación debe ser flexible para ajustarse a los cambios de normativas. Por eso se ha optado por el uso de un framework, que mantenga la estructura de la aplicación, y que permita un mantenimiento sencillo. También facilita el desarrollo al poseer muchas herramientas ya incluidas en el propio framework, y aquellas que no están pueden añadirse a través de plugins de mojolicious, o perl.

El código fuente al completo se puede encontrar en el CD adjunto a la memoria.

9.1.Detalles de implementación

9.1.1.Introducción

En este apartado se indican los aspectos más relevantes de la implementación. Se ha intentado separar en todo momento el modelo de la vista y el controlador, intentado que cada apartado sea independiente, permitiendo así una mayor flexibilidad en la aplicación.

9.1.2.Estructura interna del sistema

Uno de los aspectos fundamentales de la aplicación es la organización de todas sus pantallas.

Con el fin de agilizar todos los desarrollos de códigos, todas las herramientas siguen una misma estructura interna para cubrir las funcionalidades. La mayoría de la información puede presentarse de manera ordenada en forma de tabla siguiendo el patrón MVC. Las pantallas de equipos, usuarios, categorías, competiciones, disponibilidad de horario, etc., siguen la misma filosofía gracias a la organización de la base de datos.

El sistema está organizado en controladores, plantillas (HTML), archivos estáticos (Javascript, Bootstrap), enrutador, controladores, modelos y tablas en la base de datos. Para conocer la relación entre ellos se muestra a continuación la lógica de la aplicación ante una petición de un listado de usuarios.

En el menú superior el usuario solicita dentro de Admin, gestionar usuarios.

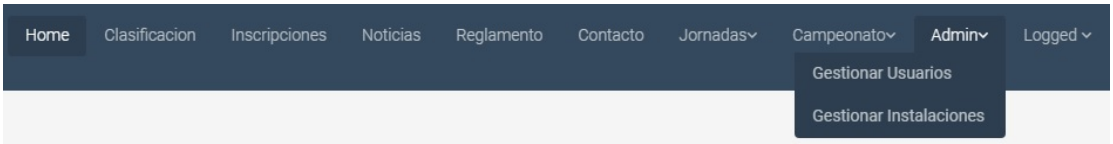


Ilustración 26. Menú superior de la aplicación

Al seleccionar la opción, se realiza una solicitud al enrutador a la ruta /users con un GET.

```
my $r_auth = $r->under->to('authentication#check')->name('authentication_check');
$r_auth->get('/users')->to('users#list');
```

El fichero enrutador busca la ruta solicitada y comprueba que está dentro de las rutas que solicitan estar autenticado. Si el usuario ha realizado el login y tiene permiso, el enrutador llama al controlador *users*, y al método *list*.

El controlador *list* realiza una llamada al modelo para obtener la lista de usuarios. El modelo realiza la consulta a la base de datos y responde al controlador. El controlador según el tipo MIME de la petición devolverá la respuesta en el formato solicitado:

```
$self->respond_to(
    html => {
      user_document => (defined($self->session->{authenticated_user}))?$self->session->{authenticated_user}-
>{document}:'',
      msg      => 'List Users',
      users    => \@users,
    },
    json =>{json=> \@users}
  );
```

Si el tipo mime solicitado es html se enviará junto con la lista de usuarios, el template (html) que se haya definido en el subdirectorio de *templates* correspondiente, en caso de que el tipo mime se json, se enviará la lista de usuario dentro de un objeto JSON.

9.1.3.Generar competiciones

La generación de competiciones permite la generación del calendario de partidos de cada competición. Para llegar a ese punto se debía desarrollar mucha funcionalidad alrededor. Era necesaria una gestión de jugadores, de equipos, de competiciones, de categorías, de clubes y de pistas.

Una vez se finalizó el desarrollo de la gestión de estos ítems, se debía generar los enfrentamientos entre los equipos, esto es, generar el calendario de partidos según el tipo de competición solicitada.

Para la generación de competiciones existen dos algoritmos según sea round-robin o por eliminatoria.

Uno de los algoritmos más usados para generar competiciones tipo liga, esto es, round-robin, es marcar un equipo como fijo, y pivotar el resto alrededor de él.

De esta manera para generar un calendario de una competición round robin quedaría:

Se crean un pivote y dos vectores de equipos, v1 y v2. Se genera la primera jornada:

```
function primeraJornada(){
    pivote = equipos[0];
    for (i=1;i<equipos.length;i++){
      if(i%2==0) v1.push(equipos[i]);
      else v2.push(equipos[i]);
    }
  }//primeraJornada
```

Se rotan los equipos:

```
  function rotarEquipos(){
    v2.unshift(v1.shift());
    v1.push(v2.pop());
  }//rotarEquipos
```

Y finalmente se generan el resto de las jornadas:

```
  function generarJornada(jornada){
pivote – v2[0]
    for( var i=1;i<v2.length;i++){
v1[i-1] vs v2[i]
    }
  }//generarJornada
```

Una vez generadas todas las jornadas de la competición se permite al administrador que genere nuevas jornadas, hasta que estas quedan definitivas.

9.1.4. Heurística para la asignación de pistas

La asignación de pistas presentaba el mayor desafío del proyecto. Se debe tener en cuenta la disponibilidad de cada uno de los jugadores de cada equipo, de cada partido y la disponibilidad de pistas de cada club.

Para ello la aplicación debe primero conocer cuál es la disponibilidad de cada jugador.
Jugador 1 del equipo 1: lunes a viernes de 16:00 a 21:00
Jugador 2 del equipo 1: lunes a viernes de 18:00 a 21:00
Jugador 1 del equipo 2: martes a sábado de 16:00 a 21:00
Jugador 1 del equipo 2: lunes a domingo de 15:00 a 22:00

Una vez conocida realiza una unión con la disponibilidad de sus compañeros de equipo, generando un vector de disponibilidades del equipo, con los jugadores más restrictivos.

Equipo 1: lunes a viernes de 18:00 a 21:00
Equipo 2: martes a sábado de 16:00 a 21:00

Conocida la disponibilidad de cada equipo, debemos conocer la disponibilidad de cada partido y ver si es viable asignar una pista automáticamente, o se debe notificar a los jugadores que deben adaptarse.

Partido 1: martes a viernes de 18:00 a 21:00

Una vez conocemos las disponibilidades de cada partido de la jornada, se ordenan de más restrictiva a menos restrictiva, y se busca la disponibilidad de pistas para ese equipo, asignando la hora en la que existan más pistas disponibles, si es posible.
Pista 1: Martes de 19:30 a 21:00
Pista 2: martes de 18:00 a 21:00

Partido 1: Martes 19:30 en pista 1

Una vez finalizado la generación de la asignación, se informa al gestor del campeonato para que lo valide y publique el resultado.

9.2.Librerías externas

9.2.1.DataTables

DataTables es un librería de la biblioteca jQuery – JavaScript. Es una herramienta muy flexible para el control de tablas HTML. De todas las características que ofrece, se destacan las siguientes:

- Paginación de longitud variable.
- Ordenación por múltiples columnas.
- Manejo inteligente de los anchos de columnas.
- Totalmente personalizable.
- Buscador dinámico.

Para que este paquete funcione correctamente se necesita una serie de librerías y hojas de estilo. Las librerías se sitúan en la vista general de la aplicación.

```
<link href="/css/jquery.dataTables.css" rel="stylesheet">
<script src="/js/jquery.js"></script>
<script src="/js/jquery.dataTables.min.js"></scrip>
<script src="/js/dataTables.tableTools.min.js"></script>
<script src="/js/jquery.dataTables.editable.js"></script>
<script src="/js/jquery.jeditable.js"></script>
```

Con las librerías incluidas adecuadamente, el siguiente paso es indicar a qué tabla del código HTML se le atribuye las características de la librería datatable:

```
        $('#table_users').DataTable({
            "data":data,
            "columns":[
              {title: 'Activo', data:"active", render: function (data){return (data==1)?'<span
        class="glyphicon glyphicon-ok" aria-hidden="true"></span>':'<span class="glyphicon
        glyphicon-remove" aria-hidden="true"></span>'}},
              {title: 'Nombre', data:"name"},
              {title: 'Apellidos', data:"surname"},
              {title: 'DNI', data:"document"},
              {title: 'Email', data:"email"},
              {title: 'Tlf 1', data:"phone"},
              {title: 'Tlf 2', data:"mobile"},
              {title: 'Enviar Email', data:"user_id", render: function (data){return '<input
        type="button" value="Enviar email" onclick="validateUser(\''+data+'\')">'}},
            ],
});
```

Datatable permite indicar en que variable vendrán los datos y cómo se define cada columna. Las columnas se pueden renderizar directamente, dónde *data* responde al campo del objeto JSON que se ha indicado en la tabla, o bien renderizar mediante una función específica.

9.2.2. Bootstrap

Bootstrap es un framework de desarrollo HTML, CSS y JS para el desarrollo de proyectos responsive en la web.

Para usarlo se debe añadir los ficheros css y javascript en el layout principal de la aplicación:

```
<link href="/css/bootstrap.min.css" rel="stylesheet">
<link href="/css/font-awesome.min.css" rel="stylesheet">
<link href="/css/prettyPhoto.css" rel="stylesheet">
<link href="/css/animate.css" rel="stylesheet">

<script src="/js/bootstrap.min.js"></script>
<script src="/js/jquery.prettyPhoto.js"></script>
```

La inclusión de bootstrap al proyecto permite que el diseño de la aplicación sea común. Crea un formato general con una librería de componentes que resuelven gran parte del diseño de la aplicación.

10. Pruebas del sistema

La realización de pruebas del sistema es una parte importante para la comprobación de que los requisitos están cubiertos. Para realizar las pruebas se ha solicitado a un gestor de torneos un listado de jugadores y disponibilidad de canchas para una liga que está organizando.

En la temporada 2016/2017, en dicha liga hay un total de 2 grupos y 12 equipos por grupo. En las pruebas del sistema se han omitido los nombres por protección de datos.

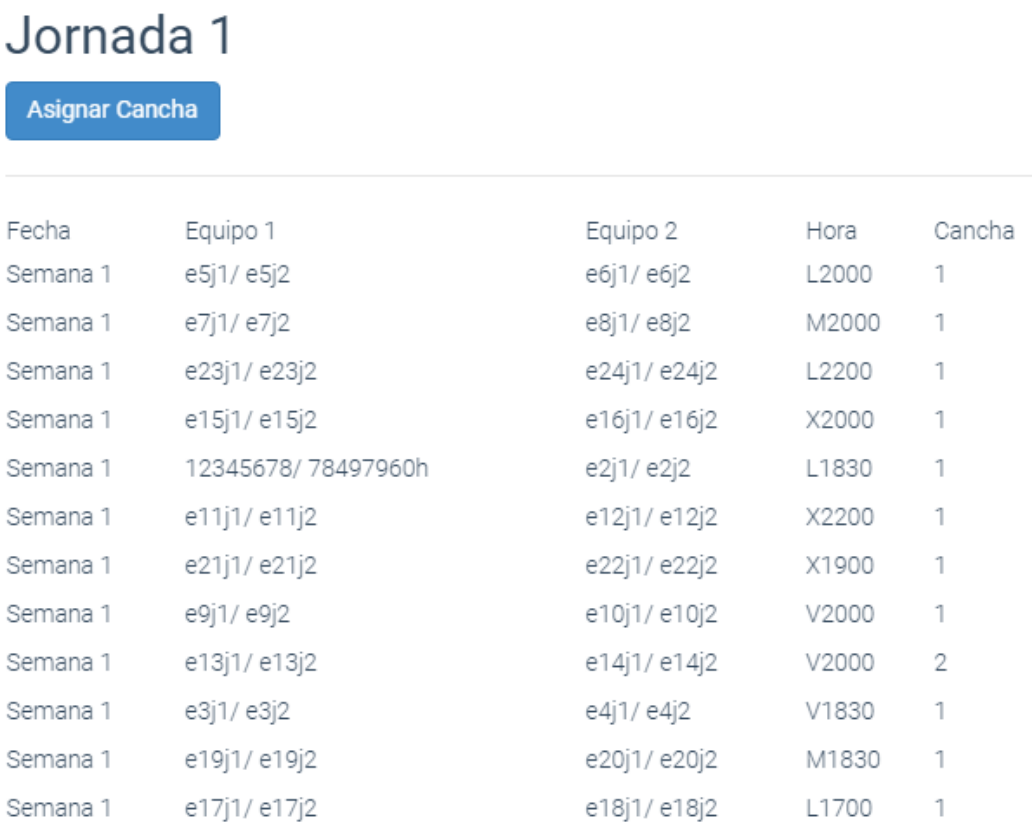
La disponibilidad de canchas queda como sigue:

Lunes: 1 cancha a las 17:00, otra a las 18:30 y una a las 21:00
Martes: 1 cancha a las 18:30, otra a las 20:00 y una a las 21:00
Miércoles: 1 cancha a las 18:30, otra a las 20:00 y una a las 21:00
Jueves: 1 cancha a las 17:00, otra a las 18:30 y una a las 21:00
Viernes: 1 cancha a las 17:00, otra a las 18:30, 2 a las 20:00 y una a las 21:00

La disponibilidad de los jugadores queda resumida en la siguiente tabla, donde cada fila se corresponde con las preferencias horarias de cada una de las parejas participantes:

	Preferencia horarios
1	Martes y jueves a partir de las 20:00. Viernes a partir de las 18:30.
2	Lunes a viernes a partir de las 19:00.
3	Lunes a viernes a partir de las 18:30.
4	Todos los días a partir de las 20:00 y sábados por la mañana
5	Lunes y miércoles desde las 18:30 hasta las 21:30 (hora max finalización del partido). Martes y viernes desde 16:00 hasta las 21:30 y fines de semana a partir de las 11:00. También por las mañanas.
6	Lunes a viernes a partir de las 18:00.
7	Lunes a viernes a partir de las 19:30.
8	Lunes a viernes a partir de las 19:30
9	Lunes a viernes a partir de las 17:00 y la última hora a la que pueden jugar es las 20:00.
10	Lunes a viernes a partir de las 20:00.
11	Lunes a viernes a partir de las 18:30.
12	Lunes a viernes a partir de las 20:30 y sábados a partir de las 14:00.
13	
14	Lunes a viernes a partir de las 19:00 y sábados por la mañana.
15	Lunes a viernes a partir de las 16:00.
16	Lunes, miércoles y viernes a partir de las 19:00.
17	Trabaja en turno de tarde.
18	Lunes a viernes a partir de las 16:30.
19	Lunes a viernes a partir de las 18:00.
20	Lunes a viernes a partir de las 18:00 y algún fin de semana.
21	Miércoles y viernes a partir de las 18:00. Jueves a las 20:00.
22	Trabajan por turnos. Los viernes pueden a cualquier hora. Turno día: a partir de las 20:30. Turno noche: pueden terminar como muy tarde a las 20:30.
23	De lunes a viernes a partir de las 18:00.
24	Lunes a jueves a últimas hora de la tarde y los sábados por la mañana.

Los resultados del algoritmo de generación automática de horarios se muestra a continuación:



Fecha	Equipo 1	Equipo 2	Hora	Cancha
Semana 1	e5j1/ e5j2	e6j1/ e6j2	L2000	1
Semana 1	e7j1/ e7j2	e8j1/ e8j2	M2000	1
Semana 1	e23j1/ e23j2	e24j1/ e24j2	L2200	1
Semana 1	e15j1/ e15j2	e16j1/ e16j2	X2000	1
Semana 1	12345678/ 78497960h	e2j1/ e2j2	L1830	1
Semana 1	e11j1/ e11j2	e12j1/ e12j2	X2200	1
Semana 1	e21j1/ e21j2	e22j1/ e22j2	X1900	1
Semana 1	e9j1/ e9j2	e10j1/ e10j2	V2000	1
Semana 1	e13j1/ e13j2	e14j1/ e14j2	V2000	2
Semana 1	e3j1/ e3j2	e4j1/ e4j2	V1830	1
Semana 1	e19j1/ e19j2	e20j1/ e20j2	M1830	1
Semana 1	e17j1/ e17j2	e18j1/ e18j2	L1700	1

Como se puede comprobar, el sistema ha sido capaz de establecer un horario para todos los partidos de una de las jornadas generada como ejemplo.

11.Resultados y conclusiones

El trabajo realizado a lo largo de este proyecto fin de carrera, en el que se han llevado a cabo las fases de estudio, análisis, diseño y desarrollo, ha dado una serie de resultados de interés y se han podido obtener diversas conclusiones.

11.1.Resultados

Los resultados presentados a continuación son el fruto del trabajo en el que se han cumplido todos los objetivos propuestos y se han realizado tareas opcionales para completar la calidad global de la aplicación.

En todos los casos la aplicación se planteó como un entorno vivo y flexible, en el que cualquier modificación pueda ser llevada a cabo sin mucho esfuerzo por parte del desarrollador.

11.1.1.Interfaz

Un objetivo clave de cualquier aplicación que interaccione con el usuario es lograr una interfaz amigable y sencilla. En este proyecto se ha intentado cuidar al máximo el diseño e implementación de la interfaz.

La interfaz de la aplicación presenta la información de forma consecuente, es decir, toda la información que se presenta al usuario está organizada de acuerdo con el diseño estándar que se mantiene en todas las presentaciones de pantalla. El resultado final ha sido la consecución de una interfaz sencilla, clara y altamente interactiva que agiliza el proceso de la gestión de torneos.

11.1.2.Gestión de torneos

La gestión de torneos ha presentado un reto en cuánto a las numerosas páginas de gestión necesarias para poder crear un calendario. Se ha desarrollado pantallas de gestión de usuarios, equipos, clubes, pistas, torneos, categorías, etc.

Se ha intentado que las páginas de gestión sean lo más común posible en cuanto a la interfaz, para facilitar a los diferentes usuarios la interacción con la aplicación.

11.1.3.Generación de torneos

La generación de torneos ha sido otra de las partes claves. Encontrar el método necesario para la generación de un torneo tipo liga ha implicado conocer los diferentes formatos de competición que existen. Esta operación ha llevado a descubrir lo extenso y variado de los formatos de competición.

11.1.4. Asignación de pistas

La asignación de pistas es la tarea clave de la aplicación. Se han aplicado diferentes heurísticas para encontrar aquella con un porcentaje mayor de acierto.

La asignación de pistas se puede hacer tantas veces como se desee, hasta que se considere que el calendario está cerrado. A partir de ese momento la gestión de los horarios pasa a un método manual.

La asignación también permite fijar ciertos partidos e intentar buscar una respuesta a los partidos pendientes, ayudando así al gestor del torneo a la generación del calendario.

La ventaja para un gestor de competición es el ahorro en tiempo debido a la automatización de tareas. El gestor pasa a ser un espectador y supervisor de la información que genera la aplicación, y actúa en aquellos casos que la aplicación no puede dar respuesta.

11.1.5.Base de Datos

Desde un inicio se planteó el diseño e implementación de la base de datos como una tarea fundamental. Se perseguía cumplir una serie de objetivos principales, no obstante, se diseñó la base de datos pensando en las posibles mejoras futuras.

Se ha conseguido implementar una base de datos robusta, bien estructurada y no redundante que es capaz de dar soporte a toda la información que puede manejar un centro.

Además, las características y el diseño de la base de datos, permiten más funcionalidades de las que actualmente posee.

11.1.6.Accesibilidad y optimización

Durante el proceso de desarrollo siempre se ha tenido presente aspectos fundamentales de cualquier aplicación basada en web. En concreto, la aplicación debe ser accesible desde cualquier dispositivo con acceso a Internet y un navegador web, además de estar optimizada para los diferentes navegadores del mercado.

En lo que se refiere a accesibilidad, se han respetado las recomendaciones establecidas por el consorcio internacional de la web, el "World Wide Web Consortium W3C". Se ha utilizado un maquetado web válido para que cualquier navegador pueda comprender la estructura y presentarla de forma adecuada al usuario.

Bootstrap ayuda en gran medida a llevar a cabo este proceso. Usando buenas prácticas de programación y la base Bootstrap con sus componentes, permite que la aplicación sea responsive, esto es, adaptable a diferentes tamaños.

11.1.7. Aportaciones en el contexto de herramientas de gestión de competiciones de pádel

La realización de este proyecto pretende innovar y abrir una nueva línea de trabajo en el desarrollo de aplicaciones web dedicadas a la gestión y generación de torneos. Se busca una aplicación integral de gestión de torneos de pádel. Es un punto de encuentro entre los participantes en el torneo y el gestor, así como el club de pádel asociado. Con un uso correcto de la aplicación, la comunicación entre los actores se reduce, y también el tiempo requerido en gestión para poder jugar partidos.

Se trata de una implementación basada en tecnologías web, por lo que los usuarios pueden conectarse en todo momento. Evitamos el uso de un equipo físico, y podemos gestionar e informarnos mediante cualquier dispositivo móvil, pudiendo realizar la gestión en cualquier momento y lugar.

11.2.Conclusiones

Desde el punto de vista de los objetivos iniciales planteados para este proyecto, los resultados obtenidos son muy satisfactorios.

Todos los objetivos propuestos han sido cumplidos. Se ha conseguido desarrollar una aplicación web capaz de generar y gestionar torneos de pádel. El punto fuerte se centra en la parte de gestión, permitiendo a un administrador de torneos gestionar su torneo de manera sencilla, evitando mucho trabajo de gestión.

El núcleo central de la aplicación se encuentra rodeado por un sistema de gestión de datos que la convierte en una aplicación completa. Por otra parte, se ha creado una interfaz sencilla, intuitiva y atractiva para los usuarios finales. Lejos de ser una

herramienta perfecta, sienta unas bases de las que pueden surgir múltiples líneas de trabajo futuro.

El enfoque que se ha seguido en el desarrollo de la aplicación no sólo trata de ser innovador, sino que pretende abrir una nueva línea de trabajo donde sea la aplicación la que cargue con todo el proceso y libere al gestor del torneo de las tareas administrativas. El gestor se convierte así en un supervisor de su torneo.

Desde el momento que se planteó el desarrollo del proyecto, se sabía que iba a suponer un gran reto. De hecho, ha sido un camino largo y con obstáculos en algunos puntos, pero lo que queda es la satisfacción de haber obtenido unos conocimientos que me han permitido crecer como profesional.

Los proyectos fin de carrera suponen una excelente oportunidad para profundizar y fortalecer todos aquellos conocimientos aprendidos y especialmente para estudiar las últimas tecnologías y tendencias que se están siguiendo en el desarrollo web. Por todo esto, se ha invertido mucho esfuerzo en tareas como el análisis, investigación y aprendizaje de lenguajes como Perl, JavaScript, Bootstrap, Postgresql y Mojolicious.

Para el desarrollo del proyecto se han tenido que combinar conocimientos adquiridos con el aprendizaje de aplicaciones y herramientas desconocidas, lo cual ha proporcionado unas nuevas aptitudes que pueden ser muy útiles en el futuro.

A nivel personal este proyecto me ha servido para profundizar en el conocimiento de Frameworks de desarrollo, y las herramientas utilizadas para el análisis y diseño de proyectos.

12.Trabajo futuro

El proyecto nació bajo unas necesidades y objetivos mínimos que debían cumplirse, pero un producto software nunca es perfecto en su primera versión y es por eso que se proponen unas líneas de trabajo futuro que pueden mejorar la aplicación:

- Gestión de notificación entre usuarios: Se trata de un sistema que permite a los equipos informarse de cambios en los partidos.
- Generación de playoffs: Una vez finalizada la liga, permitiría un sistema de playoff entre los primeros clasificados.
- Notificación de reservas a los clubes: Al conocer las necesidades de pistas, el sistema podría informar automáticamente a los clubes para realizar la reserva. Se podría comunicar con el sistema de gestión del club para realizar la reserva automáticamente.
- Lectura de CSV, XML para la disponibilidad horaria: Permite recibir de otras fuentes la información de disponibilidad horaria de pistas y jugadores.
- Generación de aplicación móvil: El sistema está preparado para añadir aplicaciones móviles nativas para los distintos dispositivos. Permite incluso que terceros tengan acceso al sistema para su uso.

13.Bibliografía

13.1.Libros.

- [PRE03] Pressman, Roger S.
  Ingeniería del Software, un enfoque práctico.
  Editorial McGraw-Hill España, 2003, pp. 21.
- [BRO75] Brooks, F.
  The mytical Man-Month.
  Addison-Wesley, 1975.
- [GIL88] Gilb, T.
  Principles of Software Engineering Management.
  Addison-Wesley, 1988.
- [BOE88] Boehm, B.
  A Spiral model for Software Development and Enhancement.
  Computer, vol. 21, nº 5, Mayo 1988, pp. 61-72
- [MAT03] Celma Giménez, Matilde.
  Bases de datos relacionales.
  Pearson Educación, Madrid, 2003.
- [MAN97] Mandel, T.
  The Elements of User Interface Design.
  Wiley, 1997.
- [COC01] Cockburn, Alistair.
  Writing effective use cases.
  Addison-Wesley, 2001.
- [ED07] Edmon, Woychowsky.
  Ajax, creating web pages with Asynchronous JavaScript and XML.
  Prentice-Hall, 2007
- [EG94] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
  Design Patterns. Elements of Reusable Object-oriented software
  Addison-Wesley, 1994

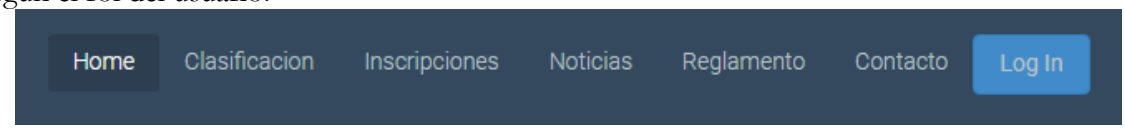
13.2.Web

- http://www.w3schools.com
- http://www.desarrolloweb.com
- http://mojolicious.org/
- https://www.cpan.org/
- http://www.postgresql.org.es/
- http://www.jquery.com
- http://www.jqueryui.com
- http://www.json.org
- http://getbootstrap.com/
- https://datatables.net/
- https://github.com/
- https://stackoverflow.com/
- https://git-scm.com/

14.Anexos

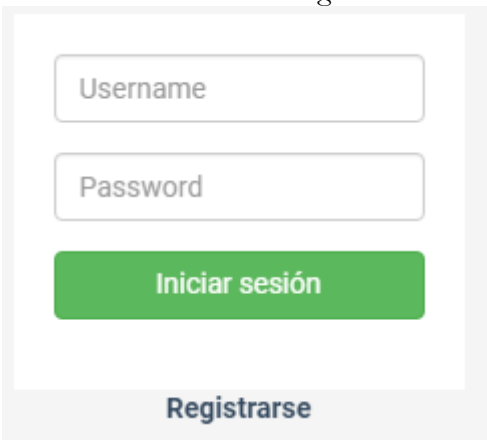
14.1.ANEXO A: Manual de usuario

En el menú superior de la página están todas las opciones que proporciona el sistema según el rol del usuario.



Existe una parte específica para conocer el reglamento del torneo. También hay una sección de noticias y para conocer la clasificación de la liga.

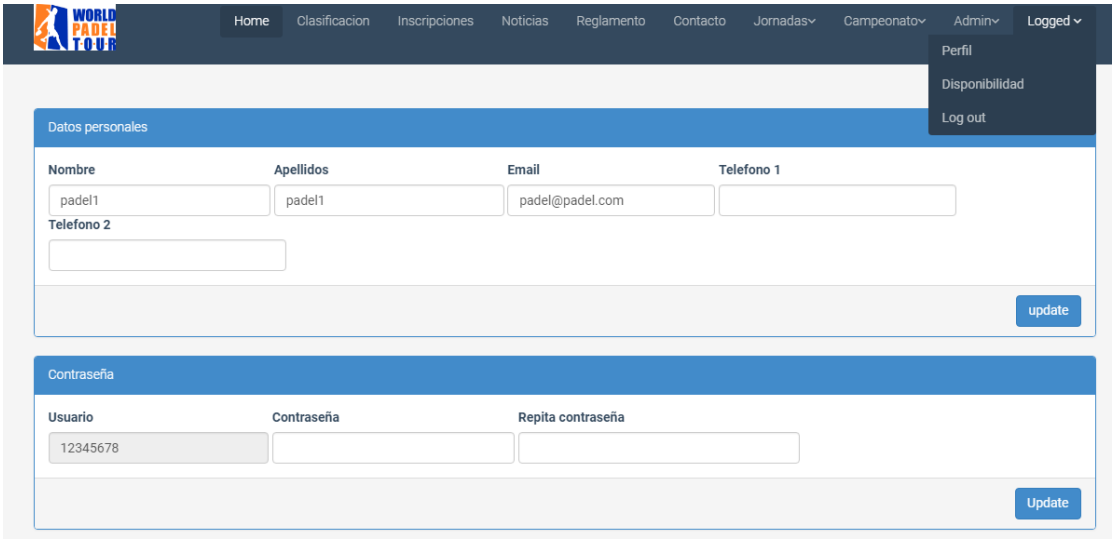
Registro:
Para registrarse en el sistema se accede desde el *login*.



Una vez en el *login*, se accede al registro. Es necesario el DNI y un correo válido para poder registrarse en el sistema. Una vez realizado el registro ya se puede acceder al sistema.

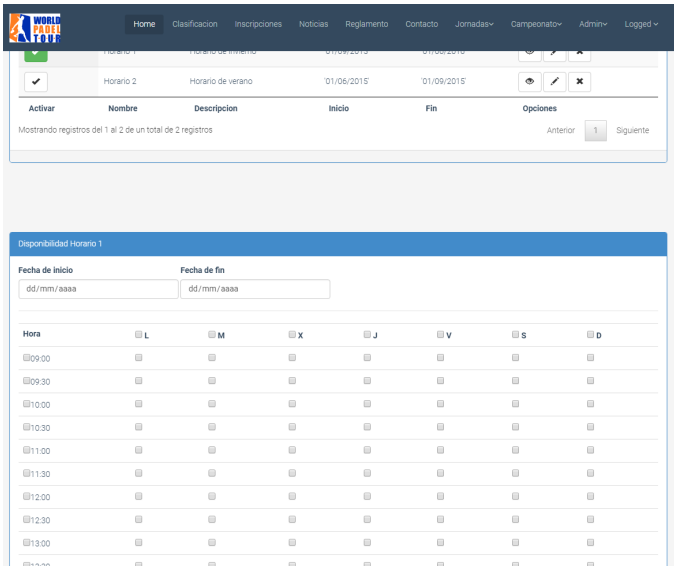
Inscripción:
Para inscribirse en un torneo se debe acceder a la sección de inscripción. Desde allí se ve un listado de los torneos activos y la opción para inscribirse. Sólo los usuarios registrados pueden inscribirse en los torneos.

Ver perfil:
Una vez que el usuario haya iniciado sesión en el sistema, el usuario puede cambiar sus datos personales y su contraseña.



Disponibilidad:

Cada usuario puede gestionar su horario de disponibilidad. Se pueden tener más de un horario disponible.



Administrar usuarios

Los usuarios administradores tienen la opción de gestionar los usuarios desde el menú Admin.

Gestión de instalaciones

Los usuarios administradores tienen la opción de gestionar las instalaciones desde el menú Admin. Desde aquí se pueden añadir nuevos clubes, pistas e indicar la disponibilidad.

Gestión de competiciones

Desde el menú de campeonato se permite gestionar las competiciones. Es necesario añadir una fecha de inicio y de fin de la competición.

Gestión de categorías

Desde el menú de campeonato se permite gestionar las categorías de cada competición. Es necesario añadir una fecha de inicio y de fin de la categoría.

Configurar competiciones

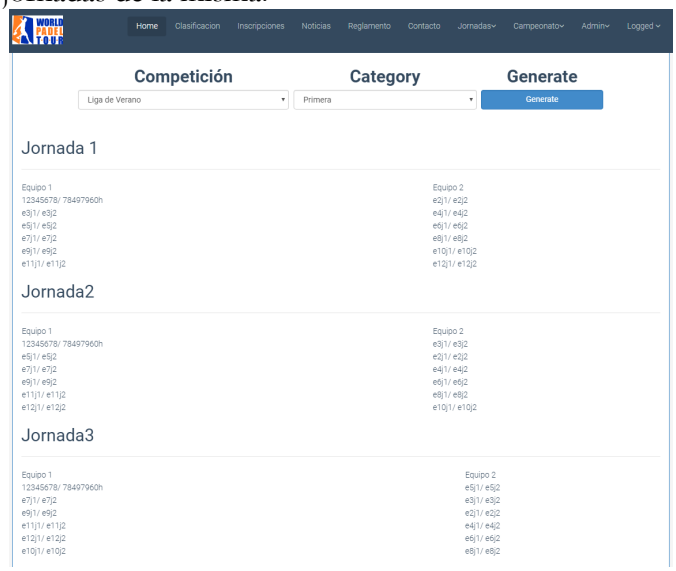
En el menú de campeonatos se permite configurar las competiciones. Aquí se determina cuantos puntos se da al ganador y cuantos al perdedor según el resultado de la competición. También se determina que tiene más valor en caso de empate.

Ver inscritos

Permite ver los inscritos en las diferentes categorías.

Generar jornadas

Una vez que la inscripción ha finalizado, el administrador del campeonato puede generar todas las jornadas de la misma.



Generar horario

Una vez generadas las jornadas, en el menú de jornadas se permite generar los horarios de cada partido y la asignación de canchas en caso de que sea posible. En este apartado es dónde los jugadores insertan los resultados al finalizar el partido.

14.2.Anexo B: Guía de instalación

En el repositorio digital se encuentra el código del proyecto y una imagen de la máquina virtual donde se ha desarrollado.

La manera más sencilla de instalación es usar una herramienta de virtualización como "VirtualBox" y cargar la imagen del proyecto, ya que están instalados todos los módulos necesarios de la máquina.

En otro caso es necesaria una distribución linux para la instalación del proyecto. El primer paso es instalar el Framework Mojolicious con esta línea:

```
$ sudo -s 'curl -L cpanmin.us | perl - Mojolicious'
```

Una vez que se haya instalado Mojolicious en el sistema, empezamos un nuevo proyecto:

```
$ mojo generate Padel
```

En la carpeta Padel encontrará la estructura de directorios con la aplicación preconfigurada. Se cambian los ficheros existentes por el código que está en el repositorio general y ya tenemos la aplicación funcionando.

El Framework mojolicious está bien documentado en: `http://mojolicious.org/perldoc/Mojolicious`

14.3.Anexo C: Plantilla de casos de uso

Nombre	APUNTARSE A UN CAMPEONATO	ID	01
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	ANÓNIMO
Personal involucrado	Anónimo: Selecciona un campeonato y se apunta a él Administrador: Ve un nuevo jugador apuntado a su campeonato
Descripción	Un nuevo jugador se apunta a un campeonato
Trigger	
Precondición	
Postcondición	El jugador aparece en la lista de inscritos
Flujo Normal	3. Rellena el formulario de los datos personales 4. El jugador se añade a la lista de inscritos
Flujo alternativo	
Excepción	El jugador ya está inscrito Se ha superado el número máximo de inscritos
Includes	
Requisitos especiales	
Notas	

Nombre	CREAR CAMPEONATOS	ID	02
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	ADMINISTRADOR
Personal involucrado	Administrador: Crea un nuevo campeonato en el sistema Anónimo: Ve un nuevo campeonato en la lista
Descripción	Se genera un nuevo campeonato en el sistema
Trigger	
Precondición	
Postcondición	Se crea un nuevo campeonato
Flujo Normal	1. El administrador inserta los datos del campeonato 2. El campeonato es visible para los usuarios
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	DAR DE ALTA CLUBES	ID	03
Creado por	Josué Caballero López	Fecha	14/02/13
Modif. por		Fecha Modif	

Actor Principal	ADMINISTRADOR
Personal involucrado	Administrador: Inserta un nuevo club en sistema
Descripción	Se añade un nuevo club para un administrador. Se insertan los detalles del club. Dirección, teléfono, correo, etc
Trigger	
Precondición	
Postcondición	Sobre el club creado se insertan los recursos de los que dispone cada campeonato
Flujo Normal	1. El administrador inserta los datos del club
Flujo alternativo	2. El administrador inserta el número de pistas que tiene disponible en el club
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	DARSE DE BAJA DE UN CAMPEONATO	ID	04
Creado por	Josué Caballero López	Fecha	14/02/13
Modif. por		Fecha Modif	

Actor Principal	USUARIO REGISTRADO
Personal involucrado	Usuario registrado: Se da de baja de un campeonato
Descripción	El usuario registrado deja de participar en un campeonato
Trigger	
Precondición	El campeonato no ha comenzado
Postcondición	Hay un hueco libre en la generación del calendario
Flujo Normal	1. El usuario selecciona el campeonato en el que está inscrito 2. Indica que se quiere dar de baja del mismo
Flujo alternativo	3. Indica la razón de baja del campeonato 4. Se informa al administrador de la baja y el motivo
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	GENERAR HORARIOS	ID	05
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	ADMINISTRADOR
Personal involucrado	Administrador: El Administrador da la orden de generación de los horarios. Anónimo: Ve los horarios de la siguiente jornada
Descripción	Se generan los horarios de una jornada en concreto
Trigger	
Precondición	Se ha generado el calendario del campeonato
Postcondición	Aparecen la jornada siguiente con su horario
Flujo Normal	1. Se selecciona la jornada

	2. Se da la orden de generar el horario 3. Aparece el horario con sus incidencias
Flujo alternativo	
Excepción	No se puede generar el horario completo por falta de recursos o incompatibilidades entre equipos
Includes	
Requisitos especiales	
Notas	

Nombre	GENERAR INCIDENCIAS	ID	06
Creado por	Josué Caballero López	Fecha	14/02/13
Modif. por		Fecha Modif	

Actor Principal	USUARIO REGISTRADO
Personal involucrado	Usuario registrado: Inserta una incidencia en el sistema Administrador: Gestiona la incidencia para dar una respuesta
Descripción	El usuario registrado desea indicar una incidencia como falta de cancha, incompatibilidad de horario, etc
Trigger	
Precondición	
Postcondición	Se muestra una incidencia en el listado del administrador
Flujo Normal	5. Se selecciona el tipo de incidencia 6. Se envía 7. El administrador es notificado de la misma
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	GESTIONAR CAMBIOS EN EL CALENDARIO	ID	07
Creado por	Josué Caballero López	Fecha	06/12/12
Modif. por		Fecha Modif	

Actor Principal	USUARIO REGISTRADO
Personal involucrado	Usuario registrado: El usuario puede modificar los fecha de su partido según la disponibilidad Usuario registrado: El resto de los usuarios implicado en el partido ve las modificaciones propuestas y actúan en consecuencia
Descripción	El usuario puede modificar su partido durante un periodo de tiempo antes del mimo
Trigger	
Precondición	Se ha generado el calendario provisional y está en periodo de cambios
Postcondición	Se informa a los participantes del cambio propuesto
Flujo Normal	1. Se seleccionan el partido a modificar 2. Se observa la disponibilidad de pistas 3. Se selecciona una de las pistas vacías para jugar
Flujo alternativo	3b. Se propone el nuevo horario 3c. Se indica en las observaciones donde se disputaría el partido
Excepción	

Includes	
Requisitos especiales	
Notas	

Nombre	GESTIONAR INCIDENCIAS	ID	08
Creado por	Josué Caballero López	Fecha	06/12/12
Modif. por		Fecha Modif	

Actor Principal	ADMINISTRADOR
Personal involucrado	Administrador: Ve el listado de incidencias y actua sobre las mismas Usuario registrado: Se notifica la respuesta del administrador
Descripción	El administrador da respuesta a la incidencia
Trigger	
Precondición	
Postcondición	La incidencia ha sido gestionada
Flujo Normal	4. Se seleccionan la incidencia sobre la lista 5. Se envía una respuesta a la misma
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	INSERTAR RECURSOS	ID	09
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	ADMINISTRADOR
Personal involucrado	Administrador: Inserta recursos al sistema (pistas).
Descripción	Se insertan y gestionan recursos de un campeonato
Trigger	
Precondición	
Postcondición	El campeonato tiene nuevos recursos
Flujo Normal	1. Se selecciona el listado de recursos del campeonato 2. Se añade o modifica los recursos del campeonato 3. El sistema tiene en cuenta este cambio para asignación de horarios
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	INSERTAR Y EDITAR PREFERENCIAS DE HORARIO	ID	10
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	USUARIO REGISTRADO
Personal involucrado	Usuario registrado: Puede modificar sus preferencias de horario Administrador: Ver las modificaciones y las tiene en cuenta para la

	generación de horarios
Descripción	Se modifica las preferencias de horarios
Trigger	
Precondición	
Postcondición	Las preferencias de horarios han sido modificadas. El sistema las tiene en cuenta para la generación del horario.
Flujo Normal	1. Se selecciona el horario actual 2. Se modifica a las nuevas preferencias
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	INTRODUCIR LOS RESULTADOS DE LA JORNADA	ID	11
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	ADMINISTRADOR
Personal involucrado	Administrador: Modifica los resultados de la jornada Anónimo: Ve el resulta de cada jornada
Descripción	Se inserta los resultados de la jornada
Trigger	
Precondición	
Postcondición	Los resultados se almacenan en el sistema. Se calcula la nueva clasificación.
Flujo Normal	1. Selecciona la jornada 2. Seleccionar el partido 3. Introducir marcador
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	INTRODUCIR RESULTADOS	ID	12
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	USUARIO REGISTRADO
Personal involucrado	Usuario registrado: Introduce los resultado del partido que ha jugado
Descripción	Se muestra el resultado del partido
Trigger	
Precondición	El partido ha finalizado
Postcondición	Se abre un periodo de modificación y aceptación de resultado
Flujo Normal	1. Se selecciona el partido disputado 2. Se introduce el resultado del mismo 3. El sistema informa al resto de los jugadores del partido del resultado introducido
Flujo alternativo	
Excepción	

Includes	
Requisitos especiales	
Notas	

Nombre	LISTAR CAMPEONATOS Y FILTRARLOS	ID	13
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	ANÓNIMO
Personal involucrado	Anónimo: Ve el listado de los campeonatos que han sido creados en el sistema
Descripción	Se muestran los campeonatos del sistema filtrados según condiciones
Trigger	
Precondición	
Postcondición	El sistema lista los campeonatos
Flujo Normal	4. Se selecciona la lista de campeonatos 5. Se muestran los campeonatos filtrados por condiciones de los mismos
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	MODIFICAR HORARIOS	ID	14
Creado por	Josué Caballero López	Fecha	06/12/12
Modif. por		Fecha Modif	

Actor Principal	ADMINISTRADOR
Personal involucrado	Administrador: Modifica los horarios generados por el sistema Anónimo: Ve la jornada con los horarios modificados
Descripción	Se modifican los horarios generados
Trigger	
Precondición	El horario ha sido generado
Postcondición	Se muestra el nuevo horario de la jornada
Flujo Normal	1. Se selecciona la jornada 2. Se muestra el horario seleccionado 3. Se realizan las modificaciones 4. Se almacena el nuevo horario de la jornada
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	REGISTRARSE EN EL SISTEMA	ID	15
Creado por	Josué Caballero López	Fecha	06/12/12
Modif. por		Fecha Modif	

Actor Principal	ANÓNIMO

Personal involucrado	Anónimo: Inserta sus datos personales, preferencias horarias y categoría
Descripción	El usuario se registra en el sistema
Trigger	
Precondición	
Postcondición	El usuario está registrado. Puede entrar en su área personal
Flujo Normal	1. Se selecciona registrarse 2. Se introduce sus datos personales y categoría 3. Se introduce sus preferencias horarias
Flujo alternativo	1a. Se inscribe en un campeonato
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	VALIDAR RESULTADOS	ID	16
Creado por	Josué Caballero López	Fecha	06/12/12
Modif. por		Fecha Modif	

Actor Principal	USUARIO REGISTRADO
Personal involucrado	Usuario registrado: Valida el resultado del partido
Descripción	El usuario valida el resultado del partido introducido por otro jugador
Trigger	
Precondición	El resultado del partido disputado ha sido introducido
Postcondición	El resultado se da por válido y cerrado
Flujo Normal	1. Se selecciona el partido jugado 2. Se valida el resultado introducido
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	VER CALENDARIO DE UN CAMPEONATO	ID	17
Creado por	Josué Caballero López	Fecha	06/12/12
Modif. por		Fecha Modif	

Actor Principal	ANÓNIMO
Personal involucrado	Anónimo: Ve el calendario del campeonato jornada por jornada y lugar
Descripción	Se muestra el calendario y lugar de los partidos de un campeonato
Trigger	
Precondición	
Postcondición	
Flujo Normal	1. Selecciona el campeonato que desea ver 2. Se muestra el detalle por jornada
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	

Notas	

Nombre	VER CALENDARIO DE LOS CAMPEONATOS EN LOS QUE PARTICIPA	ID	18
Creado por	Josué Caballero López	Fecha	06/12/12
Modif. por		Fecha Modif	

Actor Principal	USUARIO REGISTRADO
Personal involucrado	Usuario registrado: Lista sus campeonatos
Descripción	Ve los campeonatos en los que participa filtrados
Trigger	
Precondición	
Postcondición	
Flujo Normal	1. Se introduce en su área personal 2. Muestra sus campeonatos activos y finalizados
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	VER NOTICIAS	ID	19
Creado por	Josué Caballero López	Fecha	06/12/12
Modif. por		Fecha Modif	

Actor Principal	ANÓNIMO
Personal involucrado	Anónimo: Ve noticias genéricas sobre la aplicación y sobre campeonatos
Descripción	Se muestran noticias en la welcome page de la aplicación
Trigger	
Precondición	
Postcondición	
Flujo Normal	1. Entra en la página de la aplicación 2. Ve noticias sobre la aplicación o sobre campeonatos específicos
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

Nombre	VER RESULTADOS POR JORNADA	ID	20
Creado por	Josué Caballero López	Fecha	
Modif. por		Fecha Modif	

Actor Principal	ANÓNIMO
Personal involucrado	Anónimo: Ve resultados por jornada
Descripción	El anónimo puede ver resultados actuales y pasados de un campeonato jornada a jornada
Trigger	
Precondición	

Postcondición	
Flujo Normal	1. Lista los campeonatos según filtros 2. Selecciona el campeonato que desea ver 3. Ve los resultados de la jornada actual 4. Selecciona la jornada que desea ver 5. Ve sus resultados
Flujo alternativo	
Excepción	
Includes	
Requisitos especiales	
Notas	

14.4.Anexo D: Base de datos

Script de creación de la base de datos:

```
--
-- PostgreSQL database dump
--
--
-- Name: plpgsql; Type: EXTENSION; Schema: -; Owner:
--

CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg_catalog;
--
-- Name: EXTENSION plpgsql; Type: COMMENT; Schema: -; Owner:
--
COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';
SET search_path = public, pg_catalog;
SET default_tablespace = '';
SET default_with_oids = false;
--
-- Name: availability_schemas; Type: TABLE; Schema: public; Owner: root; Tablespace:
--
CREATE TABLE availability_schemas (
   availability_id integer NOT NULL,
   resource_id integer,
   mode character varying(40),
   availability character varying(250),
   remarks character varying(250),
   active integer,
   ini_date timestamp without time zone,
   end_date timestamp without time zone,
   user_insert_id integer,
   user_update_id integer,
   date_created timestamp without time zone,
   date_updated timestamp without time zone
);
CREATE SEQUENCE availability_availability_id_seq
   START WITH 1
   INCREMENT BY 1
   NO MINVALUE
   NO MAXVALUE
   CACHE 1;
--
```

```
-- Name: availability_availability_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: root
--

ALTER SEQUENCE availability_availability_id_seq OWNED BY availability.availability_id;


ALTER TABLE availability_schemas OWNER TO root;
--
-- Name: categories; Type: TABLE; Schema: public; Owner: root; Tablespace:
--
CREATE TABLE categories (
    category_id integer NOT NULL,
    competition_id integer,
    name character varying(40) NOT NULL,
    format character varying(40),
    min_players integer,
    max_players integer,
    size integer,
    groups integer,
    rounds integer,
    playoff integer,
    ini_inscription_date date,
    end_inscription_date date,
    user_insert_id integer,
    user_update_id integer,
    date_created timestamp without time zone,
    date_updated timestamp without time zone
);
ALTER TABLE categories OWNER TO root;
--
-- Name: categories_category_id_seq; Type: SEQUENCE; Schema: public; Owner: root
--
CREATE SEQUENCE categories_category_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE categories_category_id_seq OWNER TO root;

--
-- Name: categories_category_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: root
--

ALTER SEQUENCE categories_category_id_seq OWNED BY categories.category_id;

--
-- Name: clubs; Type: TABLE; Schema: public; Owner: root; Tablespace:
--
CREATE TABLE clubs (
    club_id integer NOT NULL,
    name character varying(40),
    location character varying(250),
    remarks character varying(250),
    user_insert_id integer,
    user_update_id integer,
    date_created timestamp without time zone,
    date_updated timestamp without time zone
);
```

```
ALTER TABLE clubs OWNER TO root;
--
-- Name: competitions; Type: TABLE; Schema: public; Owner: root; Tablespace:
--
CREATE TABLE competitions (
    competition_id integer NOT NULL,
    name character varying(40) NOT NULL,
    description character varying(120) NOT NULL,
    ini_date date,
    end_date date,
    user_insert_id integer,
    user_update_id integer,
    date_created timestamp without time zone,
    date_updated timestamp without time zone
);
ALTER TABLE competitions OWNER TO root;
--
-- Name: competitions_competition_id_seq; Type: SEQUENCE; Schema: public; Owner: root
--
CREATE SEQUENCE competitions_competition_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE competitions_competition_id_seq OWNER TO root;
--
-- Name: competitions_competition_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
root
--
ALTER SEQUENCE competitions_competition_id_seq OWNED BY competitions.competition_id;

--
-- Name: courts; Type: TABLE; Schema: public; Owner: root; Tablespace:
--
CREATE TABLE courts (
    court_id integer NOT NULL,
    club_id integer,
    name character varying(40),
    remarks character varying(250),
    user_insert_id integer,
    user_update_id integer,
    date_created timestamp without time zone,
    date_updated timestamp without time zone
);
ALTER TABLE courts OWNER TO root;
--
-- Name: games; Type: TABLE; Schema: public; Owner: root; Tablespace:
--
CREATE TABLE games (
    game_id integer NOT NULL,
    category_id integer,
    group_name character varying(40),
    registration_id_1 integer,
    registration_id_2 integer,
    registration_id_3 integer,
    registration_id_4 integer,
```

```
    round character varying(40),
    game_date timestamp without time zone,
    remarks character varying(250),
    points_1 integer,
    points_2 integer,
    points_3 integer,
    points_4 integer,
    location character varying(250),
    user_insert_id integer,
    user_update_id integer,
    date_created timestamp without time zone,
    date_updated timestamp without time zone
);
ALTER TABLE games OWNER TO root;
--
-- Name: registration_details; Type: TABLE; Schema: public; Owner: root; Tablespace:
--
CREATE TABLE registration_details (
    registration_detail_id integer NOT NULL,
    registration_id integer,
    player character varying(10)
);
ALTER TABLE registration_details OWNER TO root;
--
-- Name: registration_details_registration_detail_id_seq; Type: SEQUENCE; Schema: public; Owner:
root
--
CREATE SEQUENCE registration_details_registration_detail_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE registration_details_registration_detail_id_seq OWNER TO root;
--
-- Name: registration_details_registration_detail_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: root
--
ALTER SEQUENCE registration_details_registration_detail_id_seq OWNED BY
registration_details.registration_detail_id;
--
-- Name: registrations; Type: TABLE; Schema: public; Owner: root; Tablespace:
--
CREATE TABLE registrations (
    registration_id integer NOT NULL,
    category_id integer,
    user_insert_id character varying(10),
    date_created date
);

ALTER TABLE registrations OWNER TO root;
--
-- Name: registrations_registration_id_seq; Type: SEQUENCE; Schema: public; Owner: root
--
CREATE SEQUENCE registrations_registration_id_seq
    START WITH 1
    INCREMENT BY 1
```

```
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE registrations_registration_id_seq OWNER TO root;
--
-- Name: registrations_registration_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
root
--
ALTER SEQUENCE registrations_registration_id_seq OWNED BY registrations.registration_id;
--
-- Name: users; Type: TABLE; Schema: public; Owner: root; Tablespace:
--
CREATE TABLE users (
    user_id integer NOT NULL,
    name character varying(40) NOT NULL,
    surname character varying(120) NOT NULL,
    document character varying(10) NOT NULL,
    email character varying(40),
    phone integer,
    mobile integer,
    password character varying(128),
    date_created timestamp without time zone,
    date_updated timestamp without time zone,
    activo integer,
    shirt_size character varying(5)
);
ALTER TABLE users OWNER TO root;
--
-- Name: users_user_id_seq; Type: SEQUENCE; Schema: public; Owner: root
--
CREATE SEQUENCE users_user_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE users_user_id_seq OWNER TO root;
--
-- Name: users_user_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: root
--
ALTER SEQUENCE users_user_id_seq OWNED BY users.user_id;
--
-- Name: category_id; Type: DEFAULT; Schema: public; Owner: root
--
ALTER TABLE ONLY categories ALTER COLUMN category_id SET DEFAULT
nextval('categories_category_id_seq'::regclass);
--
-- Name: competition_id; Type: DEFAULT; Schema: public; Owner: root
--
ALTER TABLE ONLY competitions ALTER COLUMN competition_id SET DEFAULT
nextval('competitions_competition_id_seq'::regclass);
--
-- Name: registration_detail_id; Type: DEFAULT; Schema: public; Owner: root
--
ALTER TABLE ONLY registration_details ALTER COLUMN registration_detail_id SET DEFAULT
nextval('registration_details_registration_detail_id_seq'::regclass);
--
-- Name: registration_id; Type: DEFAULT; Schema: public; Owner: root
--
```

```
ALTER TABLE ONLY registrations ALTER COLUMN registration_id SET DEFAULT
nextval('registrations_registration_id_seq'::regclass);
--
-- Name: user_id; Type: DEFAULT; Schema: public; Owner: root
--
ALTER TABLE ONLY users ALTER COLUMN user_id SET DEFAULT
nextval('users_user_id_seq'::regclass);
--
-- Data for Name: availability_schemas; Type: TABLE DATA; Schema: public; Owner: root
--
--
-- PostgreSQL database dump complete
--
```