UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en Tecnologías de Telecomunicación

Trabajo Fin de Máster

Herramienta de programación y estimación del consumo energético para un Dispositivo Autónomo de Experimentación en Acuicultura

Autor: D. Rómel Alfredo Cabo Fernández

Tutor(es): Dr. D. Carlos Javier Sosa González
Dr. D. Juan Antonio Montiel Nelson

Fecha: Enero 2019

IUMA
t +34 928 451 150
+34 928 451 086
f +34 928 451 083
e: iuma@iuma.ulpgc.es
w: www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en Tecnologías de Telecomunicación

Trabajo Fin de Máster

Herramienta de programación y estimación del consumo energético para un Dispositivo Autónomo de Experimentación en Acuicultura

HOJA DE FIRMAS

Alumno/a: D. Rómel Alfredo Cabo Fernández Fdo.:

Tutor/a: Dr. D. Carlos Javier Sosa González Fdo.:
Tutor/a: Dr. D. Juan Antonio Montiel Nelson Fdo.:

Fecha: Enero 2019

IUMA
t +34 928 451 150
+34 928 451 086
f +34 928 451 083
e: iuma@iuma.ulpgc.es
w: www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en Tecnologías de Telecomunicación

Trabajo Fin de Máster

Herramienta de programación y estimación del consumo energético para un Dispositivo Autónomo de Experimentación en Acuicultura

HOJA DE EVALUACIÓN

Calificación: .....

Presidente Fdo.:

Secretario Fdo.:

Vocal Fdo.:

Fecha: Enero 2019

IUMA
t +34 928 451 150
+34 928 451 086
f +34 928 451 083
e: iuma@iuma.ulpgc.es
w: www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

Agradecimientos

Agradecer como estudiante extranjero proveniente de Cuba al gobierno español por darme la oportunidad de ampliar y progresar en estudios e investigación de posgrados.

Al Instituto Universitario de Microelectrónica Aplicada (IUMA) enmarcado en La Universidad de Las Palmas de Gran Canaria (ULPGC), por acogerme en el presente Máster en Tecnologías de Telecomunicación y brindarme un sin fin de conocimientos y habilidades.

A mis tutores Dr. D. Carlos Javier Sosa González y Dr. D. Juan Antonio Montiel Nelson por el apoyo, orientación y constancia en todo el proceso investigativo.

Profundo agradecimiento a mi gran familia por ser los pilares de mi formación personal y parte fundamental de mi desarrollo estudiantil.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

AJAX	Asynchronous JavaScript And XML
AMD	Advanced Micro Devices
ASD	Adaptive Software Development
AUP	Agile Unified Process
APIs	Application Programming Interface
ARC	Augmentation Research Center
CAD	Computer Aided Design
CDS	Comprehensive Display System
CLI	Command Line Interface
CRT	Cathode Ray Tube
CSS	Cascading Style Sheets
DAE	Dispositivo Autónomo de Experimentación
DARPA	Defense Advanced Research Projects Agency
DATAR	Digital Automated Tracking and Resolving
DOM	Document Object Model
DOS	Disk Operating Systems
DSDM	Dynamic Systems Development
FAT 32	File Allocation Table 32 bits
FDD	Feature-Driven Development
GNOME	GNU Network Object Model Environment
GNU	GNU is Not Unix
GPL	GNU General Public License
GS/OS	Graphics and Sound/Operating System
GTK+	GIMP ToolKit
GUI	Graphical user interface
HES	Hypertext Editing System
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IA	Inteligencia Artificial
IBM	International Business Machines Corporation
IDE	Integrated Development Environment
IPO	interacción persona-ordenador
ISO	International Organization for Standardization
IU	Interfaz de Usuario
JDK	Java Development Kit
JSD	Jackson System Development
JVM	Java Virtual Machine
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
LAMP	Linux-Apache-MySQL-PHP
LASER	Light Amplification by Stimulated Emission of Radiation
LD	Lean Development
LED	Light-Emitting Diode
LGPL	GNU Lesser General Public License
MEAN	MongoDB-AngularJS-ExpressJS-NodeJS
Memex	Memory-Index
MS	Microsoft
MS-DOS	Microsoft Disk Operating System
MVC	Model –View-Controller

MVVM	Model-View-ViewModel
MYSQL	My Structured Query Language
NASA	National Aeronautics and Space Administration
NLS	oN-Line System
NUI	Natural User interface
OMG	Object Management Group
OS/2	Operating System / 2
PARC	Palo Alto Research Center
PC	Personal Computer
POJO	Plain Old Java Object
POO	Programación Orientada a Objetos
PWA	Progressive Web Aplication
RAD	Rapid Application Development
RIA	Rich Internet Application
RTOS	Real-time operating system
RUP	Rational Unified Process
SDK	Software Development Kit
SDS	Scientific Data Systems
SGML	Standard Generalized Markup Language
SOS	Sophisticated Operating System
SRI	Stanford Research Institute
SSADM	Structured System Analysis and Design Method
Tcl	Tool Command Language
TCP/IP	Transmission Control Protocol/Internet protocol
TFM	Trabajo Fin de Master
TIOBE	The Importance of Being Earnest
Tk	ToolKit
TUI	Touch User Interface
UML	Unified Modeling language
UNIX	Uniplexed Information and Computing Service
WIMP	Windows, Icons, Menus, Pointer
WYSIWYG	What-You-See-Is-What-You-Ge
W3C	World Wide Web Consortioum
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
XNU	X is Not Unix
XP	Extreme programming
XUL	User Interface Markup Language
ZUI	Zooming User Interface
ZUML	ZK User Interface Markup Language

CAPÍTULO 1. Introducción

1.1 Antecedentes

En el contexto actual dentro del campo de la acuicultura se está invirtiendo considerables esfuerzos y dinero en mejorar la producción de diversas especies de interés industrial [1].

El conocimiento adquirido hasta la actualidad mediante los métodos tradicionales sobre por ejemplo los hábitos alimenticios y reproductivos, entre otros, ha llegado a un punto en el cual requieren la introducción de instrumental y su automatización.

En su mayoría, la implantación de sistemas electrónicos en la acuicultura con el objeto de mejorar la producción se circunscribe al control y gestión de las aguas y de los alimentos. También se hacen estudios genéticos de la población con el objeto de potenciar ciertas características fenotípicas de los individuos [2].

Sin embargo, a pesar de las numerosas innovaciones que se han producido en ámbito del diagnóstico médico a humanos que se ha trasladado a diversos ámbitos del terreno veterinario (perros, gatos, aves, tortugas y vacas entre otros), dichos progresos no han tenido traslado al área de la acuicultura.

Es evidente que el principal problema al aplicar esas innovaciones en la acuicultura es el propio medio en el que se desarrolla, que en se puede calificar de hostil para la inmensa mayoría de los dispositivos electrónicos.

El área de la acuicultura es uno de los tópicos de interés prioritario planteado en el seno de la Unión Europea. En ese sentido se han dispuesto diversas líneas de financiación para potenciar los avances en dicho terreno. Enmarcado en este contexto surge el proyecto europeo AquaExcel2020 [3], el cual tiene dentro de sus Paquetes de Trabajo (Work Package 8) que propone el desarrollo de un dispositivo autónomo de experimentación (DAE) que permita determinar la actividad de distintas especies de interés para la industria de la acuicultura.

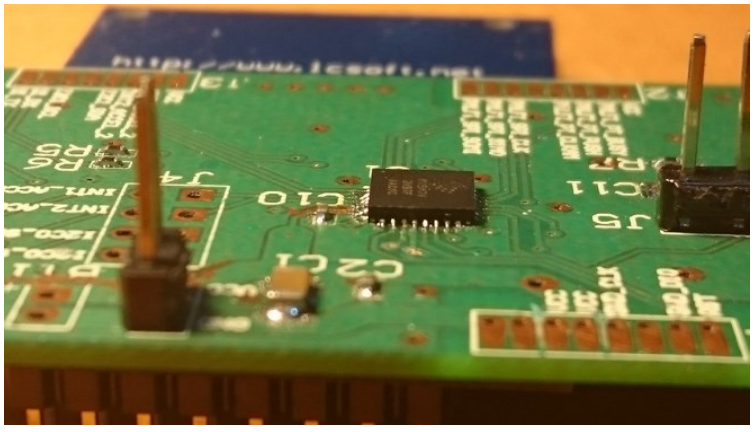


Figura 1. Vista parcial del Prototipo v0 del dispositivo autónomo de experimentación

El DAE está compuesto de un microcontrolador de ultra bajo consumo alimentado por una batería y al menos un acelerómetro y un termómetro en su prototipo

inicial. La gestión de la energía se torna un punto crítico si a lo expuesto se añade que los experimentos a ejecutar como por ejemplo determinar la actividad de un individuo se ha de adecuar a distintos tipos de especies y hábitos.

La versatilidad del DAE le permite a un usuario programar la ejecución temporizada de un conjunto de primitivas de medida y operaciones matemáticas básicas sobre dichas medidas como por ejemplo obtener el valor medio, desviación estándar, máximo y mínimo, las cuales conforman el llamado experimento. En estas condiciones es necesario determinar el horizonte temporal que cubrirá el experimento definido.

1.2 Objetivos

Este Trabajo Fin de Máster (TFM) plantea como objetivo principal: Desarrollar una herramienta software que permita describir gráficamente el experimento a desarrollar en base a primitivas y su temporización, así como evaluar el impacto en términos de duración (horizonte temporal) del experimento especificado por un operario del dispositivo autónomo de experimentación.

Este objetivo principal consta de los siguientes objetivos parciales:

1. Desarrollar una interfaz (XML o similar) que permita abstraer los detalles de implementación de las primitivas de medida y/o cómputo y su temporización (nivel de desarrollador) del nivel de usuario.
2. Implementar una interfaz gráfica de usuario (GUI) que permita programar el dispositivo de experimentación autónoma de forma intuitiva y con conocimientos mínimos de programación (algorítmica) [4].
3. Desarrollar un algoritmo que permita estimar tanto el consumo energético como el horizonte temporal de ejecución del experimento propuesto.

1.3 Estructura del documento

El documento se ha estructurado en 7 capítulos, describiéndose las diferentes etapas de investigación y cumplimiento de los objetivos planteados para el desarrollo de la "Herramienta de programación y estimación del consumo energético para un Dispositivo Autónomo de Experimentación en Acuicultura".

1- El capítulo 1 nos introduce en la línea de investigación en la cual está enfocado el presente TFM, se exponen los objetivos que se pretenden materializar y la forma en que se organizó la documentación.
2- El segundo capítulo describe el estado del arte sobre las diversas herramientas existentes para crear interfaces gráficas de usuario, discriminando y seleccionando la más adecuada para el cumplimiento de los objetivos propuestos.
3- La planificación y metodología son expuestas en el capítulo 3, dando una visión de la manera en que se programó y llevo cabo el proceso investigativo.
4- En el cuarto capítulo se detallan todos los recursos y herramientas empleadas tanto de hardware, software y económicos, que permitieron y facilitaron el desempeño del TFM.
5- El capítulo 5 titulado "Desarrollo de la herramienta software" es el que da cumplimiento al objetivo principal y a los parciales del presente trabajo. Se aplica

la metodología Rational Unified Process (RUP), definiendo los requisitos del sistema con el modelo de dominio y la lista de características, los requisitos software con los actores y casos de usos. Además, se plantea el prototipo de GUI, patrón de diseño a desarrollar y la arquitectura del sistema de la aplicación web. Se muestra el potencial de las herramientas, lenguajes y librerías empleadas, así como la complementación de estas en un mismo entorno de desarrollo.

6- Los resultados y validación del proyecto se exponen en el capítulo 6, con la puesta en marcha de la herramienta desarrollada, examinándose y comprobándose de manera intuitiva por usuarios con un mínimo de conocimientos tecnológicos.

7- A modo de conclusión se creó el capítulo 7 donde se ratifica el cumplimiento de los objetivos inicialmente expuestos y el futuro de los resultados obtenidos.

CAPÍTULO 2. Estado del arte

La interfaz gráfica de usuario por sus siglas en inglés GUI (Graphical User Interface) se puede definir como la plataforma o programa informático que contiene un dispositivo electrónico y es la interfaz con la que se interrelaciona el usuario de manera visual, auditiva, manual, mediante voz u otras formas con dicho dispositivo. Su origen, está en la evolución de las CLI (Command Line Interface) o interfaces de línea de comando que se utilizaban en los primeros sistemas operativos; es el resultado de la transición del lenguaje escrito al visual en el ámbito informático. La interfaz permite que el usuario interactúe gráficamente con las funcionalidades disponibles de sus elementos o componentes (iconos, ventanas imágenes, textos, archivos y otros objetos gráficos). Muchas GUIs se definen por el paradigma WIMP (Windows, Icons, Menus, Pointer) como por ejemplo el uso de ordenadores mediante dispositivo apuntador (ratón) y teclado; otras como las incorporadas en los actuales móviles, tabletas, portátiles, etc se emplean a través de TUI (Touch User Interface) donde el usuario se relaciona mediante la pantalla táctil del dispositivo con la interfaz gráfica. El incremento y desarrollo de las interfaces naturales de usuarios NUI (Natural User interface) implementadas en GUIs mediante los asistentes de voz o dispositivos perceptivos de movimientos, vibraciones y otros han proporcionado un sin número de funcionalidades, sin hacer uso de dispositivos físicos intermedios en la interacción con la interfaz. Además, la mayoría de GUIs disponen de interfaz de usuario con zoom ZUI (Zoom User Interface), permitiendo ampliar o reducir la escala del contenido gráfico desplazándonos en dos direcciones. En la actualidad la mayoría de los objetos de uso cotidiano en los hogares, negocios, oficinas, entorno industrial, automóviles y otros ámbitos poseen una interfaz gráfica cada vez más evolucionada y con tendencia al uso de pantallas táctiles, aumentando la interactividad entre los usuarios y los dispositivos [4]-[7].

2.1 Historia de la interfaz gráfica de usuario.

El origen de la interfaz gráfica de usuario surge de la contribución cronológica de investigadores, los avances de la electrónica, los institutos tecnológicos, universidades e instituciones del ámbito militar, aeroespacial y naval, en el contexto de las guerras mundiales, la defensa militar de los países más desarrollados y el crecimiento industrial.

Se toma como punto de partida el concepto del escritorio o biblioteca de base de datos para almacenar y buscar documentos, libros y comunicaciones denominado “Memex”, diseñado por Vannevar Bush y publicado en 1945 en su artículo “As We May Think” de la revista Atlantic Monthly. Memex es el acrónimo de Memory-Index y como se muestra en la figura 2, representa un escritorio con palancas y teclado para la búsqueda de información contenida en micro filmes, que mediante un mecanismo electromecánico se proyectaba dicha información en pantallas. Los documentos podrían ser elaborados, modificados y enlazados por asociación de un mismo tema de estudio, utilizando un panel para escribir y fotografiar los mismos. Este proceso de creación se podría almacenar en los micro filmes de ultra alta resolución, constituyendo una biblioteca de diferentes documentos y libros, posibilitando que algunos elementos de un artículo se vinculen a otros que podrían estar

interrelacionados. Además, Memex daba la posibilidad de dictado y grabación voz, mostrándola de manera escrita en pantalla. Su proyección se enfoca en una biblioteca virtual, donde la lectura por el usuario activo implica la escritura directa sobre el documento de una forma natural, simulando un documento real [8]- [10]. Este concepto no se hizo tangible, pero sin dudas fue un precursor para la invención del hipertexto e hipervínculo por Ted Nelson en el proyecto Xanadu y Andries van Dam desarrollando el sistema HES (Hypertext Editing System), la World Wide Web por Tim Berners-Lee y Robert Cailliau [11]- [15]. También la Wikipedia, enciclopedia en línea más grande del mundo creada por Jimmy Wales y Larry, el procesamiento de textos, los ordenadores personales y otros [16].

La conceptualización del Memex despertó en Ivan Edward Sutherland nombrado por muchos como el padre de la computación gráfica, la idea de crear en su tesis doctoral en 1962 la primera herramienta informática de diseño gráfico denominada Sketchpad. Este primario prototipo de interfaz gráfica de usuario empleaba en su tiempo la avanzada computadora Lincoln TX-2, con un monitor de tubo de rayos catódicos CRT (Cathode Ray Tube), un cuadro de botones, cintas perforadas y magnéticas para el almacenamiento y un lápiz óptico como dispositivo señalador para dibujar, manipular, borra, hacer zoom, duplicar, hacer restricciones de propiedades geométricas y otras funciones en tiempo real con los objetos gráficos (líneas, puntos, circunferencias, arcos, objetos 2D y 3D, etc). Los objetos se guardaban en memorias estructuradas por tipo, denominados “Maestro” e instancias dinámicas de los mismos, contribuyendo al origen de la futura POO (Programación Orientada a Objetos) como posteriormente comento Alan Kay. Demostró en su presentación la importancia de la comunicación visual e interacción persona-ordenador (IPO) y un punto de partida en el diseño asistido por computadora CAD (Computer Aided Design). Posteriormente fue muy utilizado en diseños de planos industriales, eléctricos, mecánicos, dibujos técnicos, diagramas de flujo y en la rama artística para la creación de animaciones. La figura 2 ilustra el día de su presentación en el laboratorio Lincoln del MIT (Massachusetts Institute of Technology) [17]- [25].

Figura 2. Primeros prototipos de interfaces gráficas de usuario [8], [24], [29]

Durante la segunda guerra mundial el visionario Douglas Carl Engelbart fue reclutado por la marina estadounidense y operaba como técnico de radares en Filipinas. Allí estudió en profundidad el artículo del Memex, el cual lo cautivo y en su regreso a la universidad valoraba la idea de hacer un mundo mejor con el uso de ordenadores vinculados al esfuerzo colectivo y organizado de todos, en el aprovechamiento y optimización del intelecto humano para resolver con rapidez los crecientes problemas de la humanidad.

Posteriormente en el instituto de investigación de Stanford redactó un informe con su visión y todos sus pensamientos reunidos, titulado “Aumentando el Intelecto Humano: Un Marco Conceptual” traducido al español. Influenciado por el Memex de Vannevar Bush y el Sketchpad de Sutherland, se dio a la tarea de materializar sus ideas con su equipo de trabajo en el ARC (Augmentation Research Center) del SRI (Stanford Research Institute), apoyado por DARPA (Defense Advanced Research Projects Agency) y la NASA (National Aeronautics and Space Administration). La figura 2 ilustra una instantánea del oN-Line System (NLS) presentada por Douglas C. Engelbart durante 90 minutos en la Conferencia Conjunta de Computación de Otoño en San Francisco en diciembre de 1968, la cual se titulaba “A RESEARCH CENTER FOR AUGMENTING HUMAN INTELLECT” y fue denominada posteriormente como la madre de todas las demostraciones. El equipo de trabajo y el revolucionario sistema de colaboración informática en línea de hardware y software fue distribuido geográficamente el día de su presentación, donde Engelbart desde la conferencia tenía conectado sus periféricos a módems enlazados con líneas arrendadas bidireccionales para la transferencia de datos y desde los cuales él controlaba el ordenador diseñado para sistemas de tiempo compartido entre múltiples usuarios, nombrado SDS (Scientific Data Systems) 940 que se encontraba a 30 millas en el ARC de Menlo Park California. Durante la presentación se transmitía videos de forma bidireccional por dos enlaces de microondas entre la conferencia y el ARC, además de las numerosas cámaras instaladas en ambos sitios. Demostrando las funcionalidades del sistema y con la coordinación técnica de todos sus elementos a cargo de William K. English se pudo ver en una única pantalla frente al público, la superposición de videoconferencias de los integrantes del equipo que se encontraban en el ARC y las operaciones que realizaba Engelbart en el ordenador SDS 940 con el novedoso dispositivo señalador “mouse”, teclado estándar y un conjunto de 5 teclas especiales para generar caracteres y comandos. Adelantado para la su época, se observaron muchas características y elementos de la computación moderna, mapas de bits gráficos, videoconferencia, uso de hipertexto con hipervínculos, la hipermedia, procesamiento de textos, ventanas en pantalla, red de computadoras, software multiusuario, edición colaborativa de documentos en tiempo real, el acoplamiento de avanzadas tecnologías de video y antecedentes de la GUI [26]- [38].

Figura 3. Primer dispositivo apuntador (ratón) [32]

El indicador de posición X-Y para un dispositivo de visualización, comúnmente nombrado por su aspecto como “ratón” de ordenador fue diseñado por Douglas Engelbart y confeccionado por William K. English en el SRI. Ambos no estaban de acuerdo en utilizar

los pesados lápices de luz, los voluminosos joysticks o los teclados porque en su concepto de uso práctico del ordenador, expresaban que eran ineficientes. Como se muestra en la figura 3 el primer prototipo estaba construido en madera con un botón rojo, un cable para conectarlo al ordenador y en su parte inferior tenía dos ruedas metálicas con potenciómetros analógicos para el movimiento en ambos ejes. Se concibió como el intermediario directo entre el usuario y el ordenador, siendo en su uso más fácil e intuitivo que el teclado. En la presentación del NLS se utilizó el modelo oficial con un diseño más avanzado, con tres botones en su parte superior y construido con una carcasa en plástico y metal [38]- [40].

Anteriormente en el 1946 había sido inventado el "trackball" por Ralph Benjamin que serbia a la Marina Real Británica (Royal Navy). Este era un dispositivo apuntador que formaba parte del sistema de trazado en radares denominado CDS (Comprehensive Display System) y se mantuvo oculto como secreto militar. Luego fue adaptado por la Marina Real Canadiense con el nombre de DATAR (Digital Automated Tracking and Resolving). El prototipo de dispositivo utilizaba cuatro discos para capturar los movimientos, empleando dos discos para el eje X e Y respectivamente, una bola de 5 pines de bolos canadienses y a diferencia del prototipo ingles se utilizaba un ordenador digital. También fue un secreto militar y no tuvo evolución [40]- [46].

El trackball fue la inspiración de la compañía alemana Telefunken para desarrollar y comercializar por primera vez el Rollkugel, un dispositivo señalador con bola de goma en su interior que funcionaba solo en los sistemas de la compañía [47]- [49].

William K. English en 1971 abandona el SRI y forma parte del centro de investigación en Palo Alto nombrado Xerox PARC. Allí mejora el diseño concebido junto a Engelbart y sustituye las ruedas por la esfera nombrándolo ratón de bola (Ball Mouse) e introduce las ruedas de X e Y en su interior. Su diseño era como el trackball invertido, inspirado en el Rollkugel y su uso fue extendido durante muchos años, tanto en mouse como en diferentes dispositivos. El primer ordenador moderno en usar este "ratón" con la interfaz gráfica fue la Xerox Alto. Luego la afamada Xerox Star fue el primer sistema comercial que incluía entre tantas novedades el mouse y su uso en una completa interfaz gráfica de usuario [50]- [53].

La difusión de este dispositivo se popularizó mundialmente por empresas como Apple y Microsoft que abarataron su costo y lo transformaron a conveniencia de sus sistemas y apariencias como por ejemplo el número de botones. Transcurriendo los años hasta la actualidad se sucedieron tecnologías alternativas y nuevas empresas que modificaron este dispositivo adicionando otras funcionalidades como la rueda de "scroll", botones para "gaming", giroscopios, la sustitución de la bola por objetivos ópticos como LED (Light-Emitting Diode) infrarrojo y posteriormente LASER (Light Amplification by Stimulated Emission of Radiation), mejorando cada vez más en precisión. Además, surgieron nuevos conceptos como el uso de panel táctil "trackpad" o "touchpad", los multi-touch y también una tendencia de uso del dispositivo de forma inalámbrica [53]- [59].

Xerox PARC por su anterior nombre, es una empresa de investigación y desarrollo perteneciente a Xerox Company, actualmente se denomina PARC (Palo Alto Research Center). Muchos de los conceptos de Engelbart y su equipo de trabajo se

materializaron en esta empresa, de hecho, algunos de sus integrantes formaron parte de Xerox PARC como el antes mencionado William K. English y otros. Esta institución es la cuna de relevantes creaciones en el ámbito de la informática, los ordenadores y tecnologías asociadas a estos, que cambiaron y aportaron en gran medida al desarrollo de las computadoras personales de nuestro mundo [60]- [62].

Fue aquí donde se desarrolló Xerox Alto la primera computadora con GUI, teniendo el concepto de que las personas estuvieran familiarizadas con su entorno y el uso fácil de la interfaz, reconociéndose con rapidez y sin ninguna capacitación previa. La interfaz gráfica era una metáfora de un escritorio común, con documentos, reloj, calculadora, etc. Alto permitía tener activos varios programas a la vez contenidos en ventanas "windows" que podían superponerse bloqueando los elementos detrás de estas. La GUI se diseñó con la creación del paradigma WIMP donde se introdujo las ventanas, los iconos, menús y el puntero. Se utilizó la programación dirigida por eventos, con bloques de código para la construcción de gráficos como botones, casillas de verificación, controles deslizantes, pestañas y otros. El ordenador no se comercializó, fue utilizado como herramienta dentro de la compañía y se entregaron a laboratorios de diferentes universidades. Posteriormente se fue refinando el hardware y software, culminando en el sistema Xerox Star. Con Star se extendió la metáfora de escritorio, representándose gráficamente los archivos como documentos que podrían almacenarse en carpetas y llevando al usuario a un nivel superior de abstracción. El nuevo sistema poseía avanzadas herramientas para la creación de textos y gráficos, con la introducción de los términos y acciones de copiar, pegar y cortar. La compañía Xerox que fue pionera en máquinas fotocopiadoras, ya tenía una herramienta revolucionaria para fomentar su principal producto. Para ello elaboraron el concepto de "lo que vez, es lo que obtienes" WYSIWYG (What-You-See-Is-What-You-Ge), donde una impresión en papel se observaría igual a la imagen de la interfaz gráfica en la pantalla del ordenador. Xerox Star no transcendió a las masas por su elevado precio, pero sin dudas tenían características muy avanzadas a su época [52], [53], [63]- [67].

Los sistemas de Xerox estaban dotados de tecnologías innovadora y pioneras que ellos mismo habían creado y otras habían sido heredadas y perfeccionadas de NLS como, por ejemplo: La red de área local Ethernet, una completa y moderna GUI con ventanas e iconos, control de GUI por ratón, uso de teclado rediseñados, gráficos de mapas de bits desarrollados en ordenador, editores de texto WYSIWYG, impresoras láser y otros. Además, su grupo de investigadores con figuras relevantes como Alan Kay, crearon la programación orientada a objetos, el lenguaje de programación Smalltalk, el entorno de desarrollo integrado, la programación basada en prototipos, la arquitectura de software modelo-vista –controlador y otras [63]- [67].

Alan fue alumno y colaborador de Ivan Sutherland en el Skethpad. Participó como espectador en la demostración del NLS en la madre de todas las presentaciones de Engelbart, incentivando sus conocimientos y alentándolo a seguir con su carrera. Alan Kay es conocido como el padre conceptual de la interfaz gráfica de usuario y una figura relevante en el desarrollo de la programación orientada a objetos, la creación del lenguaje Smalltalk dio origen a posteriores lenguages como Java, Rubi, Objective-C, Python y otros. El Dynabook fue un concepto enfocado a los niños, que no logró ejecutar en sus días, pero inspiro a muchos en la evolución de los Tablet PC, ordenadores portátiles y dispositivos para da educación infantil de países en desarrollo. [68]- [76].

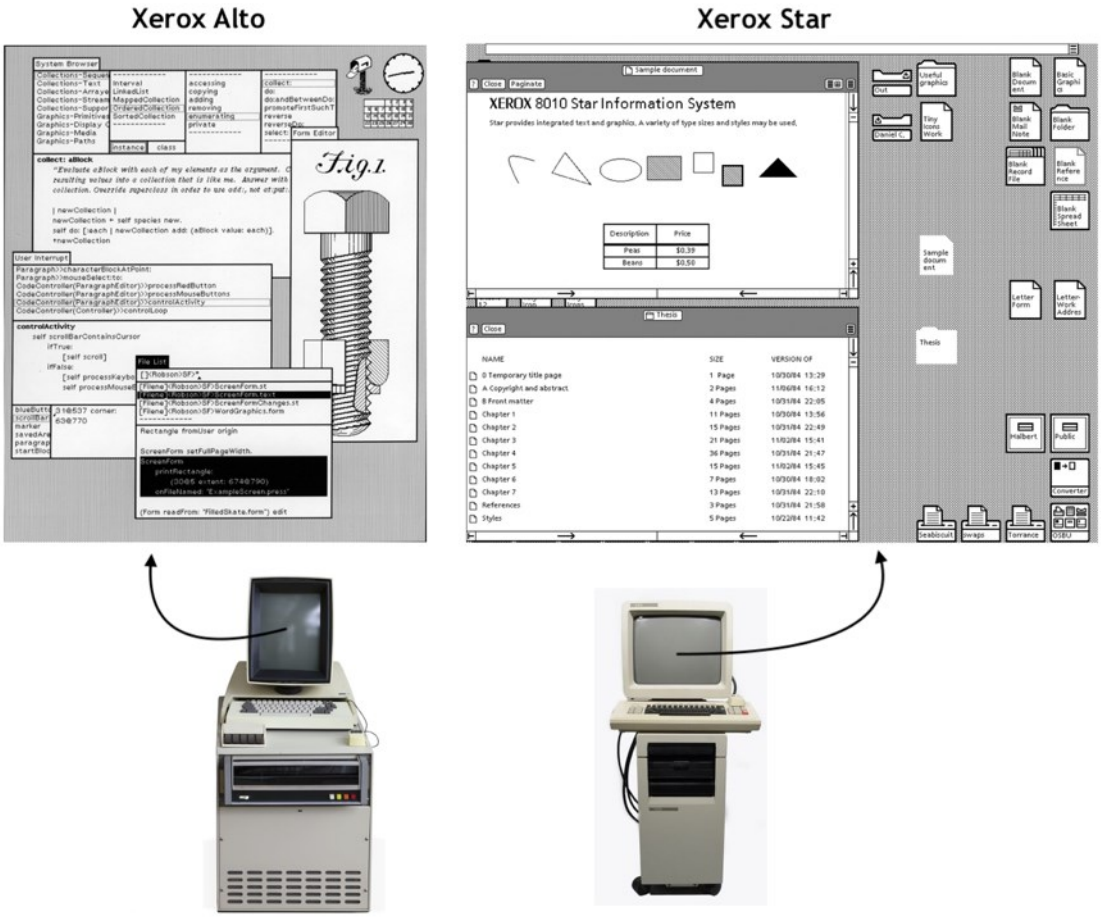


Figura 4. Interfaces gráficas de usuario de los ordenadores Xerox Alto y Xerox Star [77], [78], [77], [80]

Apple Inc. es una compañía multinacional norteamericana con sede en California, fundada por Steve Paul Jobs y Steve Wozniak. Cuenta en su recorrido hasta la actualidad con numerosos avances tecnológicos desarrollados en el ámbito del hardware y software, marcando siempre un antes y un después con conceptos muy novedosos en cuanto a diseño y funcionalidad. Las primeras GUIs desarrolladas como sistemas operativos para los ordenadores personales Apple I, II y III comienza 1976 con Apple DOS (Disk Operating Systems) un sistema de líneas de comando en los primeros años, seguido de SOS (Sophisticated Operating System), ProDOS y GS/OS (Graphics and Sound/Operating System). La línea de ordenadores Apple Lisa I y II utilizaban los sistemas Lisa OS (Office System). El microcomputador Apple Lisa incorporó las novedosas y revolucionarias ideas tomadas de Xerox PARC añadiendo una GUI mejorada, ratón, teclado, mapa de bits y la metáfora de escritorio en un ordenador muy compacto en su época. En el área de los ordenadores Macintosh se desarrolló a partir de 1984 un sistema operativo, triunfante nombrado Macintosh "System 1" o actualmente Mac OS "clásico". Este ordenador personal y portable podía ser adquirido por un precio más razonable que sus antecesores y por

primera vez se incluía una completa GUI sin utilizar líneas de comando, intuitivo al uso de cualquier tipo de usuario, con ratón, teclado, editor de textos, diseño gráfico, hojas de cálculo, etc. Este marcó el comienzo de toda la serie de sistemas denominados Mac OS hasta la llegada de Mac OS X basado en UNIX (Uniplexed Information and Computing Service) con kernel (nucléo) XNU (*X is Not Unix*) luego OS X y actualmente desde el año 2016 se denomina macOS en su versión actual macOS Mojave versión 10.14.1. La siguiente figura muestra la evolución de sus GUIs hasta la actualidad [81]- [101].

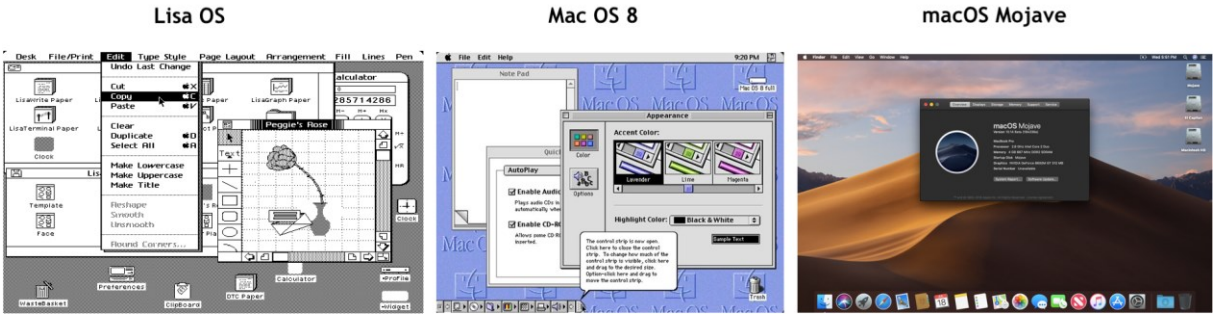


Figura 5 Evolución en Apple Inc. de la interfaz gráfica de usuario [102], [103], [104]

El sistema Mac OS X es la base de los sistemas en el resto de dispositivos de la plataforma de Apple Inc. El sistema iOS para dispositivos móviles iPhone, iPad y iPod, watchOS para los relojes inteligentes y tvOS para los Apple Tv. En la próxima ilustración se muestran los sistemas antes mencionados [105]- [108].

Figura 6. Modernas GUIs en otros dispositivos de Apple [109], [110], [111]

La multinacional norteamericana **Microsoft Corporation** (MS) fue fundada por William Henry Gates y Paul Allen. Es una compañía líder en la industria del software para ordenadores personales, servidores, electrónica de consumo y servicios. Sus principales aportes en el desarrollo y evolución de las GUIs comienzan con XENIX un sistema operativo como variante de UNIX que contaba con la primera versión del procesador de textos Microsoft Word. El primer gran éxito de la compañía viene con DOS adoptado en la plataforma de IBM PC y luego MS-DOS (Microsoft Disk Operating System) muy utilizado en las décadas de los 80 y 90 basados en líneas de comando. Junto a IBM (International Business Machines Corporation) desarrollaron OS/2 (Operating System / 2), que en años posteriores Microsoft dejaría. Su transcendente línea de sistemas operativos

Microsoft Windows para ordenadores PC (Personal Computer) comienza en 1985 con su versión 1.0 como extensión gráfica de MS-DOS, con GUI para la plataforma de PC. Sus próximas versiones serian Windows 2 con una refinada GUI, Windows 3 alcanzando el éxito comercial. Windows 95 fue un sistema operativo muy popular, éxito en el mercado, incorporando el botón inicio, la barra de tareas, el navegador Internet Explorer, el sistema de archivo FAT 32 (File Allocation Table 32 bits), funciones Plug and Play, sustituyo a MS-DOS como sistema operativo y a Windows 3 como entorno gráfico. Windows 98 fue una versión mejorada de su antecesor. Luego se crearon Windows XP con arquitectura de Windows NT y la primera activación de producto contra la piratería. Windows Vista tenía un novedoso estilo visual Aero, un nuevo diseño de GUI y tenía varios complementos de seguridad. Windows 7 fue una actualización incremental de Vista, con GUI rediseñada para su uso en interfaces táctiles. Windows 8 sustrae el botón inicio, tiene alta compatibilidad con interfaces táctiles e incorpora el concepto de Modern UI que era conocida como Metro. Windows 10 está vigente actualmente como parte aun de la familia de sistemas operativos Windows NT, su principal característica es aglutinar la funcionalidad y experiencia de usuarios entre las diferentes clases de dispositivos, además introduce el nuevo navegador Edge. La suite ofimática Microsoft Office es uno de los logros más importantes de la compañía, presente en todas las plataformas y muy usado en todos los ámbitos de ofimática durante décadas. Son creadores de Internet Explorer actual Edge se ha destacado como uno de los navegadores de internet más utilizados de la historia. La siguiente figura ilustra la evolución de los sistemas operativos de la compañía desde su comienzo hasta nuestros días [112]- [133].

Figura 7. Evolución de la interfaz gráfica de usuario en Microsoft [134], [135], [136]

Los hardware más destacados de Microsoft son las consolas de videojuegos Xbox y los ordenadores personales y portátiles Microsoft Surface con pantalla táctil. En el mundo de los móviles está presente con su sistema Windows Mobile adoptado por marcas como Nokia. A continuación, se muestran las GUI implementadas en los dispositivos antes mencionados [137]- [140].

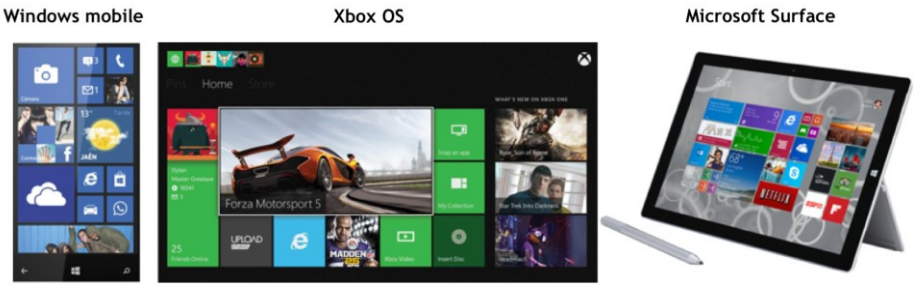


Figura 8. GUIs en otras plataformas de Microsoft [141], [142], [143]

El sistema operativo **GNU/Linux** es multiplataforma, de código abierto, gratuito, tipo UNIX, multitarea, etc. Se compone de dos importantes proyectos de desarrollo de software libre nombrados GNU (*GNU's Not Unix*) con la creación del sistema operativo y Linux quien proporciona el núcleo, muy similar a Unix. Su código fuente puedes ser utilizado bajo licencia GPL (GNU General Public License) por cualquier persona o institución. A estos sistemas y sus actualizaciones se le denominan distribuciones o distros. Son muy utilizados en superordenadores, servidores, videoconsolas, ordenadores personales, sistemas embebidos, por los desarrolladores de software y otros. Entre las distribuciones más populares podemos encontrar CentOS, Debian, GNOME, Dragora, Fedora, Gentoo, Knoppix, Kubuntu, Linux Mint, openSUSE, PCLinuxOS, Red Hat Enterprise Linux, Slackware, Tuquito, Trisquel, Ubuntu, etc. A continuación, se muestra la evolución de algunas de sus GUI hasta la actualidad [144]- [151].

Figura 9. Evolución de la interfaz gráfica de usuario en GNU/Linux [152], [153], [154]

Muchas GUIs de los dispositivos con los que interactuamos en la actualidad contienen sistemas basados en el núcleo de Linux. Android es uno de ellos, su sistema operativo está presente en una gran variedad de dispositivos móviles con pantalla táctil, televisores, relojes, automóviles, etc. Estos son fabricados por diversas marcas registradas que han logrado masificar el uso de este sistema a nivel mundial. Wear OS es el nombre actual del sistema operativo basado en Android de muchos relojes inteligentes, su nombre inicial fue Android Wear. La GUI para televisores basado en Linux denominada webOS, es un sistema multitarea, que incorpora tecnología web HTML5, CSS y JavaScript en su interfaz gráfica. Este sistema tiene una tendencia al uso con el asistente de Google mediante lenguaje natural, evolucionando cada día con el creciente desarrollo a la inteligencia artificial. En la siguiente figura se exponen los sistemas anteriormente expuestos [155]- [160].

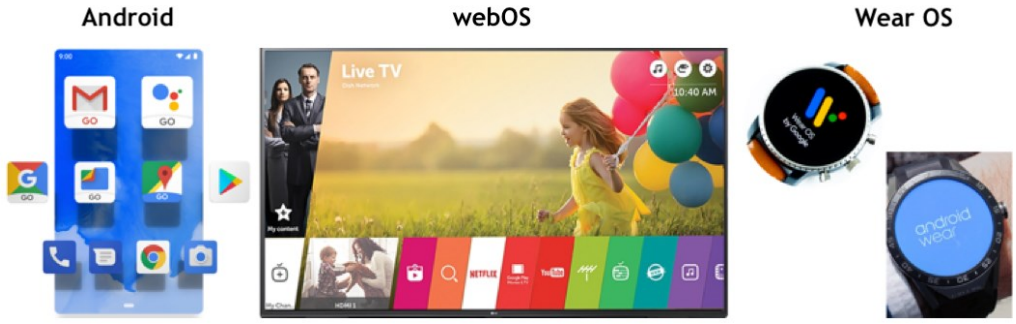


Figura 10. Interfaces gráficas de usuario en otras plataformas GNU/Linux [161], [162], [163], [164]

2.2 Herramientas para el desarrollo de GUIs

A día de hoy contamos con numerosas tecnologías y combinaciones de las mismas para el desarrollo de una aplicación software con una atractiva interfaz gráfica de usuario y compatible en todas las plataformas. Se ha realizado un estudio enfocado en las diversas herramientas informáticas para la creación de una GUI con características que permitan en particular crear, arrastrar, soltar y eliminar objetos gráficos en pantalla. Todo ello con el fin de reducir la curva de aprendizaje del usuario de la herramienta propuesta en este TFM.

2.2.1 Escritorio

Las herramientas para desarrollar GUI nativas difieren para cada sistema operativo, tomándose como referencia los tres más utilizados en la actualidad (Windows, macOS y GNU/Linux). Lo ideal sería emplear programas y lenguajes multiplataforma para optimizar el tiempo de los desarrolladores y aumentar la productividad en el ámbito. A continuación, se caracterizan algunos de ellos.

El framework multiplataforma y de código abierto **Electron,** utiliza tecnologías web para el desarrollo de aplicaciones nativas de escritorio para los sistemas Microsoft Window, macOS y GNU/Linux. Su funcionamiento se basa en el uso de HTML, CSS y JavaScript con APIs (Application Programming Interface) nativas para la acceder a las diferentes plataformas. La combinación de un subconjunto de librerías de Chromium en el frontend y NodeJS en el backend, posibilitan la creación de un solo ejecutable para cada sistema. Facebook, Microsoft y otras empresas emplean este framework en muchas de sus aplicaciones como Skype, WordPress, Visual Studio Code y otros. GitHub actualmente es quien da soporte al mismo y fue creado por Cheng Zhao [165], [166].

La biblioteca de código abierto y multiplataforma **wxWidgets** permite desarrollar con un solo código aplicaciones para todas las plataformas de sistemas

operativos existentes. La programación se realiza principalmente con C++, pero cuenta con enlaces para Ruby, Python, Java, Perl y otros. Con este kit de herramientas se obtienen GUIs muy similares a las nativas, basándose en las APIs nativas de macOS, Microsoft Windows, iOS, GNU/Linux, entre otras. Se puede usar wxWidgets libremente para proyectos gratuitos y comerciales bajo la licencia wxWindows como antiguamente se llamaba esta biblioteca y es similar a LGPL (GNU Lesser General Public License). Su uso es extensible en el sector empresarial, industrial, educacional y tecnológico como la NASA, AMD (Advanced Micro Devices), Xerox, etc [167], [168].

Como parte del proyecto GNU y Fundación GENOME se origina el kit de herramientas para el desarrollo de GUIs multiplataforma **GTK+** (GIMP ToolKit)**.** Su arquitectura se constituye de 4 bibliotecas denominadas Glib para bajo nivel con C, Pango para la internacionalización de textos y manejo de fuentes, Cairo para gráficos 2D y ATK para interfaces que proporcionan accesibilidad. Admite lenguajes como Java, Python, JavaScrip, C/C++, Rubi y etc. Junto a la herramienta GLADE aumenta la productividad en el proceso RAD (Rapid Application Development). Es de código abierto y gratuito bajo licencia LGPL 2.1, permitiendo desarrollar software libres y comerciales. Por sus características y rendimiento este kit es muy utilizado en aplicaciones de escritorio de GNOME (GNU Network Object Model Environment), en instituciones estudiantiles, sistemas de software de audio, en la bioinformática y sobre todo es muy conocido por el editor de imágenes GIMP [169], [170].

El software **LabVIEW** (Laboratory Virtual Instrument Engineering Workbench) es propiedad de National Instrument y está disponible para todos los sistemas operativos. Es una plataforma de ingeniería con un entorno de desarrollo para el diseño de sistemas, empleando el lenguaje de programación visual gráfico G. Su funcionamiento radica en un panel frontal como interfaz de usuario, para en su ejecución poder interactuar con sus funcionalidades. Además, un diagrama de bloques donde se crean, interconecta y programan los elementos del sistema y su funcionalidad. Su enfoque de programación gráfica permite una optimización del tiempo del programador, ya que tienen un gran número de bloques pre-diseñados y puede ahorrarse escribir código para centrarse en la visualización y control de cada uno de sus componentes guiadas por eventos. Esto posibilita un mejor diseño de GUIs personalizadas. Esta aplicación es muy utilizada por ingenieros en la rama de la instrumentación en los procesos industriales, la electrónica, para sistemas de hardware de diferentes proveedores, procesamiento de señales, para el desarrollo de softwares con calidad de código y creando ejecutables [171], [172].

Tcl/Tk es la combinación del lenguaje Tcl (Tool Command Language) y Tk (ToolKit). El lenguaje de programación interpretado y dinámico Tcl es multi-paradigma, de código abierto, orientado a objetos y eventos, multiplataforma, imperativo y funcional. Su uso e implementación es extensible incluyendo aplicaciones de escritorio, web, redes, bases de datos, desarrollo embebido y otros. Tk es un kit de herramientas de widgets para crear GUIs nativas de escritorios multiplataforma, de código abierto y además de ser

estándar para Tcl funciona con otros leguajes enlazados como Python, Rubi, Perl, etc. El potencial de Tcl/Tk proporciona un verdadero "native look and feel" en las aplicaciones, mostrando muchos de los elementos gráficos nativos para cada sistema operativo. Se dispone gratuitamente de las herramientas, empleándose en universidades, gobiernos, administradores de sistema, laboratorios de investigación, empresas como Oracle, IBM, NBC 24x7, Shell Oil, Motorola, Pixar, entre otras [173], [174].

2.2.2 Móvil

La creación de interfaz gráfica de usuario en el entorno de la movilidad, se sostiene de varias herramientas alternativas para los dos sistemas más utilizados hoy día (Android e iOS). Los sistemas cuentan respectivamente con sus plataformas de distribución de millones de aplicaciones tanto en Google Play para Android y en App Store para iOS. Ambos cuentan con tecnologías de desarrollo oficiales para apps nativas y se ha experimentado en los últimos tiempos una tendencia al desarrollo de aplicaciones web con diseños adaptativos o responsivos para las pantallas finales de los diversos dispositivos móviles existentes. Esto posibilita la portabilidad, no consumir espacio de almacenamiento en el dispositivo del usuario; pero son lentas y dependen de un navegador. Además, han surgido un gran número de librerías o frameworks de JavaScript como AngularJS, ReactJS y otros basados en el desarrollo web del lado del frontend denominadas PWAs (Progressive Web Aplications). Estas pueden funcionar con el sistema de notificaciones del móvil, sin conexión, pueden acceder a parte del hardware como las apps nativas, son instalables y aumentan la productividad eficiente del desarrollador/a escribiendo código una sola vez para apps multiplataforma. A continuación, se describen algunas de las herramientas de desarrollo móvil más utilizadas en la actualmente [175]- [184].

Las herramientas para la creación de aplicaciones nativas compatibles con **Android**, utilizan principalmente los lenguajes Kotlin, Java, C++ y XML. En sus comienzos se utilizaba Java y el kit de herramientas de desarrollo Android SDK (Software Development Kit), empleando el IDE Eclipse. Android Studio sustituyó a Eclipse como IDE (Integrated Development Environment) oficial para la plataforma, es gratuito, renderiza en tiempo real, tiene soporte integrado con Google Cloud Platform, posee un emulador, es multiplataforma y es compatible con C++ y otros, [178], [179], [185]- [188].

Para desarrollar aplicaciones nativas que funcionen en **iOS** es imprescindible tener como hardware un ordenador de Apple y el IDE Xcode. Se puede emplear como lenguaje Objetive-C, Swift o la combinación de ambos y también se puede agregar a un proyecto códigos fuentes de Python, Ruby, C++, Java, etc. Swift es un lenguaje moderno, su crecimiento ha sido veloz, es de código abierto, tiene un rendimiento superior a Objetive-C, compatible con este y otros lenguajes, orientado a protocolos, seguro y restrictivo para evitar que los desarrolladores cometan errores en tiempo de compilación [181], [189]- [191].

También existen en la actualidad otras herramientas de terceros muy utilizadas que son multiplataforma y gratuitas para crear aplicaciones nativas, tanto en Android como en iOS, denominadas aplicaciones Bridge. El framework React Native nos da la posibilidad de crear verdaderas apps nativas para ambos sistemas, empleando JavaScript. Además, se puede agregar código nativo de Java, Kotlin y Swift para la lógica de negocio etc. El framework multiplataforma Ionic es recomendable para aplicaciones hibridas, utilizando sólo HTML5, CSS y JavaScript. Podríamos decir que el resultado que se obtiene con este framework es una aplicación web incrustada en una móvil de forma nativa, utilizando un componente de Cordova. Por otro parte tenemos Xamarin, kit de herramientas de GUI multiplataforma para la elaboración de aplicaciones nativas para Android e iOS, con lenguaje de motor principal C# y utilizando el IDE Visual Studio [192]- [200].

2.2.3 Web

El proceso de desarrollo web para sitios y aplicaciones se divide en dos partes fundamentales denominadas frontend (lado del cliente o usuario) y backend (lado del servidor). La comunicación entre las partes se realiza con el protocolo HTTP (Hiper Text Transfer Protocol), mediante conexión unidireccional o bidireccional y síncrona o asíncrona a través de Get, Post, Ajax y Sockets. Se cuenta a día de hoy, con un crecimiento exponencial de diferentes herramientas y tecnologías de desarrollo para ambas partes [201]- [203].

Frontend

Es la capa de presentación gráfica con la que interactúa el usuario vía web presentando y recibiendo información. Esta se desarrolla con tres lenguajes fundamentales, HTML, CSS y JavaScript que son interpretados por los navegadores. HTML es el lenguaje de marcado para modelar la información, estructurándola por etiquetas para la interfaz gráfica. CSS proporciona el diseño y estilo de las etiquetas como los bordes, colores, dimensiones y otros. JavaScript es el lenguaje de programación interactivo que incorpora animación, efectos y otras funcionalidades. Se detallan a continuación las principales características de estos lenguajes [201]- [204]:

El lenguaje de marcas de hipertexto **HTML** (Hyper Text Markup Language) es un estándar asociado a las tecnologías de la W3C (World Wide Web Consortioum). Mediante código HTML se define la estructura del contenido e información de las páginas web que son interpretadas por los diferentes navegadores. La configuración de su código se realiza con etiquetas que pueden contener atributos, imágenes, videos, enlaces, formularios, scripts y etc. El código puede ser escrito o editado en herramientas de edición y procesamiento de textos que posibilitan observar en tiempo real los resultados gráficos del código HTML implementados, con el concepto de WYSIWYM. Se puede nombrar algunos de ellos como Adobe Dreamweaver, Microsoft Visual Studio para web, Sublime Text, Netbeans y otros más simples como el bloc de notas de los sistemas operativos [195], [205].

CSS (Cascading Style Sheets) traducido al español "Hojas de estilo en cascada", es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML; el lenguaje puede ser aplicado a cualquier documento XML, incluyendo XHTML, SVG, XUL, RSS, etcétera. También permite aplicar estilos no visuales, como las hojas de estilo auditivas. Existen algunos frameworks especializados en diseño web para CSS como Bootstrap, Materialize y otros [196], [206].

El lenguaje de programación interpretado e imperativo **JavaScript** o **JS**, está basado en las especificaciones del estándar ECMAScript. Es orientado a objetos, basado en prototipos y con tipado dinámico. Posibilita dinamismo en las páginas web respondiendo a las interacciones de los usuarios con la interfaz gráfica. Actualmente es un lenguaje integrador para crear de aplicaciones móviles nativas y muy usado en el desarrollo web tanto en el frontend como en backend, no siendo así en sus inicios. Existen numerosos y muy utilizados frameworks y librerías de código abierto como AngularJS, ReactJS y VueJS para el frontend y otros como NodeJS con Express para el backend [194], [207].

Backend

Es la capa que contiene la lógica del funcionamiento de la interfaz gráfica, donde se procesa y almacena la información. Esta contiene las aplicaciones alojadas en servidores web como por ejemplo Apache, NGINX, Jetty, Glassfish, Wildfly y otros. Los lenguajes de programación más utilizados en el backend y sus respectivos frameworks y librerías son: PHP con Synfony, Rubi con Ruby on Rails, Java con Spring, Python con Django y NodeJS para JavaScript. Las aplicaciones se conectan con las bases de datos para manipular la información, utilizándose mayormente los conocidos MySQL (My Structured Query Language) para las relacionales y MongoDB para las No SQL. Los usuarios no pueden acceder al backend por términos de seguridad, solo los desarrolladores. Existen eficientes prototipos de sistemas de infraestructura denominados stacks, con la combinación específica de algunas herramientas que la comunidad internacional de programadores a adoptado en muchos proyectos, dos de los más importantes son LAMP (Linux-Apache-MySQL-PHP) y MEAN (mongoDB-AngularJS-ExpressJS-NodeJS) [202]- [204], [208]- [224].

Para la selección del lenguaje y framework a utilizar en el desarrollo de la herramienta de programación y estimación del consumo energético del DAE, se tuvo en cuenta que el desarrollador tenía conocimientos básicos sobre el lenguaje Java. El framework Java “ZK” fue elegido porque cumplía con las funcionalidades, herramientas y recursos requeridos para obtener el producto final deseado. Además, con ZK no se necesitaba tener conocimientos sobre el lenguaje web JavaScript, ni de Ajax, los cuales no se dominaban y de esta forma se podía optimizar el tiempo de desarrollo [187], [225], [226].

Java versión 10.0.1 es un lenguaje de programación orientado a objetos (POO), multiplataforma, código abierto y de propósito general. Creado por la empresa Sun Microsystems y perteneciente actualmente a Oracle Corporation. Deriva del lenguaje C sus reglas de sintaxis como los bloques de códigos se modularizan en métodos y se delimitan con llaves { }, las variables se declaran antes de que se usen y estructuralmente, el lenguaje Java comienza con paquetes donde se encuentran las clases y dentro de las clases se encuentran métodos, variables, constantes, entre otros. Java es un lenguaje independiente de la plataforma y un entorno de ejecución ligero y gratuito para las plataformas más populares de forma que los códigos binarios (bytecode) de las aplicaciones Java puedan ejecutarse en cualquier plataforma por la máquina virtual Java (JVM) [187], [228].

El JDK (Java Development Kit) es un software con herramientas de desarrollo para programar programas en Java. Este JDK es distribuido bajo la licencia GNU (General Public Licence/licencia para software libre). Para Java existen diferentes entornos de trabajo. Algunos de ellos son de libre distribución (Open Source) mientras que otros son de licencia. Entre los entornos libres destacan Eclipse de Eclipse Foundation y NetBeans de Sun. Java es uno de los lenguajes de programación más utilizados por millones de programadores en todo el mundo para el desarrollo de aplicaciones de cliente-servidor web [228]- [230].

ZK versión 7.0 es un framework web Java de código abierto y multiplataforma, desarrollado por Potix Corporation para el diseño y creación de aplicaciones responsivas para web y móviles. Tiene una filosofía de desarrollo web con mecanismos conducidos por eventos en AJAX sin necesariamente tener conocimientos de JavaScript o del propio AJAX. Utiliza un lenguaje de marcado de interfaz de usuario ZK denominado ZUML (ZK User Interface Markup Language) para representar y describir una GUI. ZUML está basado en XUL (XML User Interface Language), originalmente definido por el equipo de Mozilla. Este lenguaje de marcado, tiene una línea de aprendizaje casi plana para los desarrolladores ya que es muy parecido a escribir las páginas en XHTML (eXtensible HyperText Markup Language), muy similar a HTML, pero escrito con la sintaxis de XML [225], [231].

2.3 Uso actual de los lenguajes de programación

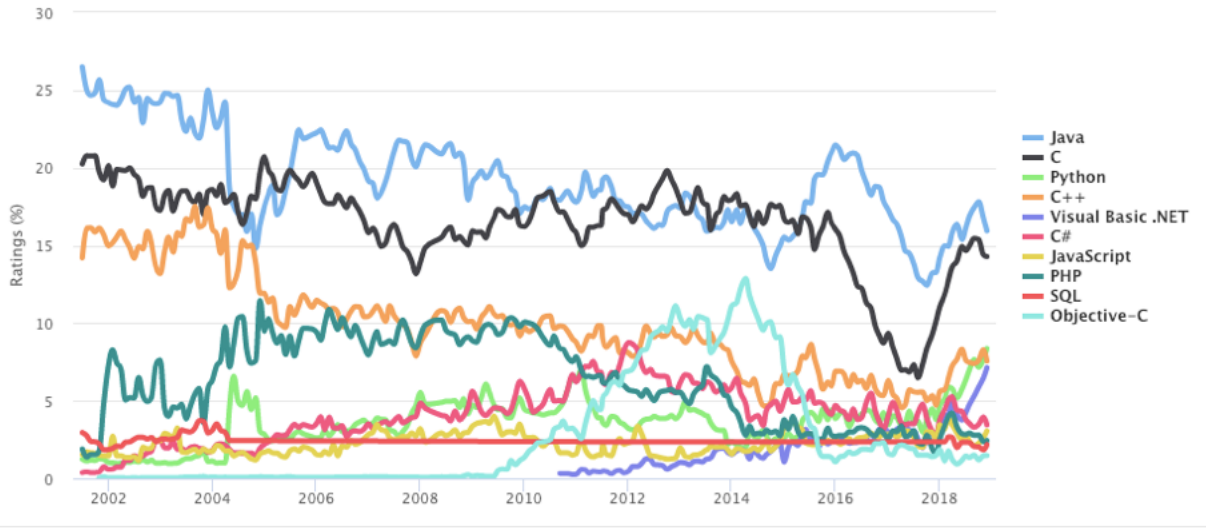


Figura 11. Índice de la Comunidad de Programación TIOBE [232]

La anterior figura nos demuestra el estado actual del uso de los diferentes lenguajes de programación alrededor del mundo según el índice de la comunidad de programación TIOBE (The Importance of Being Earnest). Este es un indicador de la popularidad de los diferentes lenguajes existentes, actualizándose mensualmente. Su calificación se basa en 25 motores de búsqueda determinados por Alexa el asistente virtual de Amazon, entre ellos los navegadores más populares. Además del número de ingenieros calificados internacionalmente, los cursos y proveedores externos. Es importante tener en cuenta que el índice TIOBE no se refiere al mejor lenguaje de programación o al lenguaje en el que se han escrito la mayoría de las líneas de código. TIOBE está especializada en evaluar, investigar y medir la calidad de un sistema de software aplicándole estándares de codificación ampliamente aceptados [232].

El lenguaje de programación Python obtuvo la mayoría de los puntos de clasificación en 2018 como el más utilizado entre la mayoría de lenguajes. Python ha ganado por 3.62%, seguido de Visual Basic .NET y Java. Se ha convertido definitivamente en parte de los grandes lenguajes de programación. C, C ++ y Java son por lo general los 3 primeros por delante del resto durante casi dos décadas. Actualmente Python tiene un creciente uso en la programación de IA (Inteligencia Artificial), el "Machine Learning", redes neuronales y ciencias de datos. Es el número uno en el dominio estadístico, computación científica, secuencias de comandos, pruebas del sistema de escritura, programación web y se enseña con mayor frecuencia en las universidades hoy en día [232], [233].

CAPÍTULO 3. Metodología y planificación.

En el proceso de desarrollo de todo software es vital la organización, estructura, guía y control a seguir mediante una metodología o conjunto de estas. Ellas nos permiten planificar de manera eficiente y eficaz las fases de desarrollo y los recursos a utilizar para obtener un producto final deseado, con la mayor calidad posible. En este capítulo se describe la metodología aplicada para la realización del proyecto y la estimación temporal de las tareas a realizar.

3.1 Metodología de desarrollo

El Lenguaje de Modelado Unificado o **UML** (Unified Modeling Language) fue creado en Rational Software. Actualmente es administrado y estandarizado por OMG (Object Management Group) y aprobado como normas ISO (International Organization for Standardization) ISO/IEC 19505-1. Se caracteriza por ser de uso general y estandarizar como lenguaje de modelado la especificación, visualización, descripción y documentación de diferentes sistemas, principalmente en el área de la ingeniería de software. Con este lenguaje se puede modelar 13 tipo de diagramas contenidos en 2 categorías principales denominadas diagramas de estructura y diagramas de comportamiento, además de una subcategoría perteneciente a los diagramas de comportamiento nombrada diagramas de interacción. La categoría de estructuras comprende los diagramas de clase, objetos, componentes, estructura compuesta, paquetes y de implementación o despliegue. La categoría de comportamiento se define con los diagramas de casos de uso, actividad y máquina de estados. Los diagramas de interacción derivados de la categoría de comportamiento, incluyen los diagramas de secuencia, comunicación, tiempo y el general de interacción. Los diagramas definen aspectos globales del sistema como los procesos, funcionalidades, artefactos y otros [234], [235].

El Proceso Unificado Racional o **RUP** (Rational Unified Process) es desarrollado por Rational Software y adquirido por IBM. Es una implementación específica del Proceso Unificado para el desarrollo de software, adaptándose a diferentes sistemas, entornos de aplicación y dimensiones de proyectos de desarrollo. En combinación con UML conforman una completa metodología estándar para los procesos de desarrollo de sistemas orientados a objetos. Se caracteriza por ser dirigido por casos de usos en el establecimiento de las funcionalidades de un sistema, centrado en la arquitectura para un eficiente funcionamiento y reutilización de sus componentes basados en óptimos patrones de diseño arquitectónicos, iterativo e incremental para retroalimentarse de su ciclo de vida comprendido por 4 fases y 9 disciplinas, además se enfoca en una minuciosa identificación de riesgos durante todo el proceso de desarrollo [236]- [238].

3.1.1 Fases y disciplinas del Proceso Unificado Relacional

Las fases que componen el ciclo de RUP se denominan Inicio, Elaboración, Construcción y Transición. Estas cuentan con varias iteraciones que van incrementando sus funcionalidades y características a lo largo del ciclo, en dependencia de un complejo de disciplinas que están definidas por actividades, flujos de trabajo, modelos y sus artefactos.

La fase inicial agrupa y describe requisitos, riesgos, factores económicos y su estimación temporal. Dentro de la fase de elaboración se define las funcionalidades del sistema, los posibles riesgos, modelo de diseño, arquitectura de desarrollo software con sus capas, librerías e interfaces. En la fase de construcción se desarrolla el software con el código y componentes estructurados por la arquitectura seleccionada, implementando los casos de uso y requisitos que componen el sistema. Se denomina fase transición al resultado y validación del producto final puesto a disposición de los clientes o usuarios, sus instrucciones y acuerdos de posteriores mantenimientos y actualizaciones. El ciclo finaliza cuando se cumples los objetivos propuestos de la fase inicial y con la satisfacción del cliente [236]- [238].

Entre las disciplinas se pueden encontrar la de modelado de negocio o empresarial para el análisis, comprensión y solución de la lógica de negocio que rige el desarrollo de la aplicación. La de requisitos estima los costos y tiempo de desarrollo total. Las disciplinas de análisis y diseño automatizan los requisitos y se estructura la arquitectura del sistema. En la implementación se desarrolla el código y la completa aplicación deseada, rigiéndose por la arquitectura y comprobándola mediante pruebas. La disciplina de prueba o test se asegura del correcto y completo funcionamiento del sistema, evaluando su calidad y validez. Despliegue consiste en las distribuciones y ejecución del sistema, disponible al uso de los clientes. Las disciplinas de cambio y configuración se encargan del control y gestión de versiones y sus artefactos. La de gestión de proyectos se centra en la planificación de recursos, personal, actividades y riesgos. Como ultima disciplina se plantea el entorno o ambiente controlado durante el desarrollo, asegurando los procesos, métodos y herramientas disponibles [236]- [238].

La siguiente figura muestra el ciclo de RUP integrando la relación y variación de las 9 disciplinas respecto a cada iteración de las 4 fases del proceso de desarrollo software.

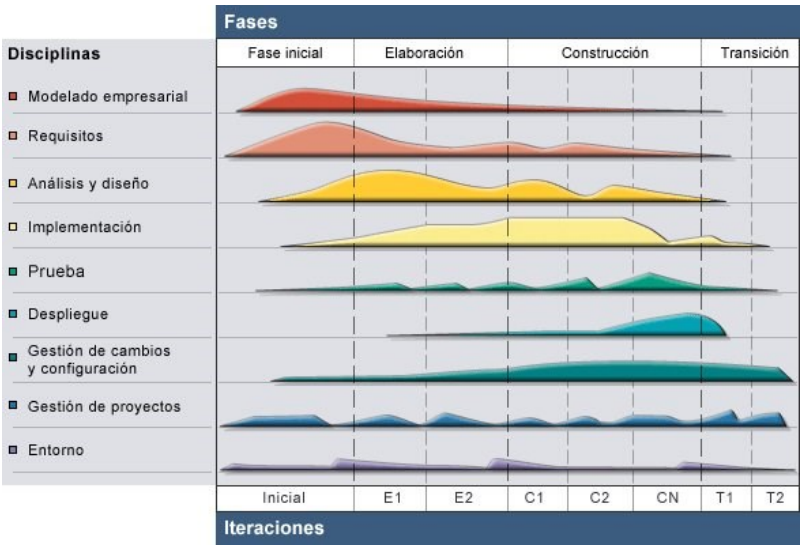


Figura 12. Fases y disciplinas RUP [239]

Existen dos vertientes importantes de metodologías de desarrollo, las tradicionales y las ágiles como por ejemplo SCRUM, Jackson System Development (JSD), Structured System Analysis and Design Method (SSADM), metodología METRICA, Rapid Application Development (RAD), Adaptive Software Development (ASD Feature-Driven Development (FDD), Extreme programming (XP), Crystal Methologies, Lean Development (LD), Agile Unified Process (AUP), Dynamic Systems Development (DSDM), etc. para algunos RUP es ágil, para otros tradicional, esta es una metodología que dependiendo de su uso y enfoque se puede clasificar de una manera u otra [240]- [242].

Por ser este un proyecto de una sola persona y con poca experiencia en el mundo del desarrollo software, se determinó hacer una selección de las directrices principales de la metodología de RUP para satisfacer las necesidades de este TFM. Además, este es un proceso guiado por la arquitectura, sus casos de uso y se trabaja en conjunto con el usuario final, que utilizará el software. La metodología se complementará con UML para la descripción de los distintos diagramas del modelado, formalizando el proceso con su documentación, puesto que proporciona los artefactos y modelos para ello.

3.2 Planificación temporal

A continuación, se describen el conjunto de actividades propuestas como planificación temporal del proyecto en desarrollo para la creación de la herramienta de programación y estimación del consumo energético del DAE en el entorno de la acuicultura.

Descripción de las tareas a realizar:

1) **Estudio del dispositivo autónomo de experimentación (T1) (1 semanas).** Se estudiará el DAE desarrollado en la división MEMS del IUMA para el proyecto europeo AE2020, centrándose en particular en las especificaciones referentes a los distintos modos de trabajo del dispositivo (run, very low power run, stop, wait entre otros). También se estudiará la especificación de experimentos basada en primitivas de medida y computo, así como su temporización.

2) **Estudio de los interfaces de gráficos de usuario (T2) (1 semanas).** Se realizará un estudio del arte de los diversos interfaces gráficos de usuario que permitan en particular arrastrar, cortar y pegar (característica drag & drop) objetos gráficos en pantalla. Todo ello con el fin de reducir la curva de aprendizaje del usuario de la herramienta propuesta en este TFM.

3) **Desarrollo del entorno de programación de experimentos (T3) (4 semanas).** En esta tarea se plantea el desarrollo un software que permita la introducción de las primitivas y sus parámetros de forma sencilla por parte de los desarrolladores del dispositivo de experimentación mediante por ejemplo lenguaje XML o similar (entrada en texto). Así mismo, se dotará al entorno de programación de experimentos de una interfaz gráfica donde se representará dichas primitivas, permitiendo realizar la especificación del experimento gráficamente (uso de primitivas y su temporización).

4) **Estimación del consumo/horizonte temporal (T4) (2 semanas).** Dentro de los parámetros a incluir en cada primitiva se ha de especificar el conjunto de valores que permitan estimar el consumo energético y su temporización. Estos datos se han de introducir por parte de los desarrolladores del dispositivo de experimentación en los ficheros XML o similares. Disponibles estos datos y en base a la programación propuesta gráficamente por el usuario, la herramienta ha de estimar el consumo y el horizonte temporal alcanzable con dicha especificación.

5) **Validación de la herramienta desarrollada (T5) (3 semanas).** Se determina el conjunto de pruebas que ha de superar la herramienta propuesta en este TFM teniendo en cuenta que el perfil del usuario que la va a utilizar está centrado en los datos del experimento y no tiene porqué conocer detalles electrónicos o de funcionamiento a bajo medio/bajo del dispositivo autónomo de experimentación.

6) **Redacción de documentación (T6) (4 semanas).** Cada tarea de este TFM tiene asociada su correspondiente documentación elaborada durante el desarrollo de cada tarea. En esta última tarea se ordenará y complementará dicha documentación en forma de memoria.

Tabla 1. Planificación temporal de las Tareas (T)

Comienzo Fin	**Tareas**					
01/01/2018 07/01/2018	**T1** (1 semana)					
08/01/2018 21/01/2018		**T2** (1 semanas)				
22/01/2018 04/02/2018			**T3** (4 semanas)			
05/02/2018 19/02/2018				**T4** (2 semanas)		
20/02/2018 05/03/2018					**T5** (3 semanas)	
06/03/2018 30/05/2018						**T6** (1 mes)

La tabla anterior nos ilustra el tiempo aproximado en semanas que se dispone para la materializar las tareas propuestas. Se puede observar que las tareas que más tiempo consumen son la 3 y 6, relacionadas respectivamente al desarrollo de la herramienta y a la documentación de la misma. La T5 referente a la validación, nos consumirá un tiempo prudencial hasta lograr la aceptación de la herramienta. Podemos decir que el proyecto tendrá una duración de aproximadamente 16 semanas, pudiéndose extender en dependencia del flujo de trabajo y el cumplimiento de las diferentes etapas de desarrollo.

CAPÍTULO 4. Recursos empleados

4.1 Recursos hardware

Especificaciones técnicas del fabricante:

 Ordenador portátil Apple MacBook Air (13 pulgadas, principios de 2014) [243].

Pantalla
- Pantalla panorámica brillante retro iluminada por LED de 13,3 pulgadas (en diagonal) compatible con millones de colores.
- Resolución: 1.440 x 900 píxeles.

Capacidad
- 256 GB de almacenamiento flash PCIe.

Procesador
- Core i5 de Intel de doble núcleo a 1,4 GHz (Turbo Boost de hasta 2,7 GHz) con 3 MB de caché de nivel 3 compartida.

Memoria
- 4 GB de memoria DDR3 integrada a 1.600 MHz.

Batería y alimentación
- Hasta 12 horas de navegación web inalámbrica.
- Batería integrada de polímeros de litio de 50 vatios/hora.
- Adaptador de corriente MagSafe 2 de 45 W.

Gráficos y compatibilidad de vídeo
- HD Graphics 5000 de Intel.
- Doble monitor y vídeo en espejo: admite simultáneamente la resolución nativa completa en la pantalla integrada y hasta 2.560 por 1.600 píxeles en un monitor externo, ambos compatibles con millones de colores.
- Salida de vídeo digital Thunderbolt.
- Salida Mini DisplayPort nativa.

Cámara
- Cámara FaceTime HD a 720p.

Conexiones y ampliación
- Auriculares
- Doble micrófono

- Dos puertos USB 3 (hasta 5 Gb/s)
- Puerto Thunderbolt (hasta 10 Gb/s)
- Toma de corriente MagSafe 2
- Ranura para tarjetas SDXC

Conexión inalámbrica

- **Wi-Fi**
  Conexión inalámbrica Wi-Fi 802.11ac;[3] compatible con las normas 802.11a/b/g/n del IEEE.
- **Bluetooth**
  Tecnología inalámbrica Bluetooth 4.0.

Audio

- Altavoces estéreo.
- Doble micrófono.
- Toma para auriculares.

Teclado y trackpad

- Teclado retro iluminado de tamaño estándar con 79 teclas, entre ellas 12 de función y 4 de flecha (dispuestas en forma de T invertida), y sensor de luz ambiental.
- Trackpad con tecnología Multi-Touch para controlar el cursor con precisión; compatible con el desplazamiento inercial. Permite pellizcar, girar, deslizar, deslizar con tres y cuatro dedos, tocar, tocar dos veces y arrastrar.

Sistema operativo

- macOS High Sierra versión 10.13.5.

4.2 Recursos software

Los softwares indispensables utilizados en el proceso de desarrollo del proyecto son:

Sistema operativo

macOS en su versión **High Sierra 10.13.5** (anteriormente Mac OS X y posterior OS X) es un sistema operativo gráfico gratuito desarrollado por la empresa Apple Inc. Es la segunda serie de sistemas operativos Macintosh destinada a la gama de ordenadores Mac con plataformas x86-64 [244].

Explorador

Google Chrome versión 67.0.3396.87 es un navegador web gratuito desarrollado por Google LLC. Según las estadísticas más recientes este navegador web multiplataforma es el más utilizado y con mayor cuota de mercado [245].

Máquina Virtual

 La **Java Virtual Machine** (Máquina Virtual Java) desarrollada por Sun Microsystems es un elemento fundamental de la plataforma de Java y se ejecuta en una maquina real (ordenador). Con el eslogan de "Escriba una vez, ejecútalo en cualquier lugar" se caracteriza por ser multiplataforma, teniendo una JVM específica por sistema operativo, la cual interpreta código binario Java(.class) y lo traduce a código nativo de la plataforma final. El Bytecodes (.class) es generado por el compilador **javac** u otro que tenga como entrada el archivo(.java) de código fuente [187], [228], [246].

Servidor web

 **(WildFly Application Server)** versión 13 "Baker's Dozen" es un servidor web gratuito y de código abierto para aplicaciones de plataforma Java, el cual pertenece a Red Hat Inc. Fue denominado anteriormente como JBoss AS y es multiplataforma dependiendo de la disponibilidad de la máquina virtual por sistema operativo [211], [247].

Lenguajes y Framework

El lenguaje de programación empleado para en el desarrollo de la herramienta es Java y el framework es ZK como se expuso en el capítulo 2 epígrafe 2.2.3 titulado "Web". Además, se utilizó el lenguaje de marcado que se describe a continuación:

 El meta-lenguaje de formato abierto **eXtensible Markup Language** (Lenguaje de Marcado Extensible) es un subconjunto de SGML (Standard Generalized Markup Language) que posibilita definir gramaticalmente lenguajes de marcado adaptado a usos específicos. Fue desarrollado por el World Wide Web Consortium (W3C) para describir información, organizar, dar soporte a base de datos, facilitar el almacenamiento y transmisión de contenido en forma legible, segura y fácil por medios de comunicación digital [248]- [250].

Entorno de desarrollo integrado(IDE)

 **Eclipse** es un entorno de desarrollo integrado(IDE) multiplataforma y de código abierto. Es creado inicialmente por IBM (International Business Machines Corporation) y desarrollado en la actualidad por La Fundación Eclipse. Es una potente plataforma de desarrollo software universal, personalizable y extensible, que emplea módulos (plug-in) y frameworks, permitiendo programar y compilar en varios lenguajes de programación como Python, PHP, Perl, C++ y el más utilizado Java. El afamado IDE de Java incluye una amplia variedad de avanzadas herramientas y funcionalidades para el desarrollo de aplicaciones Java con su propio compilador interno y el modelado de los archivos fuente de Java. Estas herramientas nos permiten gestionar mejor, desplegar, escribir, ejecutar y depurar nuestras aplicaciones con una mayor versatilidad [210], [251].

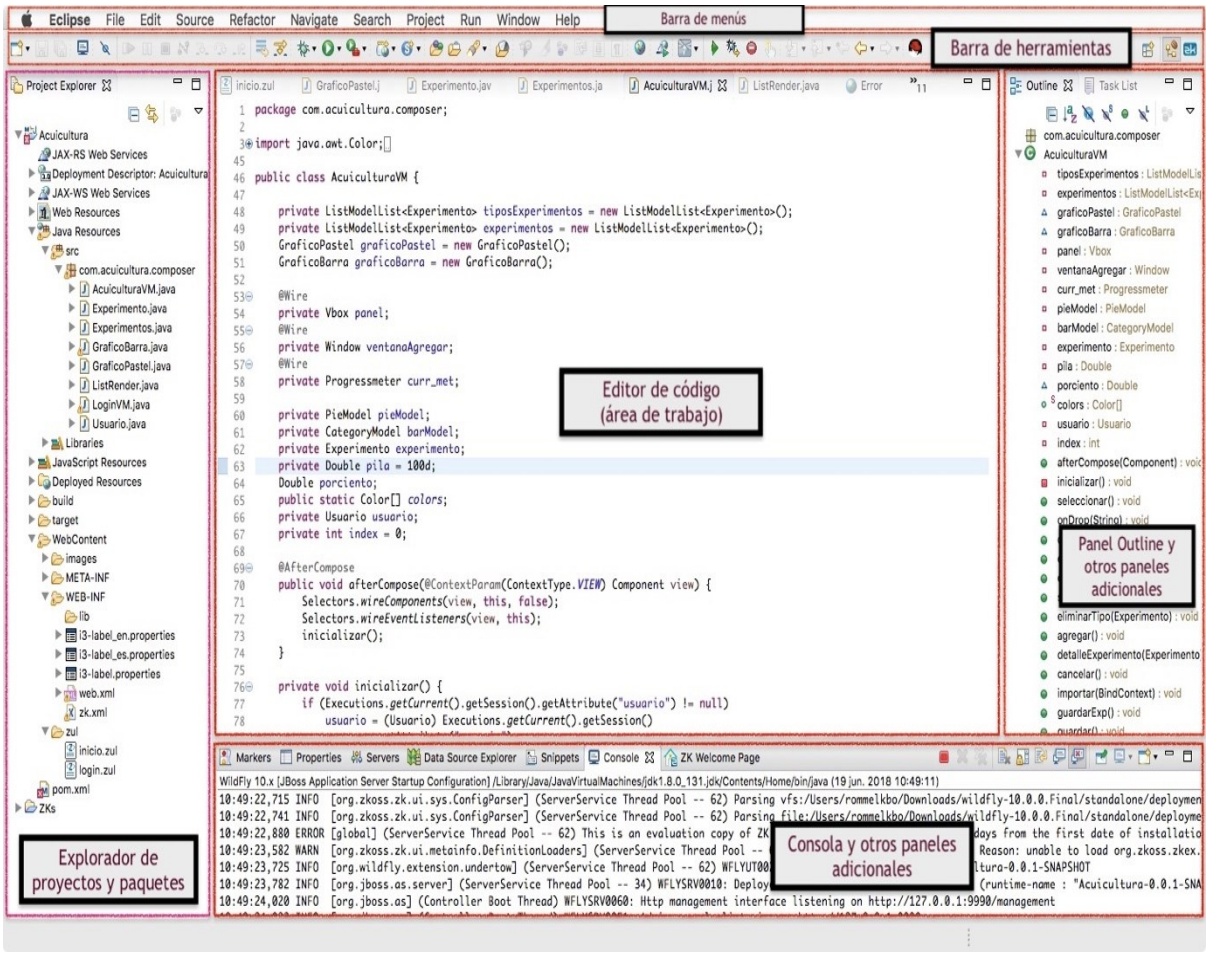


Figura 13. Entorno de desarrollo Integrado Eclipse

La figura anterior nos describe gráficamente el diseño y organización estándar de las barras, vistas o paneles del entorno de desarrollo integrado Eclipse, los cuales se explican a continuación:

- **Barra de menús**

Ubicada en la parte superior de la aplicación, muestra los menús desplegables de la plataforma software con todas las herramientas, usabilidad y opciones disponibles para el desarrollo de proyectos.

- **Barra de herramientas:**

Es una síntesis de la barra de menús con botones y accesos directos a herramientas y funciones principales como crear, guardar, ejecutar el proyecto Java, etc. Se muestra en la parte superior debajo de la barra de menús.

- **Vista de explorador de proyectos y paquetes:**

A la izquierda del entorno de desarrollo está situado el explorador de proyectos y paquetes con su jerarquía distribuida por carpetas, conteniendo los archivos, clases y recursos que lo componen. Estos archivos son editables en cuanto a cambiar su ubicación,

nombre, creación, descripción, copiar, eliminar y otros como abrirlos en el editor de código o área de trabajo con doble clic.

- **Vista de editor de código (área de trabajo):**

El panel central de la herramienta Eclipse es el área fundamental de trabajo y edición de código fuente de programación. Para cada archivo fuente seleccionado en el explorador de proyectos se genera un editor correspondiente y se aplican los cambios definitivos con el botón guardar.

- **Vista exterior(Outline) y otros:**

A la derecha del entorno de desarrollo se observa este panel, detallando la estructura esquemática de aquellos archivos seleccionados en el explorador de proyectos y visualizados en el editor de código, con sus elementos, componentes, clases, métodos, variables, etc. Además de esta vista se pueden agregar otras como el terminal, frameworks, servidor web, etc.

- **Vista de problemas y otros:**

En la parte inferior y central de la aplicación se sitúa el panel de vistas de problemas, donde a medida que editamos código en nuestro proyecto nos registra automáticamente si tenemos errores, problemas o advertencias. Esto nos da la posibilidad de conocer el estado del proyecto, los errores de sintaxis de código y problemas de los componentes de desarrollo. Podemos acceder de forma directa a la línea de código que genera una advertencia o error haciendo doble clic en el icono de los mismos. Podemos agregar otras vistas en este panel como la consola, servidores, propiedades generales y otros.

Otros

Microsoft Word versión 15.31 con licencia de pago de suscripción a Office 365 es proporcionado de forma gratuita por la Universidad de Las Palmas de Gran Canaria. Esta herramienta ofimática es un software multiplataforma propiedad de la empresa Microsoft Corporation, orientado a la creación y procesamiento de textos, y forma parte del paquete de Microsoft Office [252], [253].

Vista previa versión 10.0 es una versátil herramienta integrada en el sistema operativo macOS, que permite realizar varias acciones como visualizar, editar, anotar, dibujar y firmar archivos PDF, JPG, formularios y otros. Es capaz de compartir, exportar en diferentes formatos de salida protegiéndolos mediante cifrado y protección por contraseña [254].

StarUML en su versión 3.0.2 es una herramienta UML (Unified Modeling language) multiplataforma, gratuita y de código abierto desarrollada por MKLab. Proporciona varios tipos de diagramas y herramientas para la documentación de requisitos. Es muy utilizado por millones de usuarios e importantes empresas en la confección de diagramas de casos de usos, diagramas de clase, secuencia, colaboración, componentes, etc. Es una

herramienta modular y abierta, proporcionando compatibilidad con metamodelos y diagramas estándar de UML [255], [256].

 La herramienta **inVision** para crear prototipado de interfaces de usuario basada en el diseño de aplicaciones web y dispositivos móviles. Fundada en 2011 con sede en nueva York y es muy utilizada por diseñadores de empresas como Adobe, Twitter, Evernote, IBM, Netflix, Amazon y otros. La plataforma tiene en la actualidad más de 4 millones de usuarios en su versión gratuita, posibilitando crear prototipos, iterarlos, administrarlos y otras funcionalidades en su diseño [257].

4.3 Recursos económicos

El análisis de presupuesto económico para la realización del presente TFM supone dos costes fundamentales para el estudiante:

Coste de inventario tangible

Fue indispensable adquirir un ordenador portátil para el desarrollo y documentación de este proyecto. En el apartado anterior sobre los recursos software empleados en el proyecto se pudo inferir que todas las herramientas son de software libre y gratuito. Por tanto, no se incurrió en gastos de este tipo. El valor se puede apreciar en la siguiente tabla.

Tabla 2. Costes de inventario tangible

Concepto	Cantidad	Precio por unidad	Precio total
Equipo portátil Apple MacBook Air	1	980 €	980 €

Coste de personal

Se han determinado 10 euros por hora trabajada del estudiante, para una media de 20 días laborales por mes y 4 horas por día. Teniendo en cuenta el apartado 3.2 en cuanto a la planificación laboral, tendríamos un total de 16 semanas para elaborar el proyecto. Esto supone un total de 800€ mensuales y 3200€ del proyecto total, como muestra la siguiente tabla.

Tabla 3.Costes de Personal

Concepto	Horas	Precio por Horas	Precio total
Coste Laboral Alumno	320h	10€	3200€

El total de ambos costes seria 4.180€ y podría sumarse el coste laboral de los tutores y de las matrículas del estudiante.

CAPÍTULO 5. Desarrollo de la herramienta de software.

En este capítulo se describe y fundamenta la elaboración del proyecto, orientado por las principales directrices marcadas en la metodología de desarrollo RUP y empleando las herramientas necesarias para lograr los objetivos propuestos.

El proceso de desarrollo fue iterativo e incremental elaborándose por módulos para cada caso de uso y adicionando las nuevas funcionalidades de los mismos en concordancia a los requisitos previamente establecidos por el cliente. El lenguaje unificado de modelado UML nos facilitó parte de la documentación del presente capítulo, elaborándose un grupo de diagramas y otros.

Partiendo de los requisitos del sistema y del software se definió la amplitud de la aplicación, empleando un modelo de dominio, una lista de características, actores del sistema, casos de usos, diseño del prototipo de interfaz gráfica de usuario, patrón de diseño arquitectónico, arquitectura del sistema e implementación y construcción de la herramienta software.

5.1 Requisitos del sistema

Los requisitos previos a tener en cuenta en el proceso de desarrollo del presente proyecto, son principalmente tener conocimientos básicos sobre Java, CSS, XML, ZUML y otros. Además, el dominio del framework utilizado ZK para implementar modernos recursos y funcionalidades gráficas en una aplicación web.

El framework ZK permite reducir el tiempo de desarrollo, la reutilización de código y partes de la aplicación, una estructura bien definida y organizada sobre la base de una avanzada arquitectura Model-View-ViewModel (MVVM). Este framework nos proporciona un aumento de la productividad y eficiencia del software en desarrollo.

Además, como requisito del sistema para desplegar la aplicación web se necesita los recursos hardware anteriormente descritos en el capítulo 4, una Máquina Virtual Java (JVM) y el servidor web WildFly para aplicaciones de plataforma Java.

5.1.1 Modelo de dominio

Se puede definir a un modelo de dominio como el artefacto dentro de la disciplina de análisis de la metodología empleada, como la concepción o idea inicial para la creación de los objetos software del sistema. De forma general se utiliza como la representación de las clases conceptuales de nuestro mundo no de componentes softwares necesariamente. Se utiliza este modelo para el entendimiento previo al diseño del sistema que va ser empleado por una entidad como una empresa, negocio, industria, etc. Es un modelo factible para la programación orientada a objetos, la descripción de las clases del sistema, casos de usos, diseño de interfaz de usuario, etc.

La herramienta de programación y estimación del consumo energético para un Dispositivo Autónomo de Experimentación en Acuicultura tiene como principal objetivo la descripción gráfica de experimentos basados en parámetros y su temporización, teniendo en cuenta la duración de los experimentos y el consumo energético de la batería. La idea global es que el Usuario Acuicultura o Administrador accedan a la aplicación con sus respectivos nombres de usuarios y contraseñas, dispongan de un entorno gráfico intuitivo, atractivo y sin necesidad de tener avanzados conocimientos informáticos. Los usuarios podrán realizar el análisis de las gráficas generadas en base a los parámetros de los experimentos y gestionar el consumo de la batería, a medida que se van ordenando los experimentos en el área de programación que es el entorno de trabajo y planificación. Los experimentos solo podrán ser confeccionados o modificados por el usuario Administrador en base a primitivas y sus tiempos, archivándolos en formato .xml para exportarlos o importarlos a la aplicación y a disposición de los Usuarios Acuicultura. En la siguiente gráfica se expone las clases generales de la herramienta software y para su elaboración se utilizó la aplicación StarUML descrita en el capítulo 4 epígrafe 4.2.

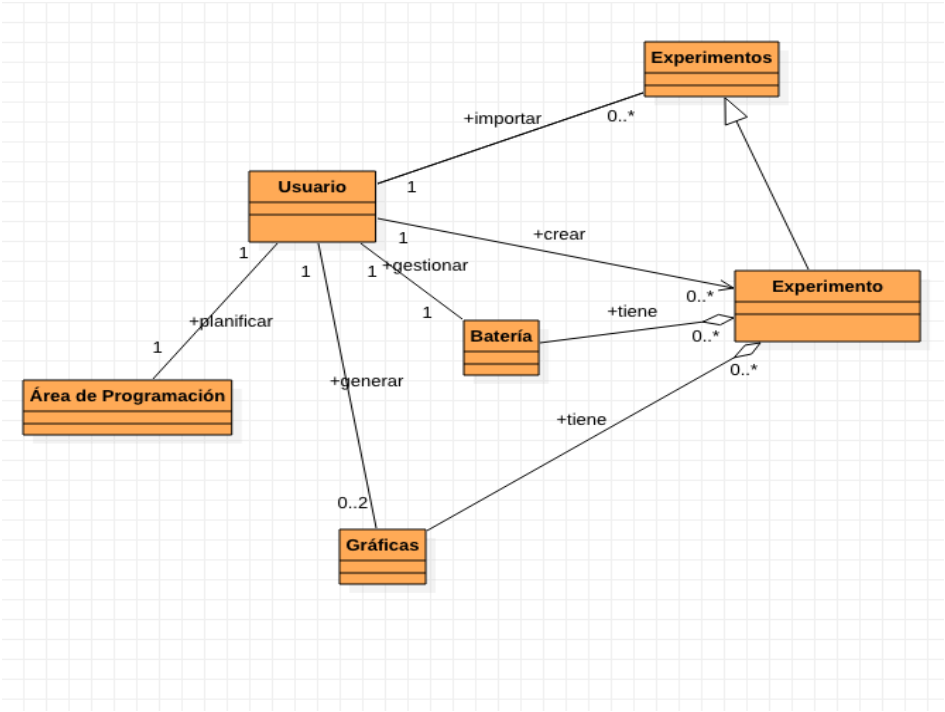


Figura 14. Diagrama de dominio

5.1.2 Lista de características

La lista de características está enmarcada en la fase de inicio y es el resultado de "Enumerar los requisitos candidatos" como tarea dentro de la metodología empleada. Su contenido son las ideas y aspectos que puede adoptar la herramienta en colaboración con todos los participantes de la misma, clientes, usuarios, desarrolladores y analistas, y que pudiera variar en el transcurso de su desarrollo. La planificación y gestión del trabajo a desarrollar parte de esta lista, originando los requisitos del software.

La composición de la tabla de lista de características está dada por los siguientes aspectos:

- **Código**: Identifica las características para cada tipo de usuario (LC + Tipo de usuario + Número de característica).
    a. AA: Administrador Acuicultura
    b. UA: Usuario Acuicultura.
- **Rol**: Define el tipo de usuario.
- **Nombre**: Nombra a la característica.
- **Descripción**: Describe la funcionalidad de la característica.
- **Prioridad**: Asigna un orden de prioridad de desarrollo a la característica (Muy alta, Alta, Media, Baja, Muy baja).
- **Estado**: Indica el progreso de desarrollo de la característica (Aceptado, Finalizado, Pendiente).

La siguiente tabla muestra la lista de características de la herramienta software propuesta.

Tabla 4. Lista de características

Código	Rol	Nombre	Descripción	Prioridad	Estado
LC-AA.1	Administrador Acuicultura	Iniciar sesión	El usuario podrá iniciar su sesión con su respectivo nombre de usuario y clave para hacer uso sus funcionalidades disponibles	Muy alta	Finalizado
LC-AA.2	Administrador Acuicultura	Cerrar sesión	El usuario podrá salir de su sesión de trabajo	Muy alta	Finalizado
LC-AA.3	Administrador Acuicultura	Crear nuevo experimento	Se le permitirá que el usuario pueda crear experimentos con sus características o parámetros y su descripción	Muy alta	Finalizado
LC-AA.4	Administrador Acuicultura	Insertar nombre de experimento	El usuario podrá añadir nombre al experimento	Alta	Finalizado
LC-AA.5	Administrador Acuicultura	Insertar etiqueta	El usuario podrá añadir etiqueta al experimento	Alta	Finalizado
LC-AA.6	Administrador Acuicultura	Elegir color de etiqueta	El usuario podrá añadir color a la etiqueta de experimento	Alta	Finalizado
LC-AA.7	Administrador Acuicultura	Insertar tiempo de ejecución	El usuario podrá añadir el tiempo de ejecución de un experimento en segundos	Alta	Finalizado
LC-AA.8	Administrador Acuicultura	Insertar energía de ejecución	El usuario podrá añadir la energía consumida en la ejecución de un experimento	Alta	Finalizado

LC-AA.9	Administrador Acuicultura	Insertar tiempo de espera	El usuario podrá añadir tiempo de espera entre un experimento y otro en minutos	Alta	Finalizado
LC-AA.10	Administrador Acuicultura	Insertar energía en espera	El usuario podrá añadir la energía consumida durante el tiempo de espera	Alta	Finalizado
LC-AA.11	Administrador Acuicultura	Observar operando 1	El usuario podrá ver el futuro parámetro a utilizar	Baja	Pendiente
LC-AA.12	Administrador Acuicultura	Observar operando 2	El usuario podrá ver el futuro parámetro a utilizar	Baja	Pendiente
LC-AA.13	Administrador Acuicultura	Insertar descripción de experimento	El usuario podrá añadir una descripción del experimento	Alta	Finalizado
LC- AA.14	Administrador Acuicultura	Guardar experimento	El usuario podrá almacenar el experimento con sus características y descripción en una lista	Muy alta	Finalizado
LC- AA.15	Administrador Acuicultura	Editar experimento	Se le permitirá al usuario modificar las características y descripción del experimento	Alta	Finalizado
LC- AA.16	Administrador Acuicultura	Mostrar experimento	El usuario podrá observar los parámetros y descripción del experimento	Alta	Finalizado
LC- AA.17	Administrador Acuicultura	Mostrar lista de experimentos	El usuario podrá observar una lista de todos los experimentos creados o importados	Muy alta	Finalizado
LC- AA.18	Administrador Acuicultura	Mostrar gráfico circular	El usuario podrá observar, analizar y comparar los parámetros de los experimentos	Alta	Finalizado
LC- AA.19	Administrador Acuicultura	Mostrar gráfico de barras	El usuario podrá observar, analizar y comparar los parámetros de los experimentos	Alta	Finalizado
LC- AA.20	Administrador Acuicultura	Mostrar gráfica de consumo de batería	El usuario podrá observar el porcentaje de consumo de la batería	Muy alta	Finalizado
LC- AA.21	Administrador Acuicultura	Planificar orden de experimentos	Se le permitirá al usuario ordenar los experimentos (Drag & Drop) en el área de programación y planificación	Alta	Finalizado

LC- AA.22	Administrador Acuicultura	Gestionar consumo de la batería	Se le permitirá al usuario gestiona el consumo de la batería en la planificación de varios experimentos	Muy alta	Finalizado
LC- AA.23	Administrador Acuicultura	Eliminar experimento	El usuario podrá eliminar experimento desde el botón en la lista o llevándolo a la papelera (Drag & Drop)	Muy alta	Finalizado
LC- AA.24	Administrador Acuicultura	Exportar experimentos	Se le permitirá al usuario exportar un conjunto de experimentos ya planificados en formato .xml	Alta	Finalizado
LC- AA.25	Administrador Acuicultura	Importar experimentos	El usuario podrá importar un archivo en formato .xml con un conjunto de experimentos programados	Alta	Finalizado
LC- AA.26	Administrador Acuicultura	Modificar idioma	Se le permitirá al usuario cambiar de idioma la aplicación (Español o Ingles)	Media	Pendiente
LC- AA.27	Administrador Acuicultura	Modificar porcentaje de batería	Se le permitirá al usuario modificar la escala de porcentaje de la batería (por defecto 100%)	Alta	Finalizado
LC-UA.1	Usuario Acuicultura	Iniciar sesión	El usuario podrá iniciar su sesión con su respectivo nombre de usuario y clave para hacer uso sus funcionalidades disponibles	Muy alta	Finalizado
LC-UA.2	Usuario Acuicultura	Cerrar sesión	El usuario podrá salir de su sesión de trabajo	Muy alta	Finalizado
LC-UA.3	Usuario Acuicultura	Importar experimentos	El usuario podrá importar un archivo en formato .xml con un conjunto de experimentos programados	Muy alta	Finalizado
LC- UA.4	Usuario Acuicultura	Editar descripción de experimento	Se le permitirá al usuario modificar descripción del experimento	Alta	Finalizado
LC- UA.5	Usuario Acuicultura	Mostrar lista de experimentos	El usuario podrá observar una lista de todos los experimentos importados	Muy alta	Finalizado
LC- UA.6	Usuario Acuicultura	Planificar orden de experimentos	Se le permitirá al usuario ordenar los experimentos (Drag & Drop) en el área de programación y planificación	Alta	Finalizado
LC- UA:7	Usuario	Gestionar consumo de	Se le permitirá al usuario gestiona el consumo de la	Muy alta	Finalizado

	Acuicultura	la batería	batería en la planificación de varios experimentos		
LC- UA:8	Usuario Acuicultura	Eliminar experimento	El usuario podrá eliminar experimento desde el botón en la lista o llevándolo a la papelera (Drag & Drop)	Muy alta	Finalizado
LC- UA:9	Usuario Acuicultura	Mostrar experimento	El usuario podrá observar los parámetros y descripción del experimento	Alta	Finalizado
LC- UA:10	Usuario Acuicultura	Mostrar gráfico circular	El usuario podrá observar, analizar y comparar los parámetros de los experimentos	Alta	Finalizado
LC- UA:11	Usuario Acuicultura	Mostrar gráfico de barras	El usuario podrá observar, analizar y comparar los parámetros de los experimentos	Alta	Finalizado
LC- UA:12	Usuario Acuicultura	Mostrar gráfica de consumo de batería	El usuario podrá observar el porcentaje de consumo de la batería	Muy alta	Finalizado
LC- UA:13	Usuario Acuicultura	Exportar experimentos	Se le permitirá al usuario exportar un conjunto de experimentos ya planificados en formato .xml	Muy alta	Finalizado
LC- UA:14	Usuario Acuicultura	Modificar porcentaje de batería	Se le permitirá al usuario modificar la escala de porcentaje de la batería (por defecto 100%)	Alta	Finalizado
LC- UA:15	Usuario Acuicultura	Modificar idioma	Se le permitirá al usuario cambiar de idioma la aplicación (Español o Ingles)	Media	Pendiente

5.2 Requisitos del software

Estos requisitos se obtienen a partir de la información expuesta en el modelo de dominio y la lista de características. El comportamiento de la herramienta software a desarrollar se regirá por la interacción de los casos de uso de los usuarios y la misma. Cada tipo de usuario tendrá su caso de uso, describiéndose sus posibles acciones y funcionalidades en la aplicación.

5.2.1 Actores del sistema

La herramienta será utilizada por dos tipos de actores y sus respectivos casos de usos. Estos actores representan los dos tipos de usuarios que se describen a continuación:

- **Usuario Acuicultura**: No tendrá acceso a todas las funcionalidades del sistema, depende del actor Administrador Acuicultura para cumplir su función.
- **Administrador Acuicultura**: Es el actor con acceso a todas las funcionalidades del sistema, administrando y confeccionando el producto con el cual trabajará el actor Usuario Acuicultura.

La gráfica a continuación describe los actores del sistema y fue confeccionada con la herramienta StarUML.

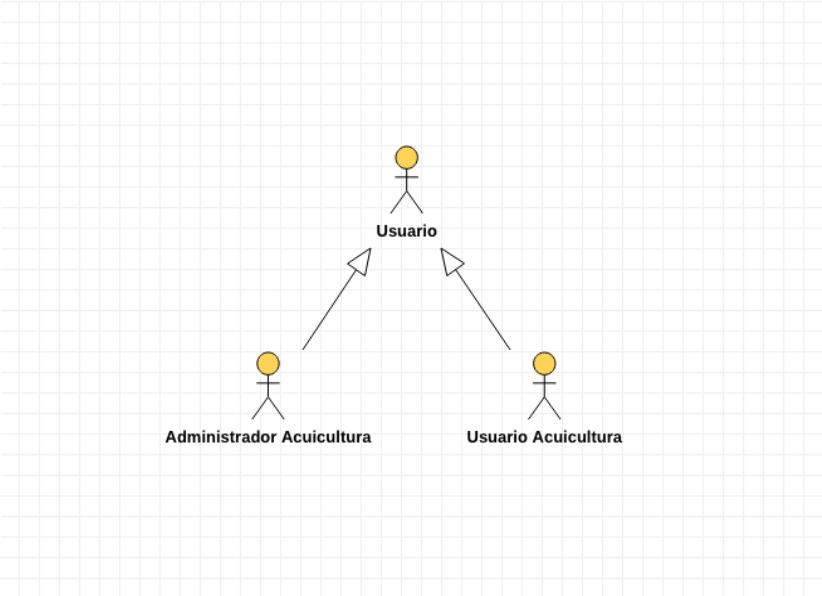


Figura 15.Diagrama de actores del sistema

5.2.2 Descripción de casos de usos

Los siguientes diagramas muestran las interacciones de cada caso de uso con la herramienta software a desarrollar. Se utilizó para la confección de estos diagramas la aplicación StarUML para dar cumplimiento al presente requisito.

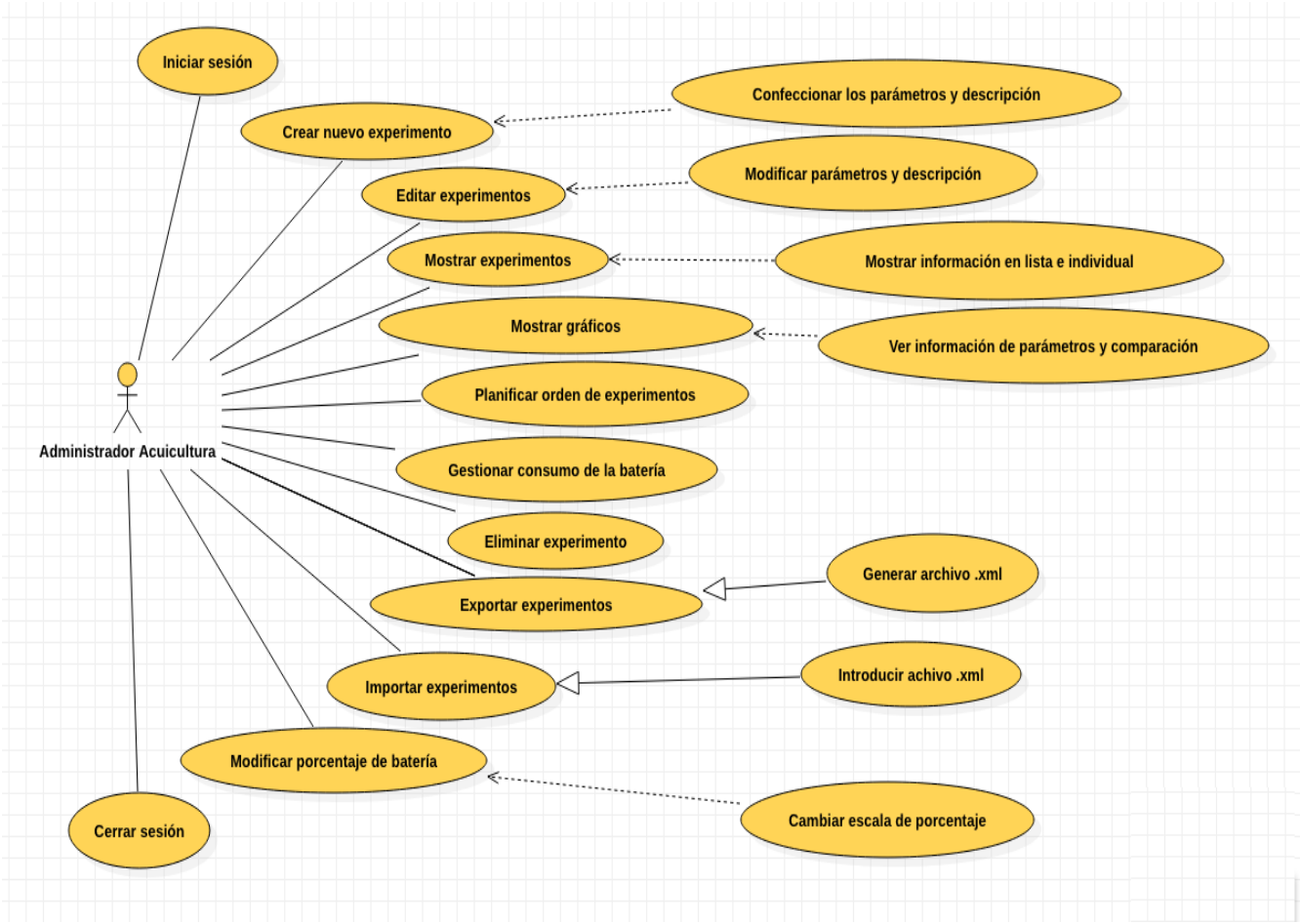


Figura 16.Diagrama casos de uso Administrador Acuicultura

Se puede observar las funcionalidades en las interacciones de los usuarios Administrador Acuicultura y Usuario Acuicultura en la aplicación, y las diferencias entre los mismos. La principal diferencia es que el actor o Usuario Acuicultura no puede crear experimentos, solo puede trabajar con los creados por el Administrador.

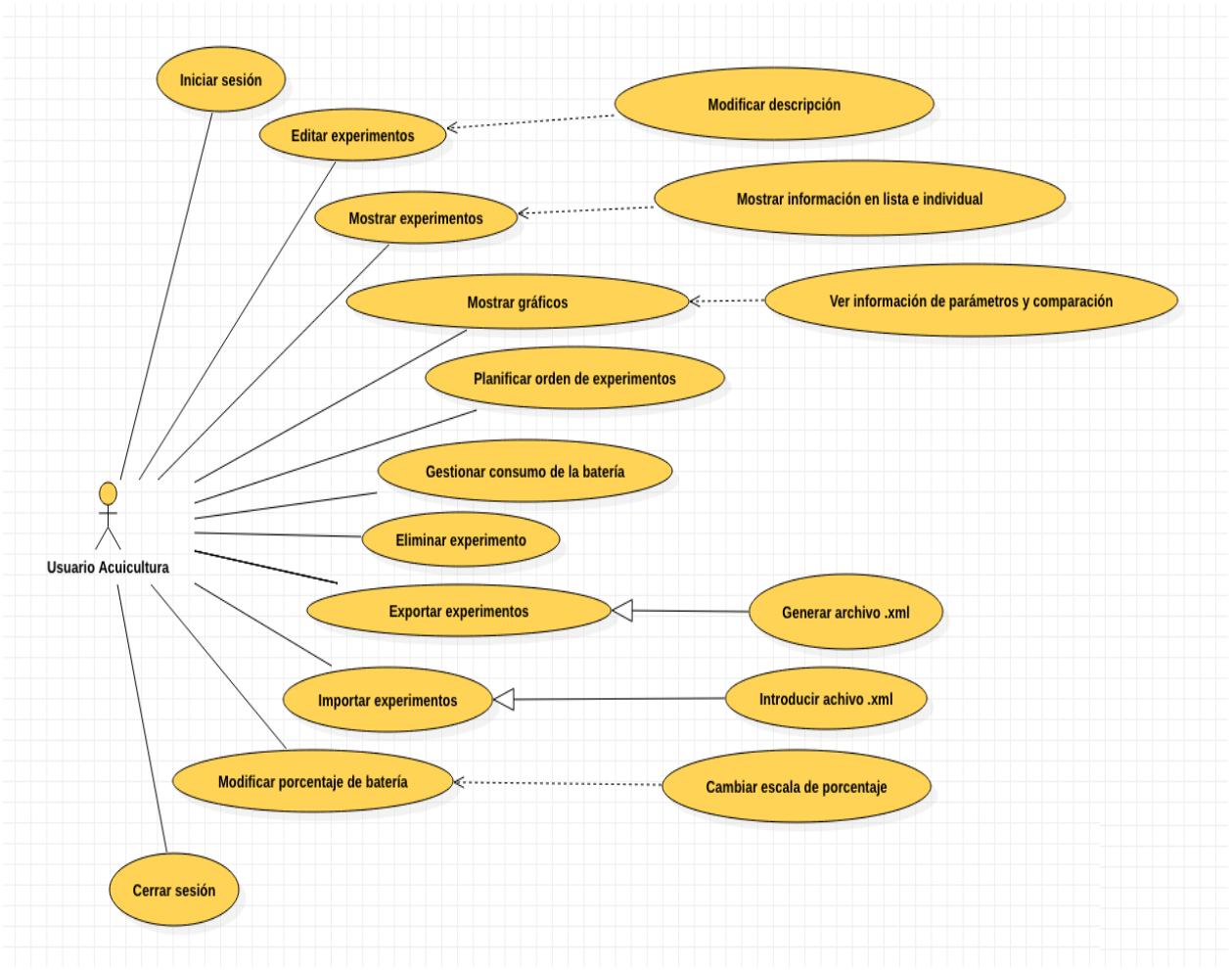


Figura 17. Diagrama casos de uso Usuario Acuicultura

5.3 Diseño

Para el diseño de la herramienta software se elaboró el prototipo de interfaz de usuario como visión gráfica del software a desarrollar, el patrón de diseño arquitectónico por el cual se rige, fluye y estructura sus partes, clases y módulos; así como la arquitectura global de los elementos de procesamiento que componen el sistema, para la implementación y despliegue de la herramienta.

5.3.1 Diseño del prototipo de interfaz de usuario

El diseño de la aplicación se ha implementado con la herramienta de prototipado de interfaces de usuario inVision, la cual se describe en el capítulo 4 epígrafe 4.2. Esta herramienta nos proporcionó las plantillas y recursos necesarios en la creación del diseño de cada parte de la aplicación en cuestión. Estos diseños iniciales son el punto de partida de cómo se observará gráficamente la aplicación final.

A continuación, se ilustran los diseños iniciales de las partes de la aplicación:

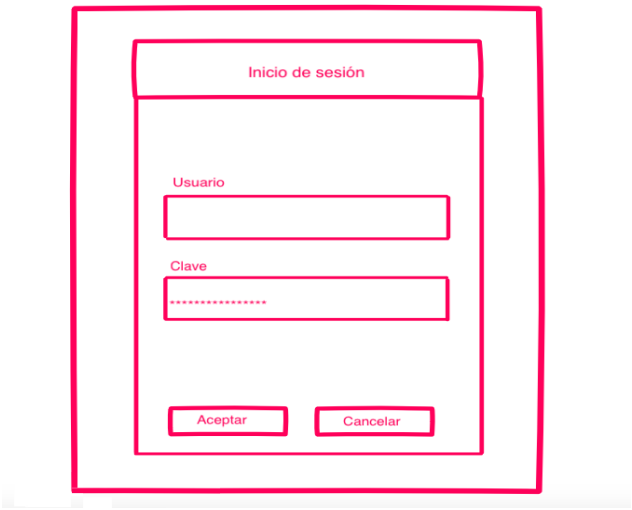


Figura 18. Diseño inicio sesión

El primer prototipo muestra el inicio de sesión para cada uno de los usuarios del sistema, su nombre de usuario y clave de acceso.

En la próxima figura se observa el diseño de la interfaz gráfica de trabajo, posterior al acceso de cada uno de los usuarios. Esta difiere gráficamente muy poco dependiendo del actor que la utiliza, por eso se muestra el diseño general y más completo de la misma. En este prototipo se encuentra algunos botones en la parte superior, el área de trabajo, planificación y gestión de los experimentos. También la lista de experimentos, descripción y botón para generar un nuevo experimento, papelera para eliminar experimento, gráficas de batería, circular y de barras.

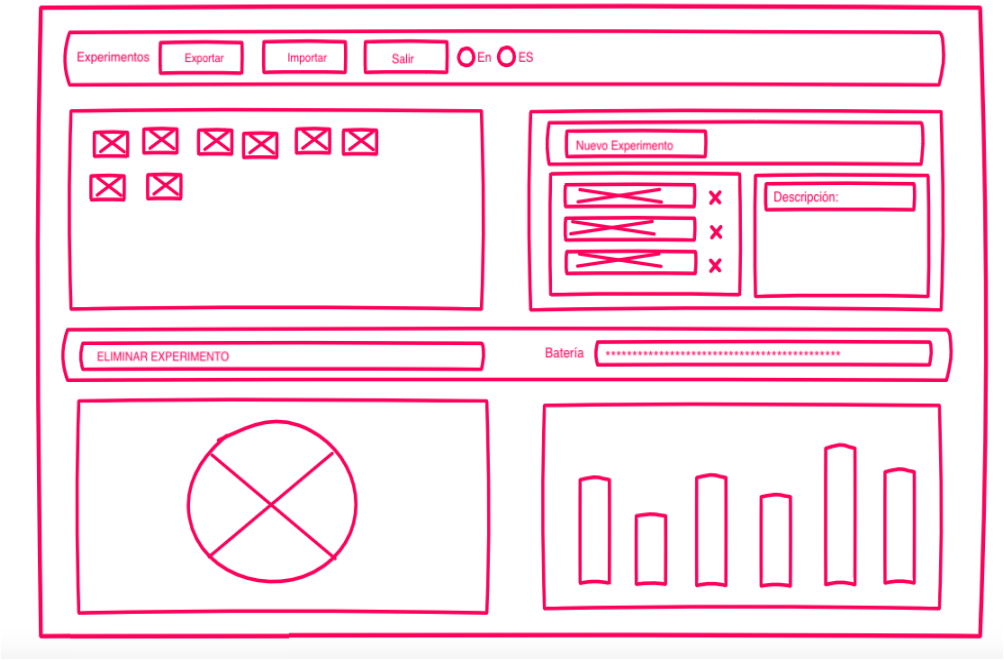


Figura 19.Diseño de área de experimentación en acuicultura

Una vez pulsado el botón de nuevo experimento o introduciéndonos en cualquier experimento, se mostrará una ventana como se observa en la siguiente figura. La ventana referente a la configuración de experimentos contiene los parámetros propios de cada experimento y una descripción de los mismos. Los parámetros solo pueden ser configurados por el Usuario Administrador, ya sea nombre, etiqueta, color de etiqueta, tiempo y energía de ejecución, tiempo y energía de espera. Los operandos 1 y 2 solo se mostrarán en el diseño, su funcionalidad será activada en una versión posterior de la herramienta.

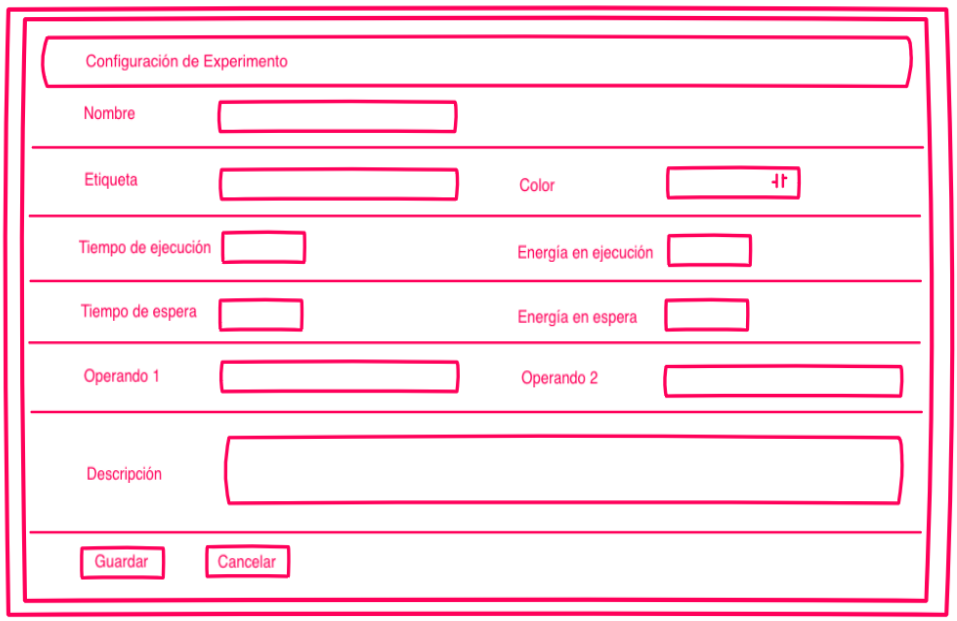


Figura 20. Diseño de ventana de configuración experimento

5.3.2 Patrón de diseño arquitectónico de la aplicación

Para el desarrollo de la herramienta software se utilizará el framework ZK, anteriormente definido en los capítulos 2 y 4. Este permite diseñar aplicaciones con dos tipos de arquitecturas, MVC (Model –View-Controller) y MVVM (Model-View-ViewModel). La aplicación a desarrollar tendrá un patrón de diseño MVVM, el cual es novedoso y posibilita una serie de ventajas importantes [225].

La arquitectura se conforma en tres partes:

- **El Modelo o Model** contiene las clases con los datos e información de la aplicación, además de las reglas de negocio.

- **La Vista o View** es la capa de interfaz gráfica de usuario, definida en el archivo .zul con los componentes ZK. La interacción de un usuario con componentes desencadena eventos que se envían al ViewModel.

- **VistaModelo o ViewModel** es responsable de exponer los datos del Modelo a la Vista y de proporcionar las acciones y la lógica de negocio necesaria para la interacción de los usuarios desde la Vista. Es una abstracción de la Vista conteniendo su estado y comportamiento, pero no debe tener referencias a los componentes de la interfaz gráfica de usuario; el propio framework se encarga del control de la comunicación y sincronización entre la Vista y VistaModelo. También es una abstracción del modelo extrayendo los datos necesarios para mostrarse en la Vista desde una o más clases de Modelo. Esos datos se exponen a través del método getter y setter como la propiedad de JavaBean.

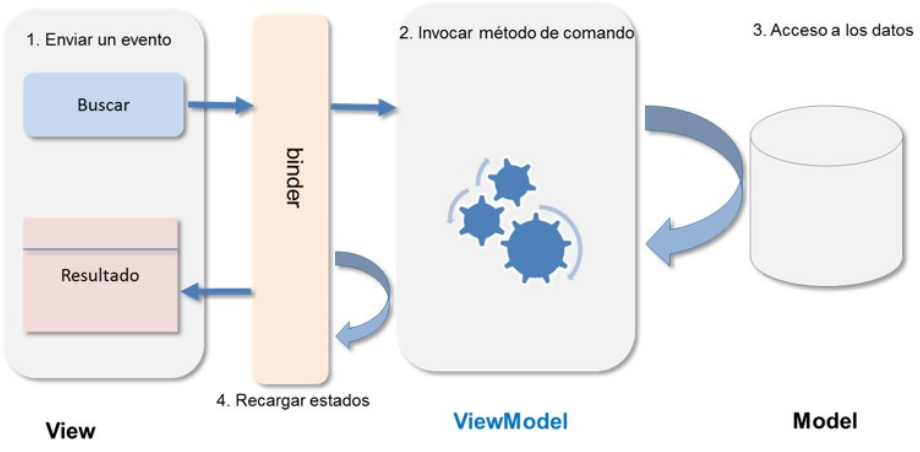


Figura 21.Flujo de patrón de diseño MVVM [258]

La figura muestra las partes y funcionamiento de un ejemplo de aplicación con patrón de diseño arquitectónico MVVM, desarrollada con el framework ZK. Su flujo de ejecución tiene los siguientes pasos:

- El usuario en View hace clic en el botón “buscar” y envía un evento correspondiente.
- El “binder” invoca el método de comando correspondiente en la capa ViewModel y lo ejecuta.
- El método accede a los datos de la capa Model y actualiza las propiedades correspondientes de la capa ViewModel, notificando los cambios.
- El “binder” carga las propiedades modificadas de ViewModel y recarga o actualiza el estado de los componentes correspondientes de la interfaz gráfica de usuario.

El **“Binder”** es el elemento del patrón de diseño MVVM encargado de la sincronización de datos entre View y ViewModel, garantizando que cualquier cambio realizado en los componentes de la GUI de una Vista se transfieran automáticamente al ViewModel y viceversa. Los desarrolladores de aplicaciones solo tienen que definir la relación de enlace de datos entre el atributo del componente UI y el objeto de destino, normalmente es un ViewModel, por expresión de anotación de enlace de datos [225].

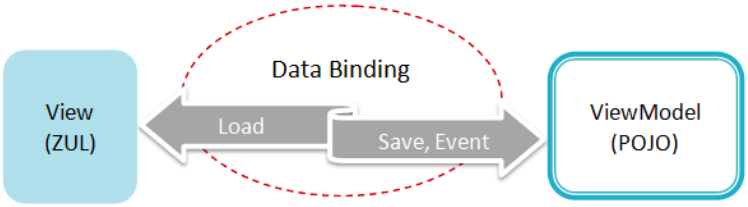


Figura 22. Flujo del Data Binding [259]

Como resultado de la comparación del patrón de diseño MVVM y el tradicional MVC obtenemos el siguiente esquema:

Tabla 5. Comparativa de patrones de diseño

ASPECTO	MVC	MVVM
Acoplamiento con vista	Muy poco con plantilla	Muy poco
Acoplamiento con componente	Un poco	Muy poco
Codificación en vista	Mediante el ID del componente	A través de una expresión “Data binding”
Implementación del controlador	Extendemos ZK’s Composer	POJO (Plain Old Java Object)
Acceso a datos de interfaz de usuario	Acceso directo	Automático
Acceso a datos de fondo	Acceso directo	Acceso directo
Actualización de GUI	Manipulamos directamente los componentes	Automático (@NotifyChange)
Componente que controla la granularidad	Elevado, control total	Normal
Actuación	Alto	Normal

Las mayores diferencias entre ambos patrones de diseño es que el Controlador en el MVC cambia a ViewModel en MVVM. ViewModel no contiene ninguna referencia a los componentes de la interfaz de usuario, desconociendo los elementos visuales de la Vista. En MVVM el "binder" es quien sincroniza los datos en lugar de un Controller. MVC puede resultar más intuitivo porque se controla directamente los componentes que se observan en la Vista, pudiendo crear componentes secundarios dinámicamente y controlar componentes personalizados. Ambas arquitecturas tienen como semejanza que la Vista y el Modelo conservan las mismas características.

Ventajas MVVM:

- En la programación de **diseño por contrato** es muy adecuado utilizarlo. Mientras se realice el contrato (qué datos mostrar y qué acciones realizar), el diseño y la codificación de la interfaz de usuario de ViewModel pueden realizarse en paralelo e independientemente. Cualquiera de los lados no bloqueará el camino del otro.
- **Acoplamiento suelto con Vista**: El diseño de la interfaz de usuario se puede cambiar fácilmente de vez en cuando sin modificar el ViewModel siempre que el contrato no cambie.
- **Mejor comprobabilidad**: Como ViewModel no "ve" la capa de presentación, los desarrolladores pueden probar la clase de ViewModel fácilmente sin elementos de la interfaz de usuario.
- **Mejor reutilización**: Será más económico diseñar diferentes Vistas para diferentes dispositivos con un ViewModel común en un diseño responsivo. Esto permite desarrollar una sola aplicación, que se adapta por si sola a las diferentes pantallas de los terminales.

5.3.3 Arquitectura del sistema

Los elementos principales que conforman la arquitectura del sistema de la aplicación web son:

- **Máquina Virtual Java**: es un elemento fundamental de la plataforma de Java, ejecutándose en un ordenador. Esta interpreta código binario Java(.class) y lo traduce a código nativo de la plataforma final.
- **XML**: Lenguaje de Marcado Extensible es un subconjunto de SGML que posibilita definir gramaticalmente lenguajes de marcado adaptado a usos específicos. Su extensión de archivo es .xml.
- **Navegador**: Es el cliente e interpretan peticiones del usuario solicitando recursos a al servidor web y presentan los resultados al mismo. Los principales son Google Chrome, Edge, Mozilla Firefox y Safari.
- **Protocolo HTTP**: Es el protocolo basado en TCP/IP que se utiliza para que el navegador realice las peticiones al servidor web y este responda.
- **Widgets ZK**: Son los objetos con apariencia visual de la interfaz gráfica, con los que interactúa el usuario. Estos objetos JavaScript, manejan los eventos que se ejecutan en el cliente. Cada componente que se ejecuta en el servidor está asociado con un widget [225].

- **AJAX**: *Asynchronous JavaScript And XML* es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). 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, mejorando la interactividad, velocidad y usabilidad en las aplicaciones [225].
- **Componentes ZK**: Son objetos Java que se ejecuta en el servidor y representa un objeto UI que puede ser manipulado por una aplicación Java. Un componente tiene todo el comportamiento de un objeto IU, excepto que no tiene una parte visual [225].
- **Servidor Web**: Recibe peticiones de clientes (navegadores) y responde a las peticiones enviando un recurso o notificando un error si no existe el recurso. Se utilizará WildFly como servidor web para la aplicación Java a desarrollar.
- La **aplicación ZK**: Se ejecuta en el servidor accediendo a los recursos de back-end, ensamblando la interfaz de usuario con componentes. Además, escucha las interacciones del usuario para luego manipular los componentes y actualizar la interfaz gráfica de usuario. La aplicación tiene acceso a todas las pilas de tecnología Java. Las interacciones del usuario, como Ajax y Server Push, se resumen en los objetos de eventos. La interfaz de usuario está compuesta por componentes similares a POJO (Plain Old Java Object) [225].

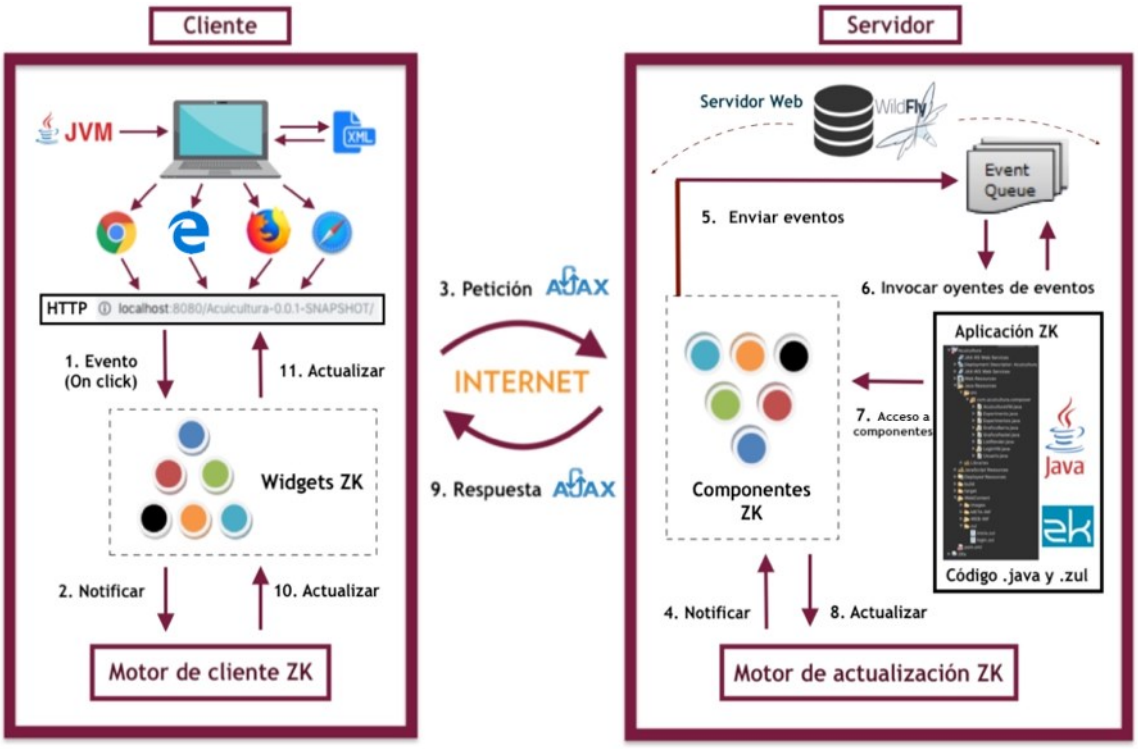


Figura 23. Flujo de arquitectura del sistema

Flujo de ejecución:

1. El flujo de ejecución se inicia desde un widget o la aplicación. Esto suele deberse a la actividad del usuario (o al requisito de la aplicación) y se realiza mediante la publicación de un evento del lado del cliente en un widget.
2. Luego, el evento escala por el árbol DOM (Document Object Model) hasta el padre del widget, el padre del padre y, finalmente, el motor de cliente ZK. El motor de cliente ZK luego decide si y cuándo enviar el evento de vuelta al servidor en una solicitud de Ajax.
3. Si es necesario, el motor de cliente ZK enviará una petición Ajax al motor de actualización ZK en el servidor.
4. Al recibir solicitudes de Ajax, el motor de actualización de ZK invocará para manejar una solicitud.
5. Cómo se puede manejar la solicitud es realmente con un componente. Pero, el componente que maneja la solicitud generalmente actualiza los estados, si es necesario, y luego notifica a la aplicación mediante la publicación de eventos en la ejecución actual.
6. Si se publica algún evento, el motor de actualización ZK los procesará uno por uno invocando a los oyentes del evento.
7. El detector de eventos, proporcionado por una aplicación, puede elegir entre actualizar los recursos de back-end o los componentes o publicar otros eventos.
8. Finalmente, el motor de actualización de ZK recopila todas las actualizaciones de los componentes, incluidos los cambios de estado, la conexión y el desapego, para su optimización.
9. Luego envía una colección de comandos al cliente como repuesta AJAX.
10. El motor de cliente ZK evalúa cada uno de estos comandos para actualizar los widgets en consecuencia.
11. Luego, los widgets actualizarán el árbol DOM del navegador para que estén disponibles para el usuario.

La **arquitectura Fusion Server + client de** ZK permite la sincronización de los estados de los componentes entre el navegador y el servidor de forma automática y transparente a la aplicación. El motor de cliente ZK y el motor de actualización Zk simplifican la implementación funcionando de manera conjunta con seguridad, eficiencia y robustez. Se puede agregar funcionalidad opcional del lado del cliente para una mayor interactividad, como el manejo de eventos, la personalización de efectos visuales o incluso la composición de la interfaz de usuario sin la codificación del lado del servidor. ZK permite fusiones sin interrupciones que van desde centradas exclusivamente en el servidor hasta centradas en el cliente de manera productiva y flexible [225].

5.4 Implementación

Se cuenta con todas las herramientas, recursos y conocimientos básicos para comenzar a desarrollar la herramienta de programación y estimación del consumo energético para un DAE. Toda la implementación se realiza en el entorno de desarrollo integrado Eclipse,

empleando el lenguaje de programación Java para la lógica de negocio y el lenguaje de marcado ZUML del framework ZK para describir la interfaz gráfica de usuario.

La aplicación se rige por la arquitectura y patrón de diseño anteriormente analizados. Se compone de archivos .zul para la interfaz gráfica de usuario y .java para la lógica de negocio y datos. La aplicación se divide en dos partes como se representada en el siguiente árbol y en el orden en que fue desarrollada. Una primera parte para el inicio de sesión de los usuarios y una segunda parte para el área de trabajo con los experimentos de acuicultura, en la programación y estimación del consumo energético del DAE.

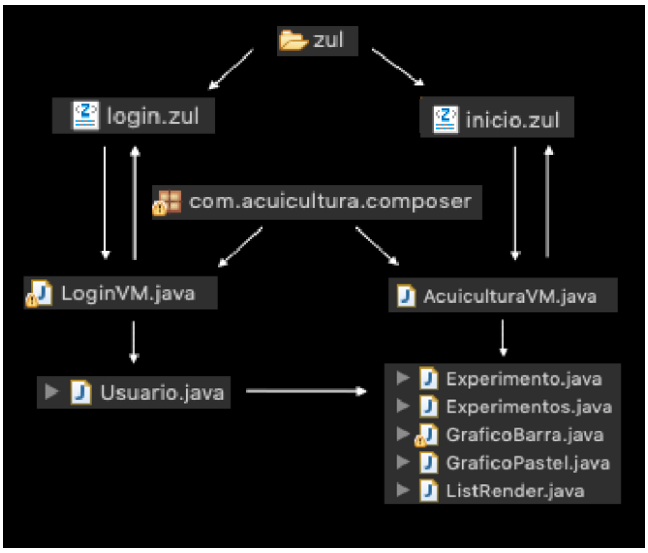


Figura 24. Archivos de la aplicación

5.4.1 Construcción de la interfaz gráfica de usuario “Inicio de sesión”

El diseño de interfaz de usuario es un buen comienzo para crear la aplicación, ya que nos ayuda a definir el alcance de la aplicación. ZK proporciona cientos de componentes de interfaz de usuario preparados para construir de manera rápida aplicaciones, combinando componentes sin tener que crearlos desde cero.

En ZK se emplea ZK User Interface Markup Language (ZUML), un lenguaje con formato XML, para describir la IU. Según la convención de ZK, los archivos que describen la interfaz de usuario con ZUML usan **.zul** como sufijo del nombre. En los archivos zul, un componente se puede representar como un elemento XML (etiqueta) y se puede configurar su comportamiento, estilo y función como atributos.

En la aplicación se creó un archivo login.zul para la interfaz gráfica de inicio de sesión, que a continuación se muestra su código y componentes.

```
<?taglib uri="http://www.zkoss.org/dsp/web/core" prefix="c"?>

<window contentStyle="vertical-align: bottom;" height="100%"
    apply="org.zkoss.bind.BindComposer"
    viewModel="@id('vm') @init('com.acuicultura.composer.LoginVM')">

    <hbox height="30%"></hbox>
    <div align="center" height="100%" width="100%">
        <groupbox width="250px" mold="3d" closable="false"
            style="align:center">
            <caption label="Inicio de sesión"></caption>
            <grid style="border: 0px;">
                <columns>
                    <column width="100%"></column>
                </columns>
                <rows>
                    <row>
                        <grid>
                            <columns>
                                <column width="100%"></column>
                            </columns>
                            <rows>
                                <row>
                                    <label value="${labels.user }"></label>

                                </row>
                                <row>
                                    <textbox width="100%"
                                        value="@bind(vm.usuario.user)">
                                    </textbox>
                                </row>
                                <row>
                                <label value="${c:l('pass')}"></label>

                                </row>
                                <row>
                                    <textbox width="100%"
                                        type="password"
                                        value="@bind(vm.usuario.pass)">
                                    </textbox>
                                </row>
                                <row>
                                    <label value="@load(vm.mensaje)"
                                        style="color: red;">
                                    </label>
                                </row>
                            </rows>
                        </grid>

                    </row>

                </rows>
            </grid>
            <separator></separator>
            <hbox>
                <button label="Aceptar" onClick="@command('login')"></button>
                <button label="Cancelar" onClick="@command('cancelar')"></button>
            </hbox>
        </groupbox>
    </div>
</window>
```

Figura 25. Archivo login.zul

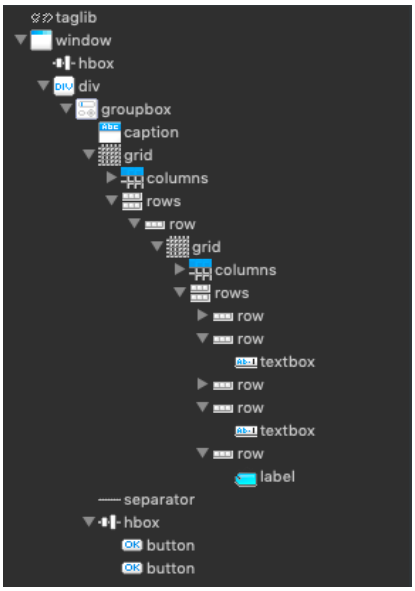


Figura 26. Componentes gráficos del archivo login.zul

Como *"window"* es el componente más externo, se llama *componente raíz*. *"Window"* es un contenedor de uso común, un elemento de visualización básico de la aplicación e incluye otros componentes. Todos los otros componentes dentro de la *ventana* se llaman *componentes secundarios*, colocándose en el cuerpo de la etiqueta de la ventana. Aquí se especifica un atributo "id" para algunos componentes que le permiten controlarlos al hacer referencia a su *id*. Se usa CSS como la sintaxis "100%" para el atributo "altura", el atributo "estilo" y otros. En un elemento HTML, se puede escribir directamente las reglas CSS como el valor del atributo. El *hbox* se emplea como componente de diseño para organizar sus componentes secundarios en una fila horizontalmente. Su atributo *align* controla la alineación vertical. Algunos componentes como *grid* admiten componentes secundarios limitados, observándose las relaciones jerárquicas entre estos, por ejemplo, las *filas* solo pueden contener *filas* y sólo se puede colocar *columna* dentro de *las columnas*.

Para permitir que los usuarios inicien sesión, se necesita dos componentes *textbox* (cuadro de texto). Uno para ingresar su nombre de usuario y otro para ingresar palabra clave tipo "password". Los componentes *textbox* tienen como atributos el ancho, tipo y el valor que conecta mediante anotación @bind con la lógica de negocio para notificar el evento seleccionado. Las etiquetas de cada textbox tienen como atributo su valor y existe una etiqueta especial para cuando el usuario o clave son incorrectos, mostrando un mensaje de error de autenticación en la aplicación.

Además, se necesitan los botones para activar la entrada o cancelarla. Estos componentes tienen como atributos sus etiquetas y eventos "onClick" asociados por anotaciones @command a implementar por la lógica de negocios.

La apariencia de la interfaz gráfica de usuario muestra sus componentes alineados verticalmente, un título en la parte superior central de la ventana, dos textbox con sus respectivas etiquetas y los botones de aceptar y cancelar. A continuación, se muestra el resultado.

Figura 27. Inicio de sesión

Enfoque MVVM

El framework ZK controla por sí mismo la interacción del usuario sobre los componentes de la vista siguiendo el patrón de diseño **Modelo-Vista-VistaModelo (MVVM)**. Para ello se tiene conformado las siguientes partes de la aplicación:

- **Modelo** consta de los datos para el inicio de sesión por tipo de usuario y las reglas de negocio. Está compuesto por la clase **Usuario** dentro del archivo Usuario.java.
- **Vista** es la interfaz de usuario que contiene los componentes de ZK, creada en el archivo login.zul. La interacción del usuario con los componentes de la vista genera eventos que son enviados a VistaModelo, en este caso LoginVM.
- **VistaModelo** expone la información de la clase Usuario a la Vista login.zul, procesando los eventos que la Vista solicite al interactuar el usuario con la misma. Está conformada por la clase LoginVM que contiene la lógica de negocio y métodos de comandos en el archivo LoginVM.java.

La capa de LoginVM.java es una abstracción de la **Vista** login.zul, porque en el diseño de esta capa, parte del análisis de las funcionalidades de la interfaz de usuario (**Vista**), identificando el estado de los componentes y el comportamiento que esperamos de los mismos como se muestra a continuación.

Estado:

- Nombre de usuario introducido
- Clave de usuario correspondiente introducida
- Resultado del inicio de sesión

Comportamiento:

- Iniciar sesión
- Cancelar inicio de sesión

LoginVM es también una abstracción del **Modelo**, tiene 6 variables para los estados y 2 métodos para el comportamiento. La creación de esta capa se elabora como propiedades de un JavaBean (POJO), a través de métodos setters y getters. El método login para realizar el inicio de sesión implementa una serie de condicionales comparando los datos introducidos por el usuario con el de nombre de usuario y clave predefinidos para cada tipo de sesión de usuario. El segundo método se implementa para cancelar el inicio de sesión, donde usuario toma el valor de nuevo Usuario, actualizando sus propiedades iniciales correspondientes.

A continuación, se muestran los archivos creados VistaModelo LoginVM.java y el Modelo Usuario.java respectivamente.

```java
package com.acuicultura.composer;

import java.util.Locale;

public class LoginVM {

    private Usuario usuario = new Usuario();
    private final String USER_ADMIN = "admin";
    private final String PASS_ADMIN = "admin*1";

    private final String USER = "user";
    private final String PASS_USER = "user*1";
    private String mensaje;

    @AfterCompose
    public void afterCompose(@ContextParam(ContextType.VIEW) Component view) {
        Selectors.wireComponents(view, this, false);
        Selectors.wireEventListeners(view, this);
        inicializar();
//      Locale prefer_locale = new Locale("es");
//      Executions
//              .getCurrent()
//              .getSession()
//              .setAttribute(org.zkoss.web.Attributes.PREFERRED_LOCALE,
//                      prefer_locale);
//      Executions.sendRedirect("");
    }

    private void inicializar() {

    }

    @NotifyChange("mensaje")
    @Command
    public void login()
    {
        if (usuario.getUser().equals(USER_ADMIN)
                && usuario.getPass().equals(PASS_ADMIN)) {
            usuario.setAdmin(true);
            Executions.getCurrent().getSession()
                    .setAttribute("usuario", usuario);
            Executions.sendRedirect("/");
        } else if (usuario.getUser().equals(USER)
                && usuario.getPass().equals(PASS_USER)) {
            Executions.getCurrent().getSession()
                    .setAttribute("usuario", usuario);
            Executions.sendRedirect("/");
        } else
            mensaje = "Error de Autenticación";
    }

    @NotifyChange("usuario")
    @Command
    public void cancelar() {
        usuario = new Usuario();
    }

    public Usuario getUsuario() {
        return usuario;
    }

    public String getMensaje() {
        return mensaje;
    }
}
```

Figura 28. Archivo loginVM.java

```java
package com.acuicultura.composer;

public class Usuario {
    private String user = "";
    private String pass = "";
    boolean admin = false;

    public Usuario() {

    }

    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getPass() {
        return pass;
    }

    public void setPass(String pass) {
        this.pass = pass;
    }

    public boolean isAdmin() {
        return admin;
    }

    public void setAdmin(boolean admin) {
        this.admin = admin;
    }
}
```

Figura 29. Archivo Usuario.java

Anotaciones y librerías

Las anotaciones en ZK son coherentes con el estilo de anotación de Java, estas expresiones se han utilizado para conectar y sincronizar los componentes de la Vista con VistaModelo. Con ellas se establecen las relaciones de cambios realizados en los datos de los componentes de una IU, transfiriéndose automáticamente al ViewModel en función de una relación de enlace predefinida. El "binder" en ZK juega un papel fundamental de intermediario en todo el mecanismo, controlando por sí mismo los componentes, sincronizando los datos, manejando los eventos y respondiendo en ambos sentidos la relación entre Vista y VistaModelo o VistaModelo y Vista.

A continuación, se exponen las anotaciones y librerías que contienen los archivos de las figuras anteriores 25 y 28 respectivamente. Para una mejor comprensión de las anotaciones se le asignan las viñetas (▪) a las de la Vista y (➢) a las de la VistaModelo.

Archivo.zul:

- "@id('vm') @init('com.acuicultura.composer.LoginVM')"
- "@bind(vm.usuario.user)"
- "@bind(vm.usuario.pass)"
- "@load(vm.mensaje)"
- "@command('login')"
- "@command('cancelar')"

Archivo.java:

- ➢ @AfterCompose
- ➢ @ContextParam
- ➢ @NotifyChange("mensaje")
  @Command
  - public void login()
- ➢ @NotifyChange("usuario")
  @Command
  - public void cancelar()

Librerías:

1. java.util.Locale :
   - import org.zkoss.bind.annotation.AfterCompose
   - import org.zkoss.bind.annotation.Command
   - import org.zkoss.bind.annotation.ContextParam
   - import org.zkoss.bind.annotation.ContextType
   - import org.zkoss.bind.annotation.NotifyChange
   - import org.zkoss.zk.ui.Component
   - import org.zkoss.zk.ui.Executions
   - import org.zkoss.zk.ui.select.Selectors

Enlaces y funcionamientos de los componentes gráficos

- **<window** *viewModel*= **"@id('vm') @init('com.acuicultura.composer.LoginVM')">**

Aplicamos a la ventana (componente global) un "composer" llamado "**org.zkoss.bind.BindComposer**" que procesa las expresiones de "data binding" e inicializa la clase del ViewModel Login.VM . El @id () es usado para establecer el ID del ViewModel LoginVM, haciendo referencia a las propiedades de esta capa (por ejemplo vm.usuario) como expresión de "data binding". Se provee un nombre de clase completamente cualificado para @init (), inicializándose el objeto VistaModelo. Luego de vincular el LoginVM al componente, todos sus componentes secundarios pueden acceder al mismo ViewModel y sus propiedades.

- **<textbox** *value*= **"@bind(vm.usuario.user)">**
  <textbox *value*= **"@bind(vm.usuario.pass)">**

Utilizamos esta expresión @bind para vincular datos con parámetros, para cargar y guardar. Las variables declaradas para los estados de los componentes en la clase LoginVM de la capa de VistaModelo, se asocian a los atributos de los componentes de la Vista login.zul. ZK sincroniza automáticamente por si sola la información contenida en el atributo y la propiedad correspondiente en la capa de LoginVM.java. Asociamos la capa VistaModelo con el componente y todos sus componentes hijos pueden acceder a las propiedades de la capa VistaModelo asociada al componente padre. La función de iniciar sesión tiene 2 estados relacionados con ella para ser almacenados en la capa de VistaModelo. Primero guardamos el valor del primer "textbox" en la propiedad "usuario" de la capa de VistaModelo y vinculamos el atributo "value" del "textbox" a la propiedad del objeto en la capa VistaModelo con "@bind(vm.usuario.user)". Segundo, asociamos el atributo "value" del segundo "textbox" a la propiedad del objeto en LoginVM.java simplemente con "@bind(vm.usuario.pass)".

- **<label** *value*= **"@load(vm.mensaje)">**

Se utiliza enlazando datos y comandos junto con parámetros para cargar datos de destino. Esta etiqueta con estilo de color rojo es un componente creado para denotar un error de autenticación en el inicio de sesión. Tiene como valor la propiedad de mensaje en la clase LoginVM del archivo correspondiente.

- **<button** *label*= **"Aceptar"** *onClick*= **"@command('login')">**
  <button *label*= **"Cancelar"** *onClick* = **"@command('cancelar')">**

Los comportamientos requeridos por la Vista login.zul se crean como comandos con la anotación de tipo @Command sobre el método en la capa de VistaModelo LoginVM.java. Enlazamos el evento "onClick" de los componentes (botones) de la vista con sus respectivos comandos (métodos) en la VistaModelo, para que ZK encuentre e invoque los métodos cuando estos eventos sean lanzados desde la Vista. Se han marcado los métodos "login()" y "cancelar()" como comandos con sus nombres por defecto "login" y "cancelar", que son los mismo nombres de los métodos.

- **@NotifyChange(“mensaje”)**
  @Command
  - **public void login()**

Luego de asociar el evento “onClick”, cuando el usuario haga clic en el botón “Aceptar”, ZK invocará el método “login()” de comando en LoginVM.java, implementará sus condicionales para iniciar sesión, y de no ser posible para ambos tipos de usuarios, se recargará la propiedad “mensaje” como se ha indicado en la anotación: @NotifyChange en la capa ViewModel

- **@NotifyChange(“usuario”)**
  @Command
  - **public void cancelar()**

Cuando el usuario haga clic en el botón “Cancelar”, ZK invocará el método “cancelar()” de comando y recargará la propiedad “usuario”, indicado en la expresión @NotifyChange.

- **@AfterCompose**

En el archivo LoginVM.java la anotación tiene como propósito identificar un método de ciclo de vida que se invocará en doAfterCompose () de BindComposer en la Vista. Sólo se permite un método anotado @ AfterCompose como máximo en una clase ViewModel.

- **@ContextParam**

Es un parámetro de un método (para los métodos iniciales y de comando) y su propósito es decirle al “binder” que pase el objeto de contexto con el tipo especificado. Esta anotación se aplica al parámetro del método inicial (o comando). Los métodos pueden obtener varios objetos de contexto ZK como página o escritorio mediante la aplicación de anotaciones en los parámetros.

5.4.2 Construcción de la interfaz gráfica de usuario “Experimentos en Acuicultura”

En la segunda parte de la aplicación se creó un archivo inicio.zul para la interfaz gráfica de experimentación en Acuicultura. A continuación, se muestra el código y árbol de estructura de sus componentes gráficos.

Figura 22. Archivo inicio.zul (línea de código 1 hasta 120)

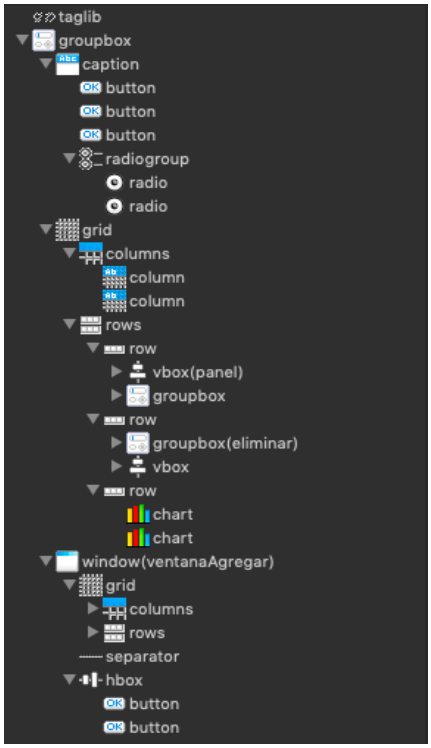


Figura 30. Componentes gráficos del archivo inicio.zul

Groupbox es el elemento global que se utiliza para agrupar todos los componentes gráficos de esta interfaz. *En él se incluye* un caption con la etiqueta que lleva por título "Experimentos", tres botones para exportar, importar y salir respectivamente. Además, dos radios para seleccionar los idiomas, que en un futuro se implementaran. Debajo se tiene un grid que contiene los principales componentes de la interfaz, organizados en dos columnas y tres filas.

En la primera columna y fila de izquierda a derecha se ubica un vbox con id("panel") que es el panel con características "Drag and Drop", que será el área de planificación y gestión de los experimentos. En la misma fila y segunda columna se encuentra un groupbox que contiene un caption con un botón etiquetado como "Nuevo Experimento", el cual genera el evento "onClick" para acceder a la venta de configuración de experimento que posteriormente se describirá. Debajo de este se tiene un grid de dos columnas y una fila que incluye en la primera columna un listbox de id ("lbTiposExperimentos") con características "droppable" y "draggable" para una lista de experimentos. En la segunda columna un groupbox con caption y textbox para la descripción de un experimento seleccionado por el usuario.

En la segunda fila, primera columna de la izquierda se encuentra un groupbox de id (eliminar) con propiedades "droppable" y una etiqueta de color rojo "ELIMINAR EXPERIMENTO". En esta fila, la siguiente columna contiene un hbox etiquetado como "Batería", una barra de progreso del consumo de la batería con id("curr_met") y un doublebox para cambiar el porcentaje de la misma.

La tercera fila está compuesta por un gráfico circular a la izquierda y un gráfico de barras a la derecha. Ambos tienen como propiedades el modelo a cumplir y su funcionamiento, que posteriormente se profundizará en los mismos.

Se utiliza las reglas CSS en la sintaxis de los componentes, agregando valores de diseño a sus atributos como por ejemplo el ancho, altura, estilo y otros.

La siguiente figura muestra la apariencia gráfica de UI, sus componentes organizados con sus etiquetas, botones, gráficos y áreas de experimentación.

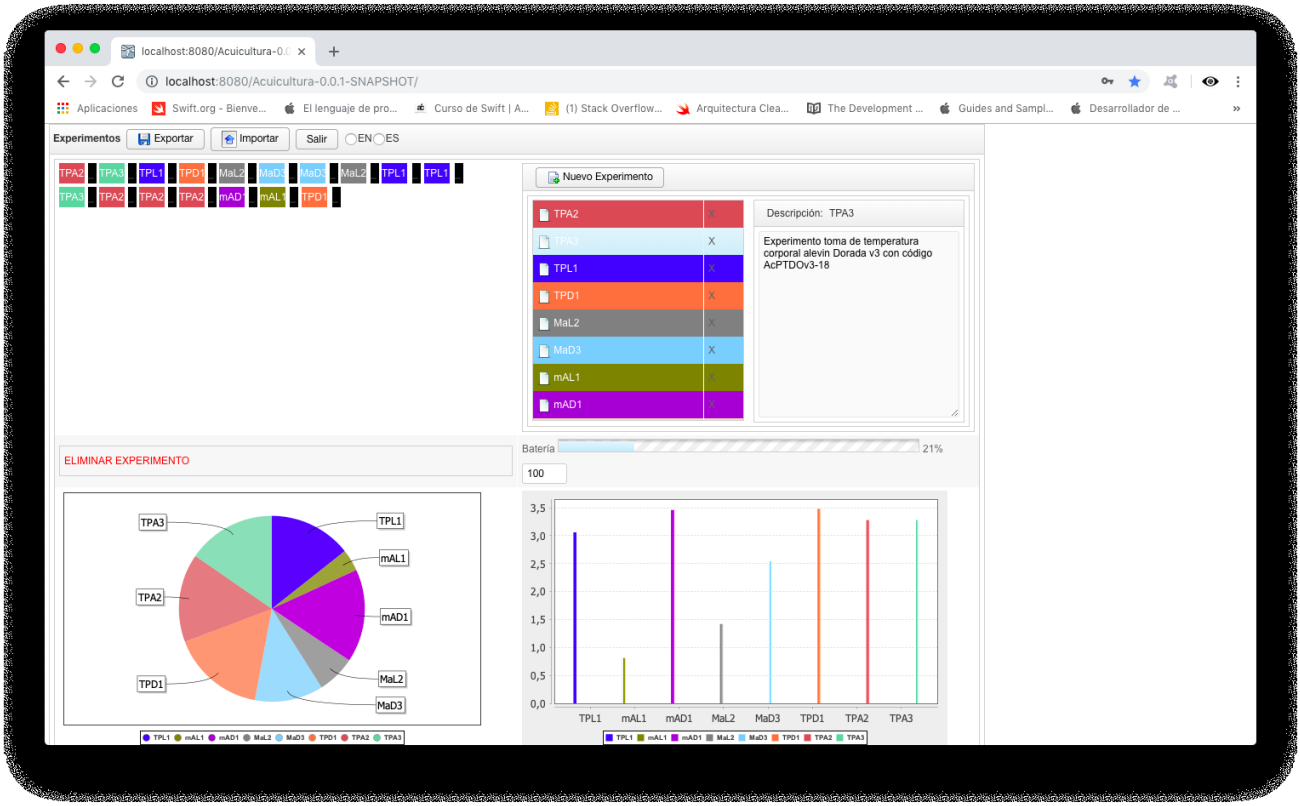


Figura 31. Interfaz gráfica de usuario Administrador Acuicultura

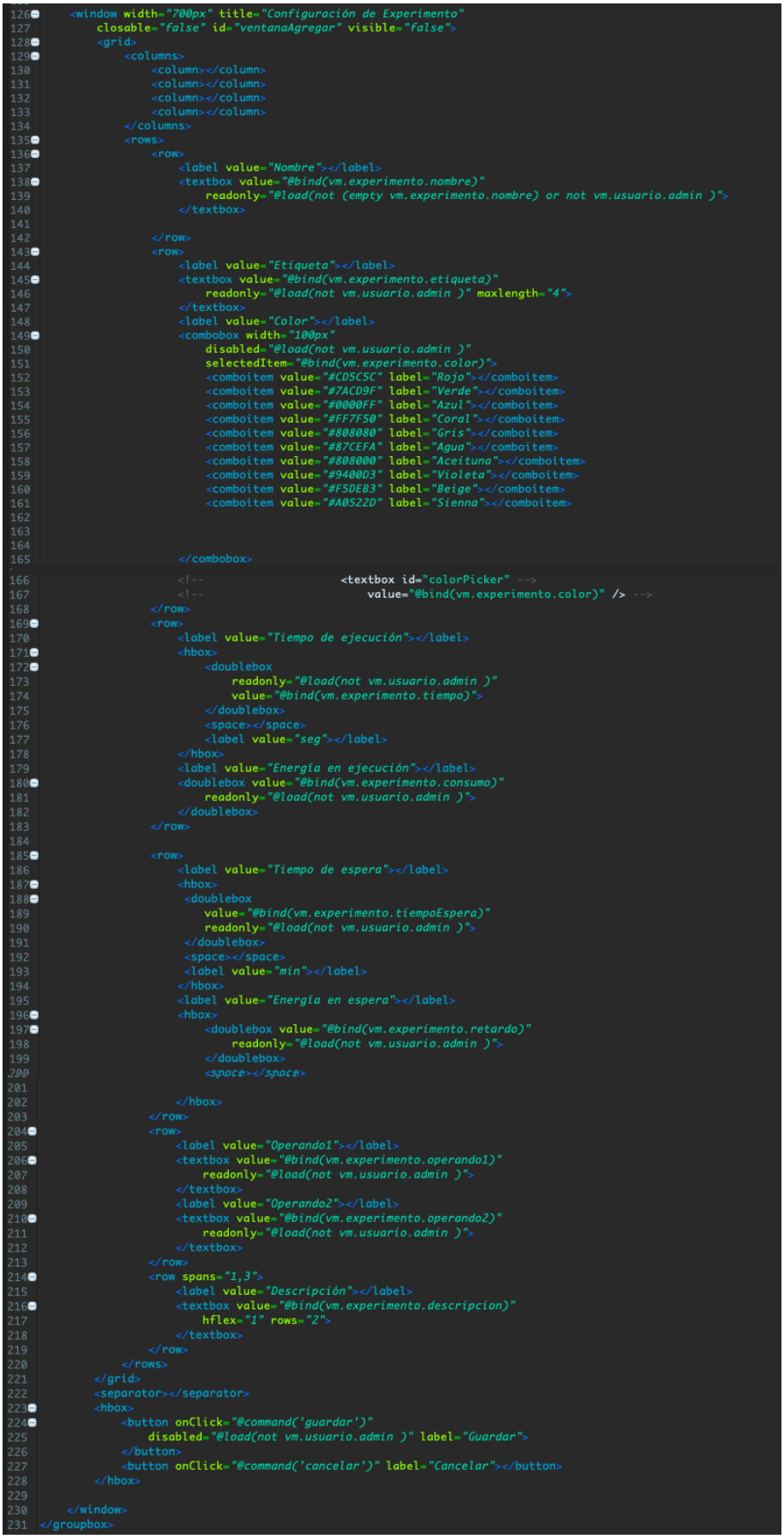


Figura 32. Archivo inicio.zul (línea de código 126 hasta 231)

Luego de generar el evento “onClick” del botón “Nuevo Experimento” se visualiza una venta con los elementos gráficos creados en la capa de Vista inicio.zul, como se muestra en la figura anterior. La ventana es el componente global que contiene como título “Configuración de Experimento”, el id “ventana agregar” y un grid de 4 columnas y 6 filas.

La primera fila y columna de izquierda a derecha, contiene una tiqueta “Nombre” y un textbox para guardar o cargar un nombre del experimento en la segunda columna.

La segunda fila y primera columna tiene una etiqueta “Etiqueta” y un textbox con un máximo de 4 caracteres para etiquetar al experimento en la segunda columna. Además, en la tercera y cuarta columna de la misma fila, se encuentra una etiqueta “Color” y un combobox para seleccionar colores respectivamente. El combobox tiene 10 valores de códigos de colores y sus respectivas etiquetas como por ejemplo rojo, verde, azul, etc.

En la tercera fila y primera columna está conformada por una etiqueta “Tiempo de ejecución”. La segunda columna tiene un hbox con un doublebox para el valor del tiempo y una etiqueta “seg” para la unidad de medida. La tercera columna representa una etiqueta “Energía en ejecución” y un doublebox en la cuarta columna para el valor de esta energía.

La cuarta fila y primera columna está conformada por una etiqueta “Tiempo de ejecución”. La segunda columna tiene un hbox con un doublebox para el valor del tiempo y una etiqueta “seg” para la unidad de medida. La tercera columna representa una etiqueta “Energía en ejecución” y un doublebox en la cuarta columna para el valor de esta energía.

La cuarta fila y primera columna contiene una etiqueta “Tiempo de espera”. La segunda columna tiene un hbox con un doublebox para el valor del tiempo y una etiqueta “min” para la unidad de medida. La tercera columna representa una etiqueta “Energía en espera” y un doublebox en la cuarta columna para el valor de la misma.

La quinta fila y primera columna tiene una etiqueta “Operando 1” que se utilizará en en el futuro como una nueva propiedad de los experimentos. La segunda columna tiene un textbox para darle un posible valor. La tercera columna representa una etiqueta “Operando 2” y un textbox en la cuarta columna, con la misma proyección del primer operando.

La sexta fila y primera columna está compuesta por una etiqueta “Descripción”. Y el luego contiene un textbox para la descripción del experimento.

Cerrando el anterior grid, tenemos un separador y le sigue un hbox con dos botones etiquetados como “Guardar”, “Cancelar” y sus atributos de eventos correspondientes.

La interfaz gráfica de la ventana de configuración de experimentos se muestra con la siguiente apariencia:

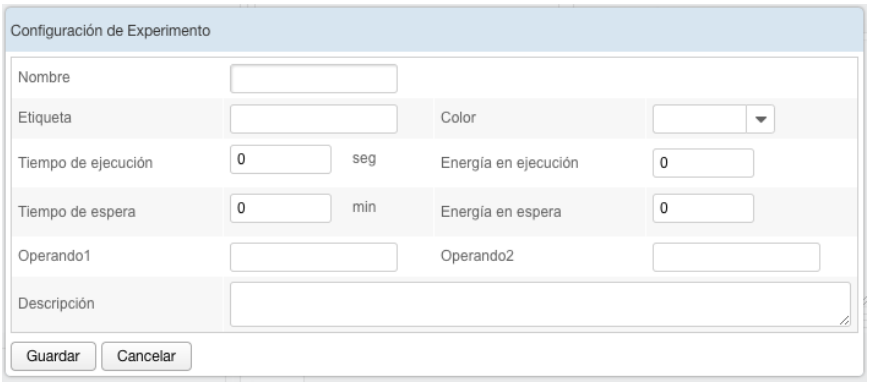


Figura 33. Interfaz gráfica configuración de experimento

Enfoque MVVM

El patrón de diseño **Modelo-Vista-VistaModelo** utilizado en esta parte del desarrollo del software dio lugar a organizar los archivos y sus clases de la siguiente manera:

- **Modelo** está conformado fundamentalmente por los archivos Experimento, Experimentos, GraficoBarra, GraficoPastel, ListRender y Usuario, todos con extensión .java. Estos contienen las clases, datos, getters y setters para los experimentos en acuicultura.
- **Vista** se compone del archivo inicio.zul como la capa de interfaz gráfica. Los componentes con los que interactúa el usuario originan eventos conectados por anotaciones a la VistaModelo.
- **VistaModelo** es el archivo AcuiculturaVM.java, el cual tiene implementada la lógica de negocio y métodos para exponer y procesar la información del Modelo a la Vista inicio.zul.

La capa de AcuiculturaVM.java es una abstracción de la inicio.zul, porque en el diseño de esta capa, parte del análisis de las funcionalidades de la interfaz de usuario (**Vista**), identificando el estado de los componentes y el comportamiento que esperamos de los mismos como se muestra a continuación.

Estado:

- Son todas las variables creadas en el archivo AcuiculturaVM.java

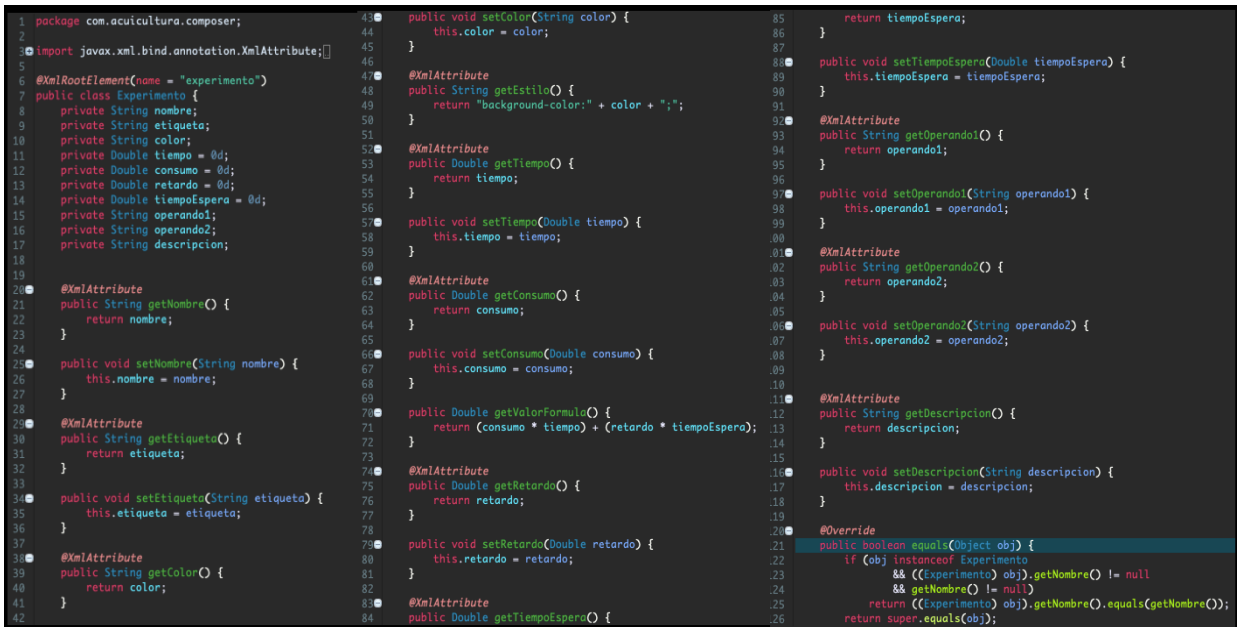
Comportamiento:

Son todos los métodos creados en el archivo AcuiculturaVM.java para:

- Exportar
- Importar
- Salir de sesión
- Seleccionar idioma
- Arrastrar y soltar experimentos
- Graficar experimentos
- Eliminar experimento arrastrando y soltando
- Cambiar escala de porcentaje de la batería
- Ver configuración de experimento
- Ver descripción de experimento
- Editar descripción
- Eliminar experimento por botón X
- Configurar nuevo experimento (solo usuario Administrador Acuicultura)
- Otros

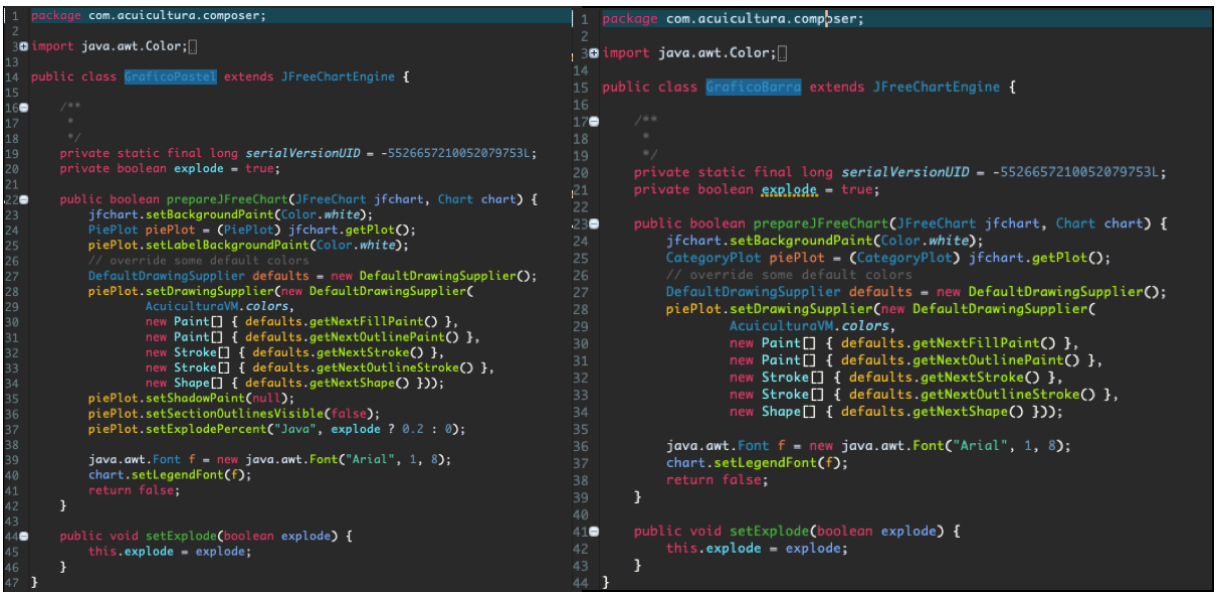
VistaModelo es también una abstracción del **Modelo**, tiene 15 variables para los estados y 17 métodos para el comportamiento. La creación de esta capa se elabora como POJO, a través de métodos setters y getters.

A continuación, se muestran los archivos creados para el **Modelo**.



```java
package com.acuicultura.composer;

import javax.xml.bind.annotation.XmlAttribute;

@XmlRootElement(name = "experimento")
public class Experimento {
    private String nombre;
    private String etiqueta;
    private String color;
    private Double tiempo = 0d;
    private Double consumo = 0d;
    private Double retardo = 0d;
    private Double tiempoEspera = 0d;
    private String operando1;
    private String operando2;
    private String descripcion;


    @XmlAttribute
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    @XmlAttribute
    public String getEtiqueta() {
        return etiqueta;
    }

    public void setEtiqueta(String etiqueta) {
        this.etiqueta = etiqueta;
    }

    @XmlAttribute
    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    @XmlAttribute
    public String getEstilo() {
        return "background-color:" + color + ";";
    }

    @XmlAttribute
    public Double getTiempo() {
        return tiempo;
    }

    public void setTiempo(Double tiempo) {
        this.tiempo = tiempo;
    }

    @XmlAttribute
    public Double getConsumo() {
        return consumo;
    }

    public void setConsumo(Double consumo) {
        this.consumo = consumo;
    }

    public Double getValorFormula() {
        return (consumo * tiempo) + (retardo * tiempoEspera);
    }

    @XmlAttribute
    public Double getRetardo() {
        return retardo;
    }

    public void setRetardo(Double retardo) {
        this.retardo = retardo;
    }

    @XmlAttribute
    public Double getTiempoEspera() {
        return tiempoEspera;
    }

    public void setTiempoEspera(Double tiempoEspera) {
        this.tiempoEspera = tiempoEspera;
    }

    @XmlAttribute
    public String getOperando1() {
        return operando1;
    }

    public void setOperando1(String operando1) {
        this.operando1 = operando1;
    }

    @XmlAttribute
    public String getOperando2() {
        return operando2;
    }

    public void setOperando2(String operando2) {
        this.operando2 = operando2;
    }


    @XmlAttribute
    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj instanceof Experimento
                && ((Experimento) obj).getNombre() != null
                && getNombre() != null)
            return ((Experimento) obj).getNombre().equals(getNombre());
        return super.equals(obj);
```

Figura 34. Archivo Experimento.java



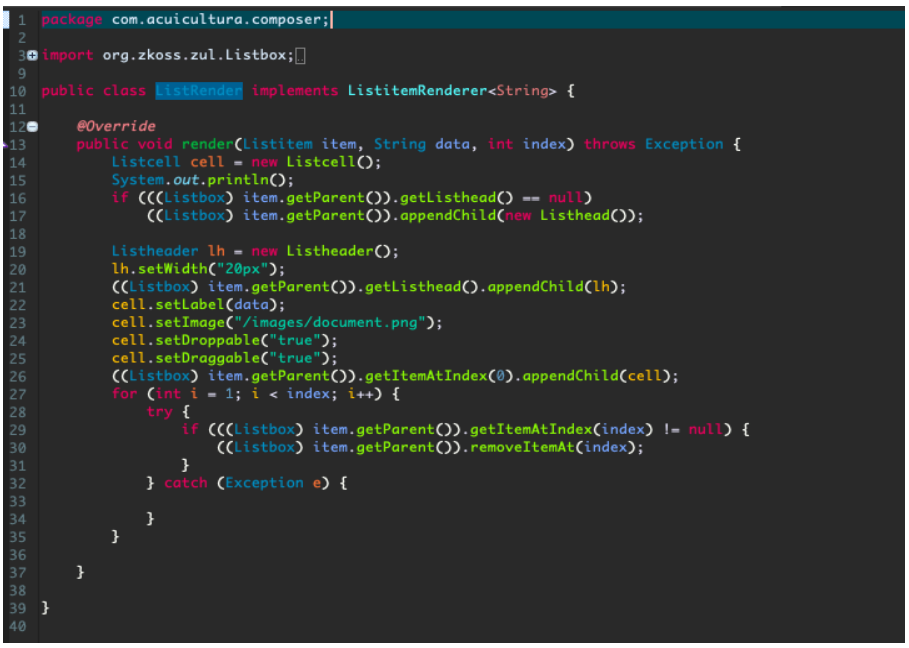
```java
package com.acuicultura.composer;

import java.awt.Color;

public class GraficoPastel extends JFreeChartEngine {

    /**
     *
     */
    private static final long serialVersionUID = -5526657210052079753L;
    private boolean explode = true;

    public boolean prepareJFreeChart(JFreeChart jfchart, Chart chart) {
        jfchart.setBackgroundPaint(Color.white);
        PiePlot piePlot = (PiePlot) jfchart.getPlot();
        piePlot.setLabelBackgroundPaint(Color.white);
        // override some default colors
        DefaultDrawingSupplier defaults = new DefaultDrawingSupplier();
        piePlot.setDrawingSupplier(new DefaultDrawingSupplier(
                AcuiculturaVM.colors,
                new Paint[] { defaults.getNextFillPaint() },
                new Paint[] { defaults.getNextOutlinePaint() },
                new Stroke[] { defaults.getNextStroke() },
                new Stroke[] { defaults.getNextOutlineStroke() },
                new Shape[] { defaults.getNextShape() }));
        piePlot.setShadowPaint(null);
        piePlot.setSectionOutlinesVisible(false);
        piePlot.setExplodePercent("Java", explode ? 0.2 : 0);

        java.awt.Font f = new java.awt.Font("Arial", 1, 8);
        chart.setLegendFont(f);
        return false;
    }

    public void setExplode(boolean explode) {
        this.explode = explode;
    }
}
```

```java
package com.acuicultura.composer;

import java.awt.Color;

public class GraficoBarra extends JFreeChartEngine {

    /**
     *
     */
    private static final long serialVersionUID = -5526657210052079753L;
    private boolean explode = true;

    public boolean prepareJFreeChart(JFreeChart jfchart, Chart chart) {
        jfchart.setBackgroundPaint(Color.white);
        CategoryPlot piePlot = (CategoryPlot) jfchart.getPlot();
        // override some default colors
        DefaultDrawingSupplier defaults = new DefaultDrawingSupplier();
        piePlot.setDrawingSupplier(new DefaultDrawingSupplier(
                AcuiculturaVM.colors,
                new Paint[] { defaults.getNextFillPaint() },
                new Paint[] { defaults.getNextOutlinePaint() },
                new Stroke[] { defaults.getNextStroke() },
                new Stroke[] { defaults.getNextOutlineStroke() },
                new Shape[] { defaults.getNextShape() }));

        java.awt.Font f = new java.awt.Font("Arial", 1, 8);
        chart.setLegendFont(f);
        return false;
    }

    public void setExplode(boolean explode) {
        this.explode = explode;
    }
}
```

Figura 35. Archivos GraficoPastel.java (izquierda) y GraficoBarras.java (derecha).

```java
package com.acuicultura.composer;

import java.util.ArrayList;

@XmlRootElement(name = "experimentos")
public class Experimentos {

    private List<Experimento> experimentos = new ArrayList<Experimento>();

    public Experimentos() {
    }

    @XmlElement(name = "experimento")
    public List<Experimento> getExperimentos() {
        return experimentos;
    }

    public void setExperimentos(List<Experimento> experimentos1) {
        this.experimentos = experimentos1;
    }
}
```

Figura 36. Archivo Experimentos.java



```java
package com.acuicultura.composer;

import org.zkoss.zul.Listbox;

public class ListRender implements ListitemRenderer<String> {

    @Override
    public void render(Listitem item, String data, int index) throws Exception {
        Listcell cell = new Listcell();
        System.out.println();
        if (((Listbox) item.getParent()).getListhead() == null)
            ((Listbox) item.getParent()).appendChild(new Listhead());

        Listheader lh = new Listheader();
        lh.setWidth("20px");
        ((Listbox) item.getParent()).getListhead().appendChild(lh);
        cell.setLabel(data);
        cell.setImage("/images/document.png");
        cell.setDroppable("true");
        cell.setDraggable("true");
        ((Listbox) item.getParent()).getItemAtIndex(0).appendChild(cell);
        for (int i = 1; i < index; i++) {
            try {
                if (((Listbox) item.getParent()).getItemAtIndex(index) != null) {
                    ((Listbox) item.getParent()).removeItemAt(index);
                }
            } catch (Exception e) {

            }
        }

    }

}
```

Figura 37. Archivo ListRender.java

Anotaciones y librerías

A continuación, se exponen las anotaciones utilizadas en los archivos para la interfaz gráfica de usuario en los experimentos de acuicultura. Para una mejor comprensión de las anotaciones se le asignan las viñetas (▪) a las de la Vista y (➢) a las de la VistaModelo.

Archivo.zul:

- "@id('vm') @init('com.acuicultura.composer.AcuiculturaVM')"
- "@command('guardarExp')"
- "@command('importar')"
- "@command('salir')"
- "@command('esp')"
- "@command('eng')"

- "@command('drop')"
- "@load(vm.usuario.admin)"
- "@command('agregar')"
- "@load(vm.tiposExperimentos)"
- "@command('seleccionar')"
- "@bind(vm.experimento)"
- "@command('detalleExperimento', experimento=each)"
- "@load(each.estilo)"
- "@command('onDrop', item=each)"
- "@command('eliminarTipo', tipo=each)"
- "@load(vm.experimento.etiqueta)"
- "@load(vm.experimento.descripcion)"
- "@load(vm.porciento)"
- "@bind(vm.pila)"
- "@command('actualizarPila')"
- "@bind(vm.pieModel)"
- "@bind(vm.graficoBarra)"
- "@bind(vm.experimento.nombre)"
- "@load(not (empty vm.experimento.nombre) or not vm.usuario.admin)"
- "@bind(vm.experimento.etiqueta)"
- "@load(not vm.usuario.admin)"
- "@load(not vm.usuario.admin)"
- "@bind(vm.experimento.color)"
- "@load(not vm.usuario.admin)"
- "@bind(vm.experimento.tiempo)"
- "@bind(vm.experimento.consumo)"
- "@load(not vm.usuario.admin)"
- "@bind(vm.experimento.tiempoEspera)"
- "@load(not vm.usuario.admin)"
- "@bind(vm.experimento.retardo)"
- "@load(not vm.usuario.admin)"
- "@bind(vm.experimento.operando1)"
- "@load(not vm.usuario.admin)"
- "@bind(vm.experimento.operando2)"
- "@load(not vm.usuario.admin)"
- "@bind(vm.experimento.descripcion)"
- "@command('guardar')"
- "@load(not vm.usuario.admin)"
- "@command('cancelar')"

Archivos.java:

- @Wire
- @AfterCompose
- @NotifyChange("experimento")
    - @Command
    - public void seleccionar() {
- @Command
    - public void onDrop
- @Command
    - public void drop
- @Command
    - public void esp

- @Command
  - public void eng()
- @Command
  - public void salir()
- @Command
  - public void eliminarTipo (@BindingParam
- @NotifyChange("experimento")
  - @Command
  - public void agregar()
- @NotifyChange("experimento")
  - @Command
  - public void detalleExperimento(@BindingParam("experimento")
- @Command
  - public void cancelar()
- @Command
  - public void importar(@ContextParam
- @Command
  - public void guardarExp()
- @Command
  - public void guardar()
- @Command
  - public void actualizarPila()
- @Listen("onDrop=#panel")
- @Listen("onDrop=#eliminar")
- @Listen("onDrop=#lbTiposExperimentos")
- @XmlRootElement(name = "experimento")
- @XmlAttribute
- @XmlElement(name = "experimento")

Librerías:

1. java.awt.Color:
   - import java.io.ByteArrayInputStream;
   - import java.io.ByteArrayOutputStream;
   - import java.io.InputStream;
   - import java.io.InputStreamReader;
   - import java.io.Reader;
   - import java.util.HashMap;
   - import java.util.Map;
   - import javax.xml.bind.JAXBContext;
   - import javax.xml.bind.Unmarshaller;
   - import org.zkoss.bind.BindContext;
   - import org.zkoss.bind.BindUtils;
   - import org.zkoss.bind.annotation.AfterCompose;
   - import org.zkoss.bind.annotation.BindingParam;
   - import org.zkoss.bind.annotation.Command;
   - import org.zkoss.bind.annotation.ContextParam;
   - import org.zkoss.bind.annotation.ContextType;
   - import org.zkoss.bind.annotation.NotifyChange;
   - import org.zkoss.util.media.Media;
   - import org.zkoss.zhtml.Filedownload;
   - import org.zkoss.zhtml.Messagebox;
   - import org.zkoss.zk.ui.Component;

- import org.zkoss.zk.ui.Executions;
- import org.zkoss.zk.ui.event.DropEvent;
- import org.zkoss.zk.ui.event.UploadEvent;
- import org.zkoss.zk.ui.select.Selectors;
- import org.zkoss.zk.ui.select.annotation.Listen;
- import org.zkoss.zk.ui.select.annotation.Wire;
- import org.zkoss.zul.CategoryModel;
- import org.zkoss.zul.Div;
- import org.zkoss.zul.Hbox;
- import org.zkoss.zul.Label;
- import org.zkoss.zul.ListModelList;
- import org.zkoss.zul.Listcell;
- import org.zkoss.zul.PieModel;
- import org.zkoss.zul.Progressmeter;
- import org.zkoss.zul.SimpleCategoryModel;
- import org.zkoss.zul.SimplePieModel;
- import org.zkoss.zul.Vbox;
- import org.zkoss.zul.Window;
- import java.awt.Paint;
- import java.awt.Shape;
- import java.awt.Stroke;
- import org.jfree.chart.JFreeChart;
- import org.jfree.chart.plot.CategoryPlot;
- import org.jfree.chart.plot.DefaultDrawingSupplier;
- import org.jfree.chart.plot.PiePlot;
- import org.zkoss.zkex.zul.impl.JFreeChartEngine;
- import org.zkoss.zul.Chart;
- import org.jfree.chart.plot.CategoryPlot;

2. javax.xml.bind.annotation.XmlAttribute:
   - import javax.xml.bind.annotation.XmlRootElement;
3. java.util.ArrayList:
   - import java.util.List;
   - import javax.xml.bind.annotation.XmlElement;
   - import javax.xml.bind.annotation.XmlRootElement;
4. org.zkoss.zul.Listbox:
   - import org.zkoss.zul.Listcell;
   - import org.zkoss.zul.Listhead;
   - import org.zkoss.zul.Listheader;
   - import org.zkoss.zul.Listitem;
   - import org.zkoss.zul.ListitemRenderer;

Enlaces y funcionamientos de los componentes gráficos

A continuación, se describen las anotaciones de enlaces comprendidas en la Vista inicio.zul y la VistaModelo AcuiculturaVM.java para el funcionamiento de la interfaz gráfica de usuario principal.

- **<groupbox**

  "@id('vm') @init('com.acuicultura.composer.AcuiculturaVM')"

Similar a la primera parte, aplicamos el compositor "**org.zkoss.bind.BindComposer**" en la Vista inicio.zul para procesar las expresiones de "data binding" e inicializar las clases de VistaModelo AcuiculturaVM.java. Mediante el id se hará referencia a las propiedades de la clase AcuiculturaVM (ejemplo vm.experimento) con las expresiones "data binding".

- **<button = "Exportar"**

  "@command('guardarExp')"

Enlazamos el evento "onClick" del componente (botón) de la vista con su respectivo comando (método) en la VistaModelo AcuiculturaVM.java, para que ZK encuentre e invoque el método cuando este evento sea lanzado desde la Vista inicio.zul. Se ha marcado el método "guardarExp()" como comando, que es el mismo nombre del método como se ilustra a continuación.

```
94    @Command
95    public void guardarExp() throws Exception {
96        Experimentos experimentos = new Experimentos();
97        experimentos.getExperimentos().addAll(tiposExperimentos);
98
99        JAXBContext jaxbContext = JAXBContext.newInstance(Experimentos.class);
00        javax.xml.bind.Marshaller jaxbMarshaller = jaxbContext
01                .createMarshaller();
02        jaxbMarshaller.setProperty(javax.xml.bind.Marshaller.JAXB_ENCODING,
03                "utf-8");
04        jaxbMarshaller.setProperty(
05                javax.xml.bind.Marshaller.JAXB_FORMATTED_OUTPUT, true);
06        ByteArrayOutputStream os = new ByteArrayOutputStream();
07        jaxbMarshaller.marshal(experimentos, os);
08        byte[] fileByte = os.toByteArray();
09        os.close();
10        Filedownload.save(fileByte, "APPLICATION_XML", "Experimentos.xml");
11    }
```

Figura 38. Método para exportar experimentos

Este método nos permite guardar en formato con extensión .xml los experimentos en el almacenamiento interno del ordenador.

- **<button = "Importar"**

  "@command('importar')"

De igual manera para importar un archivo .xml se creó el método "importar" con su anotación de comando en el VistaModelo enlazado a la Vista con el evento "onUpload". Este método se define como muestra la siguiente imagen.

```
77    @Command
78    public void importar(@ContextParam(ContextType.BIND_CONTEXT) BindContext ctx)
79            throws Exception {
80        UploadEvent upEvent = (UploadEvent) ctx.getTriggerEvent();
81        Media archivo = upEvent.getMedia();
82
83        JAXBContext jaxbContext = JAXBContext.newInstance(Experimentos.class);
84        InputStream is = new ByteArrayInputStream(archivo.getStringData()
85                .getBytes());
86        Reader reader = new InputStreamReader(is);
87        Unmarshaller jaxbUnMarshaller = jaxbContext.createUnmarshaller();
88        Experimentos experimentos = (Experimentos) jaxbUnMarshaller
89                .unmarshal(reader);
90        tiposExperimentos.clear();
91        tiposExperimentos.addAll(experimentos.getExperimentos());
92    }
```

Figura 39. Método para importar experimentos

- **<button = "Salir"**

  "@command('salir')"

Para salir de la sesión tenemos el botón “Salir” que contiene el evento “onClick” en la Vista, enlazado al comando de VistaModelo con el método “salir”. Es el método mostrado a continuación donde nos redirige al inicio de sesión de la Vista login.zul

```
122    @Command
123    public void salir() {
124        Executions.getCurrent().getSession().setAttribute("usuario", null);
125        Executions.sendRedirect("/zul/login.zul");
```

Figura 40. Método para salir de la sesión

- **<vbox: id="panel”**

El vbox con atributo droppable se identifica como “panel” en la Vista para la instanciarse en el ModeloVista. El Vbox “panel” está cableado desde la Vista con VistaModelo por la anotación @Wire que posteriormente explicaremos.

"@command('drop')"

El evento onDrop del “panel” (Vista) tiene como comando el método “drop” (VistaModelo) que se observa en la siguiente figura.

```
97     @Command
98     public void drop() {
99         System.out.println();
100    }
```

Figura 41. Método “drop” de experimentos hacia el panel

➢ **@Listen("onDrop=#panel")**

```
257    @Listen("onDrop=#panel")
258    public void drop(DropEvent event) {
259        if (event.getDragged().getFirstChild() instanceof Listcell) {
260            String s = ((Listcell) event.getDragged().getFirstChild())
261                    .getLabel();
262            Experimento e = obtenerExperimento(s);
263            if (!pilaLlena(e)) {
264                if (panel.getChildren().get(index).getChildren().size() == 20) {
265                    index++;
266                    panel.getChildren().add(new Hbox());
267                }
268                Label l = new Label(s);
269                l.setStyle("color: white;");
270                Div d = new Div();
271                d.setWidth("30px");
272                d.setStyle(e.getEstilo());
273                d.setDroppable("true");
274                d.setDraggable("true");
275                d.appendChild(l);
276                Div d1 = new Div();
277                d1.setStyle("background-color: black;");
278                d1.setWidth("10px");
279                d1.setTooltip("Retardo: " + e.getRetardo());
280                Label l1 = new Label("_");
281                d1.appendChild(l1);
282                panel.getChildren().get(index).getChildren().add(d);
283                panel.getChildren().get(index).getChildren().add(d1);
284                experimentos.add(e);
285            } else
286                Messagebox.show("Bateria agotada", "Error", Messagebox.OK,
287                        Messagebox.ERROR, null);
288        }
289
290        actualizarPila();
291
292    }
```

Figura 42. Método para “Drag & Drop” en el panel de programación de experimentos

La anotación @listen se emplea para declarar un método de escucha de eventos en VistaModelo. El método detecta o escucha si el usuario arrastra un experimento hacia el panel. Para ello previamente se debe denotar el Selectors.wireEventListeners() en el método @AfterCompose, que posteriormente veremos.

- **<button ="Nuevo Experimento"**

  "@load(vm.usuario.admin)"

Este botón en la Vista tiene el atributo de ser "visible" solo para el usuario Administrador Acuicultura y para ello se enlaza mediante la anotación hacia el getter de "usuario" en la VistaModelo y luego al getter "is admin" de la clase Usuario del modelo Usuario.java.

"@command('agregar')"

Además, el botón tiene el atributo de evento "onClick" (Vista) enlazado con el comando del método "agregar" (VistaModelo) como se muestra en la siguiente figura.

```
@NotifyChange("experimento")
@Command
public void agregar() {
    ventanaAgregar.doModal();
    ventanaAgregar.setVisible(true);
    experimento = new Experimento();
}
```

Figura 43. Método para crear nuevo experimento

- **@NotifyChange("mensaje")**

Esta anotación notificara al "binder" de los cambios en las propiedades de VistaModelo

- **<listbox: id="lbTiposExperimentos"**

  "@load(vm.tiposExperimentos)"

La caja de listas en la Vista tiene como id "lbTiposExperimentos" y atributo "model" para cargar la información del getter tiposExperimentos como un ListModelList de tipo experimento.

- **@Listen("onDrop=#lbTiposExperimentos")**

Se utiliza declarando el método de escucha de eventos en VistaModelo para detectar si el usuario realiza un "drop" de un tipo de experimento de la lista.

```
@Listen("onDrop=#lbTiposExperimentos")
public void dropTipos(DropEvent event) {
    System.out.println(event.getDragged());
}
```

Figura 44. Método para "drop" de tipos de experimentos

"@command('seleccionar')"

Listbox tiene además el atributo de evento “onSelect” (Vista) enlazado con el comando del método “seleccionar” (VistaModelo), mostrándose en la siguiente ilustración.

```
86    @NotifyChange("experimento")
87    @Command
88    public void seleccionar() {
89
90    }
```

Figura 45. Método para seleccionar experimento

"@bind(vm.experimento)"

También listbox tiene el atributo de “selectedItem” utilizando la anotación @bind para vincular datos con parámetros, cargando y guardando en el getter de VistaModelo.

"@command('detalleExperimento', experimento=each)"

Dentro del listbox tenemos un listitem con atributos “draggable y droppable” y un atributo de evento “onDoubleClick” enlazado con el comando del método “detalleExperimento”, especificándose para cada uno de estos.

```
41    @NotifyChange("experimento")
42    @Command
43    public void detalleExperimento(@BindingParam("experimento") Experimento experimento) {
44        this.experimento = experimento;
45        ventanaAgregar.doModal();
46        ventanaAgregar.setVisible(true);
47    }
```

Figura 46. Método para visualizar detalle de experimento

La anterior figura describe el método si realizamos doble clic sobre un experimento en la interfaz gráfica. Se observa además el parámetro de método de comando **@bindingParam**, este tiene como propósito notificarle al binder que recupere el parámetro con una clave específica del argumento de enlace en la Vista.

"@load(each.estilo)"

El atributo “style” se enlaza con la VistaModelo para cargar los parámetros y datos de destino en el getter “getEstilo” para cada experimento.

"@command('onDrop', ítem=each)"

El evento “onDrop” de listitem tiene como comando el método “onDrop” que se observa en la siguiente figura.

```
92    @Command
93    public void onDrop(@BindingParam("item") String item) {
94        System.out.println(item);
95    }
```

Figura 47. Método “onDrop” de lista de experimentos

"@load(each.etiqueta)"

Dentro del listitem se tiene un listcell que como atributo de etiqueta va a cargar para cada experimento, su etiqueta correspondiente desde el getter "getEtiqueta" desde el modelo de Experimento.java.

"@command('eliminarTipo', tipo=each)"

Para cada experimento tendrá asociado de forma continua una listcell con etiqueta "X" y el atributo de evento "onClick", enlazado al comando de método "eliminarTipo" para eliminar dicho experimento. Este método se muestra a continuación.

```
128    @Command
129    public void eliminarTipo(@BindingParam("tipo") Experimento tipo) {
130        tiposExperimentos.remove(tipo);
131    }
```

Figura 48. Método para eliminar experimentos de tipo lista

- **<groupbox**

"@load(vm.experimento.etiqueta)"

"@load(vm.experimento.descripción)"

El groupbox tiene acoplado un caption y un texbox para cargar la información respectivamente contenida en los getters "getEtiqueta" y "getDescripcion" en la clase Experimento del Modelo Experimento.java.

- **<groupbox: id="eliminar"**

  - **@Listen("onDrop=#eliminar")**

El groupbox con id "eliminar" tiene asociado en VistaModelo la anotación @listen para declarar el método de escucha de eventos "eliminar". El método detecta si el usuario arrastra un experimento hasta el groupbox con el fin de eliminarlo.

```
306    @Listen("onDrop=#eliminar")
307    public void eliminar(DropEvent event) {
308        if (event.getDragged().getFirstChild() instanceof Label) {
309            event.getDragged().getNextSibling().detach();
310            event.getDragged().detach();
311            experimentos.clear();
312            for (int i = 0; i <= index; i++) {
313                for (Component c : panel.getChildren().get(i).getChildren()) {
314                    Label l = (Label) c.getFirstChild();
315                    if (l != null && !(l.getValue().equals("_"))) {
316                        Experimento e = obtenerExperimento(l.getValue());
317                        experimentos.add(e);
318                    }
319                }
320            }
321        }
322        actualizarPila();
323    }
```

Figura 49. Método para eliminar experimentos con "Drag & Drop" hacia el depósito de Eliminación.

- **<hbox: id="curr_met"**

El progressmeter con id “curr_met” que está dentro del hbox con etiqueta “Batería”, cableado desde la Vista con VistaModelo por la anotación @Wire que luego analizaremos.

"@load(vm.porciento)"

El hbox tiene como atributo una etiqueta para mostrar el porciento de batería definida por la anotación @load que carga los datos del getter “getPorciento” de la VistaModelo.

"@bind(vm.pila)"

Luego se representa un doublebox enlazado por @bind desde la Vista con VistaModelo para cargar y guardar datos en el getter “getPila”.

"@command('actualizarPila')"

El doublebox posee el atributo de evento “onChange” que permite cambiar mediante su comando de método “actualizarPila” la escala de porcentaje de la batería. Se representa en la siguiente figura.

```
215    @Command
216    public void actualizarPila() {
217        double total = 0;
218        colors = new Color[experimentos.size()];
219        for (Experimento exp : experimentos) {
220            total += exp.getValorFormula();
221        }
222        porciento = total * 100 / pila;
223        if (!(porciento > 100)) {
224            curr_met.setValue(porciento.intValue());
225            curr_met.setTooltiptext(porciento.toString());
226            actualizarGrafico();
227        } else
228            Messagebox.show("Batería agotada", "Error", Messagebox.OK,
229                    Messagebox.ERROR, null);
230        BindUtils.postNotifyChange(null, null, this, "porciento");
231    }
```

Figura 50. Método para actualizar el porcentaje de la batería

En la figura anterior se puede observar el getter “getValorFormula” de tipo Double, el cual está definido en el archivo Experimento.java para formular el consumo energético de la batería, como la sumatoria de los productos del consumo por tiempo y retardo por tiempo de espera (**consumo** * **tiempo**) + (**retardo** * **tiempoEspera**). Entiéndase por **consumo** la energía en ejecución del experimento, **tiempo** es el tiempo de ejecución. Por otra parte, **retardo** es la energía en espera y **tiempoEspera** es el tiempo de espera. La fórmula se ha establecido para la actualización de la batería y los gráficos que veremos a continuación.

- **<chart**

"@bind(vm.pieModel)"

El chart para representar la gráfica de pastel en la interfaz de usuario tiene como atributo de modelo (model) una anotación @bind para cargar y guardar estos en el getter “getPieModel” de tipo PieModel en la VistaModelo. La siguiente figura nos describe el método utilizado para actualizar este gráfico.

"@bind(vm.graficoPastel)"

Tambien tiene como atributo en la Vista el motor (engine) enlazado con la anotación @bind para cargar y guardar estos en el getter “getGraficoPastel” del tipo GraficoPastel en la VistaModelo. La siguiente figura nos describe el método utilizado para actualizar este gráfico.

```
private void actualizarGrafico() {
    this.pieModel = new SimplePieModel();
    this.barModel = new SimpleCategoryModel();

    Map<Experimento, Double> graf = new HashMap<Experimento, Double>();
    for (Experimento exp : experimentos) {
        if (graf.containsKey(exp))
            graf.put(exp, graf.get(exp) + exp.getValorFormula());
        else
            graf.put(exp, exp.getValorFormula());
    }
    int index = 0;
    for (Experimento exp : graf.keySet()) {
        pieModel.setValue(exp.getEtiqueta(), graf.get(exp));
        barModel.setValue(exp.getEtiqueta(), exp.getEtiqueta(),
                graf.get(exp));
        colors[index] = hex2Rgb(exp.getColor());
        index++;
    }

    BindUtils.postNotifyChange(null, null, this, "pieModel");
    BindUtils.postNotifyChange(null, null, this, "barModel");
}
```

Figura 51. Método para actualizar los gráficos de pastel y barra.

- **<chart**

"@load(vm.barModel)"

De igual forma se presenta el chart de barras en la Vista con el atributo de modelo (model) cargargando los datos desde el getter “getBarModel” de tipo CategoryModel en la VistaModelo. Se puede observar en la anterior figura el método de actualizar el gráfico de barras.

"@bind(vm.graficoBarra)"

Como atributo motor (engine) se enlazada con la anotación @bind para cargar y guardar en el getter “getGraficoBarra” del tipo GraficoBarra en la VistaModelo del archivo AcuiculturaVM.java.

- **<window: id="ventanaAgregar"**

Windows es la ventana de configuración de experimentos con id “ventanaAgregar” que está cableada desde la Vista con VistaModelo por la anotación @Wire que posteriormente se explicara.

- **<textbox**

"@bind(vm.experimento.nombre)"

Se representa en la ventana un textbox cargando y guardando información, utilizando la anotación @bind para acceder a los getter “getExperimento” de tipo Experimento en

VistaModelo y el getter “getNombre” de tipo String en el Modelo del archivo Experimento.java.

"@load(not (empty vm.experimento.nombre) or not vm.usuario.admin)"

Esta anotación se define para establecer al usuario Administrador Acuicultura como el único que puede crear o modificar el parámetro nombre. Esta anotación se emplea para todos los componentes gráficos de la ventana que siguen a continuación exceptuando el de la descripción de experimentos.

- **<textbox**

"@bind(vm.experimento.etiqueta)"

La ventana muestra un textbox cargando y guardando los datos, utilizando la anotación @bind para acceder a los getter “getExperimento” de tipo Experimento en VistaModelo y el getter “getEtiqueta” de tipo String en el Modelo del archivo Experimento.java.

"@load(not vm.usuario.admin)"

- **<combobox**

"@bind(vm.experimento.color)"

En la representación de la ventana se observa un combobox desplegable, cargando y guardando información con la anotación @bind para acceder a los getter “getExperimento” de tipo Experimento en VistaModelo y el getter “getColor” de tipo String en el Modelo del archivo Experimento.java.

"@load(not vm.usuario.admin)"

- **<doublebox**

"@bind(vm.experimento.tiempo)"

Se muestra en la ventana un doublebox cargando y guardando los datos, utilizando la anotación @bind para acceder a los getter “getExperimento” de tipo Experimento en VistaModelo y el getter “getTiempo” de tipo Double en el Modelo del archivo Experimento.java.

"@load(not vm.usuario.admin)"

- **<doublebox**

"@bind(vm.experimento.consumo)"

La ventana muestra un doubletbox cargando y guardando información, empleando la anotación @bind para acceder a los getter “getExperimento” de tipo Experimento en

VistaModelo y el getter “getConsumo” de tipo Double en el Modelo del archivo Experimento.java.

"@load(not vm.usuario.admin)"

- **<doublebox**

"@bind(vm.experimento.tiempoEspera)"

De igual manera la ventana muestra un doubletbox cargando y guardando información, empleando la anotación @bind para acceder a los getter “getExperimento” de tipo Experimento en VistaModelo y el getter “getTiempoEspera” de tipo Double en el Modelo del archivo Experimento.java

"@load(not vm.usuario.admin)"

- **<doublebox**

"@bind(vm.experimento.retardo)"

La ventana presenta un doubletbox cargando y guardando información, empleando la anotación @bind para acceder a los getter “getExperimento” de tipo Experimento en VistaModelo y el getter “getRetardo” de tipo Double en el Modelo del archivo Experimento.java

"@load(not vm.usuario.admin)"

- **<textbox**

"@bind(vm.experimento.descripcion)"

La descripción se observa en la ventana un textbox cargando y guardando información, utilizando la anotación @bind para acceder a los getter “getExperimento” de tipo Experimento en VistaModelo y el getter “getDescripcion” de tipo String en el Modelo del archivo Experimento.java.

- **<button = "Guardar"**

"@command('guardar')"

Enlazamos el evento “onClick” del componente (botón) de la ventana con su respectivo comando (método) en la VistaModelo AcuiculturaVM.java, para que ZK encuentre e invoque el método cuando este evento sea lanzado desde la Vista inicio.zul. Se ha marcado el método “guardar()” como comando, que es el mismo nombre del método como se ilustra a continuación.

"@load(not vm.usuario.admin)"

```
@Command
public void guardar() {
    if (experimento.getNombre() != null && experimento.getColor() != null
            && experimento.getConsumo() != null
            && experimento.getEtiqueta() != null
            && experimento.getTiempo() != null) {
        if (!tiposExperimentos.contains(experimento))
            tiposExperimentos.add(experimento);
        else
            tiposExperimentos.set(tiposExperimentos.indexOf(experimento),
                    experimento);

        ventanaAgregar.doEmbedded();
    }
    ventanaAgregar.setVisible(false);
    this.experimento = null;
    BindUtils.postNotifyChange(null, null, this, "tiposExperimentos");
    BindUtils.postNotifyChange(null, null, this, "experimento");
}
```

Figura 52. Método para guardar configuración de experimento

- **<button = "Cancelar"**

  "@command('cancelar')"

El atributo de evento "onClick" del botón de la ventana se enlaza con el comando de método en la VistaModelo denominado "cancelar()", como se ilustra a continuación.

```
@Command
public void cancelar() {
    ventanaAgregar.doEmbedded();
    ventanaAgregar.setVisible(false);
    this.experimento = null;
    BindUtils.postNotifyChange(null, null, this, "experimento");
}
```

Figura 53. Método para cancelar configuración de experimento

- **@Wire**

Podemos usar esta anotación para cablear variables de objetos implícitos o variables registradas en un ViewModel dentro del compositor. BindComposer transfiere esas variables por nosotros, antes de invocar el método inicial. Pero no serán componentes de cableado y oyentes automáticamente en un compositor. Para esto se deben pasar los componentes como parámetros en el enlace de comandos o invocar Selectors.wireComponents() en un método con @AfterComposer como muestra la figura 54. Esta anotación se utilizó como se ilustra en la siguiente figura en tres variables para cablear la Vista con el VistaModelo. El "panel" (área dinámica de programación de experimentos), "ventanaAgregar" (ventana de configuración de experimentos) y "curr_met" (progressmeter de la batería).

```
@Wire
private Vbox panel;
@Wire
private Window ventanaAgregar;
@Wire
private Progressmeter curr_met;
```

Figura 54. Variables de cableado

@AfterCompose

Luego de establecer la anotación de método de clase, se inicializa definiendo la interfaz gráfica de usuario inicial, a la hora de ejecutar la aplicación web. Como se ilustra en la siguiente figura podemos observar que comenzará siempre con el inicio de sesión de la Vista login.zul.

```
68
69    @AfterCompose
70    public void afterCompose(@ContextParam(ContextType.VIEW) Component view) {
71        Selectors.wireComponents(view, this, false);
72        Selectors.wireEventListeners(view, this);
73        inicializar();
74    }
75
76    private void inicializar() {
77        if (Executions.getCurrent().getSession().getAttribute("usuario") != null)
78            usuario = (Usuario) Executions.getCurrent().getSession()
79                    .getAttribute("usuario");
80        else
81            Executions.sendRedirect("/zul/login.zul");
82        System.out.println(Executions.getCurrent().getSession()
83                .getAttribute(org.zkoss.web.Attributes.PREFERRED_LOCALE));
84    }
85
```

Figura 55. Método para determinar vista inicial

También podemos ver otras anotaciones en los archivos Experimento.java y Experimentos.java para componer el archivo .xml con su ruta, elementos y atributos.

@XmlRootElement(name = "experimento")

@XmlElement(name = "experimento")

@XmlAttribute

CAPÍTULO 6. Resultados y validación

En el presente capítulo se expone como resultado la herramienta de programación y estimación del consumo energético para un Dispositivo Autónomo de Experimentación en Acuicultura, con las historias de usuarios ya implementadas. Se demuestran las funcionalidades del módulo de Administración Acuicultura y Usuario Acuicultura mediante ilustraciones y descripciones que explican todas las utilidades y aportaciones en el sistema.

Además, se realiza la validación de la aplicación software mediante un conjunto de pruebas que garantizan la aceptación de la misma por parte de los usuarios finales en su uso, y se comprueba que el software desarrollado es multiplataforma, desplegándose en los sistemas operativos actuales más utilizados, de forma ágil, eficaz y con escasos recursos necesarios.

6.1. Inicio de sesión

La interfaz inicial del sistema es la de inicio de sesión donde un usuario accede a la aplicación en dependencia del tipo de usuario y su clave correspondiente. La herramienta software es accesible solamente por dos tipos de usuarios, Usuario Acuicultura y Administrador Acuicultura. La clave y tipo de usuario es proporcionada por el desarrollador de la aplicación, y solo él como responsable puede modificarla con la debida autorización y convenio con el cliente.

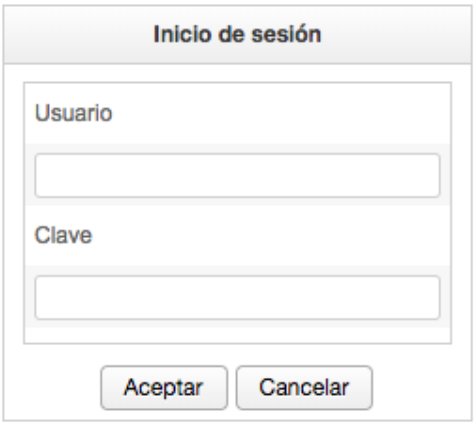


Figura 56. Interfaz gráfica de inicio de sesión

6.2. Usuarios

Una vez logueado el usuario será redireccionado a su módulo correspondiente como Administrador Acuicultura o Usuario Acuicultura, donde podrá utilizar las interfaces y funcionalidades relacionadas con la programación y gestión de los experimentos de

acuicultura en función del consumo energético del (DAE).

6.2.1 Administrador Acuicultura

En la ilustración siguiente se muestra el módulo de interfaz de usuario Administrador Acuicultura una vez introducido su nombre de usuario y contraseña en el inicio de sesión. Esta interfaz gráfica se comporta de manera similar en los diferentes navegadores más utilizados a día de hoy como Google Chrome, Safari, Edge, Mozilla Firefox, Opera, entre otros.

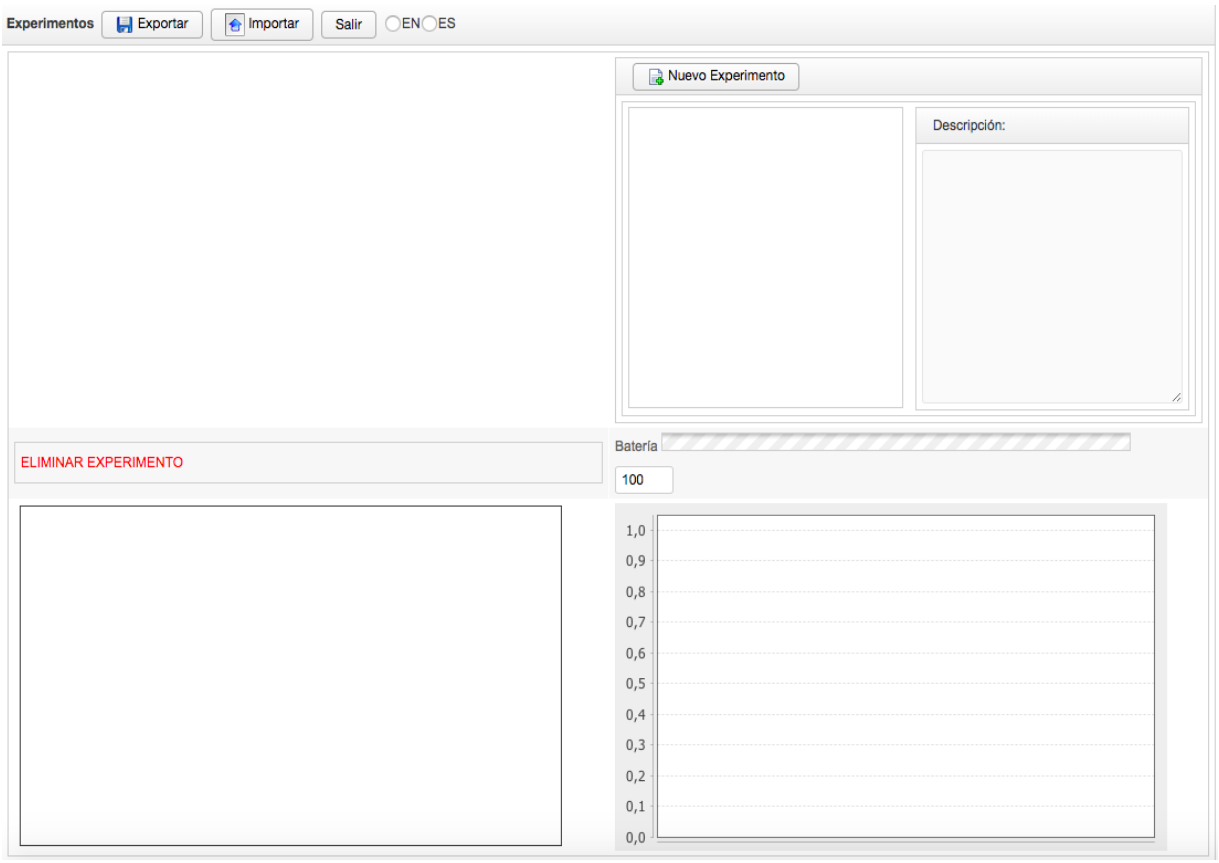


Figura 57. Interfaz gráfica de usuario Administrador Acuicultura

A continuación, se describen cada una de las áreas, barras, herramientas de trabajo, gráficos y funcionalidades, apoyándonos de un proyecto ejemplo desarrollado como objeto de estudio. El proyecto está conformado por un total de 10 experimentos, repitiendo algunos de ellos en la programación y gestión de la autonomía de la batería del dispositivo DAE.

Los experimentos se enfocaron en primitivas y parámetros medidos por el DAE en algunas especies de peces como la lubina, dorada y alevines de las mismas. Auxiliándonos de la siguiente figura se detallan las partes y componentes de este módulo.

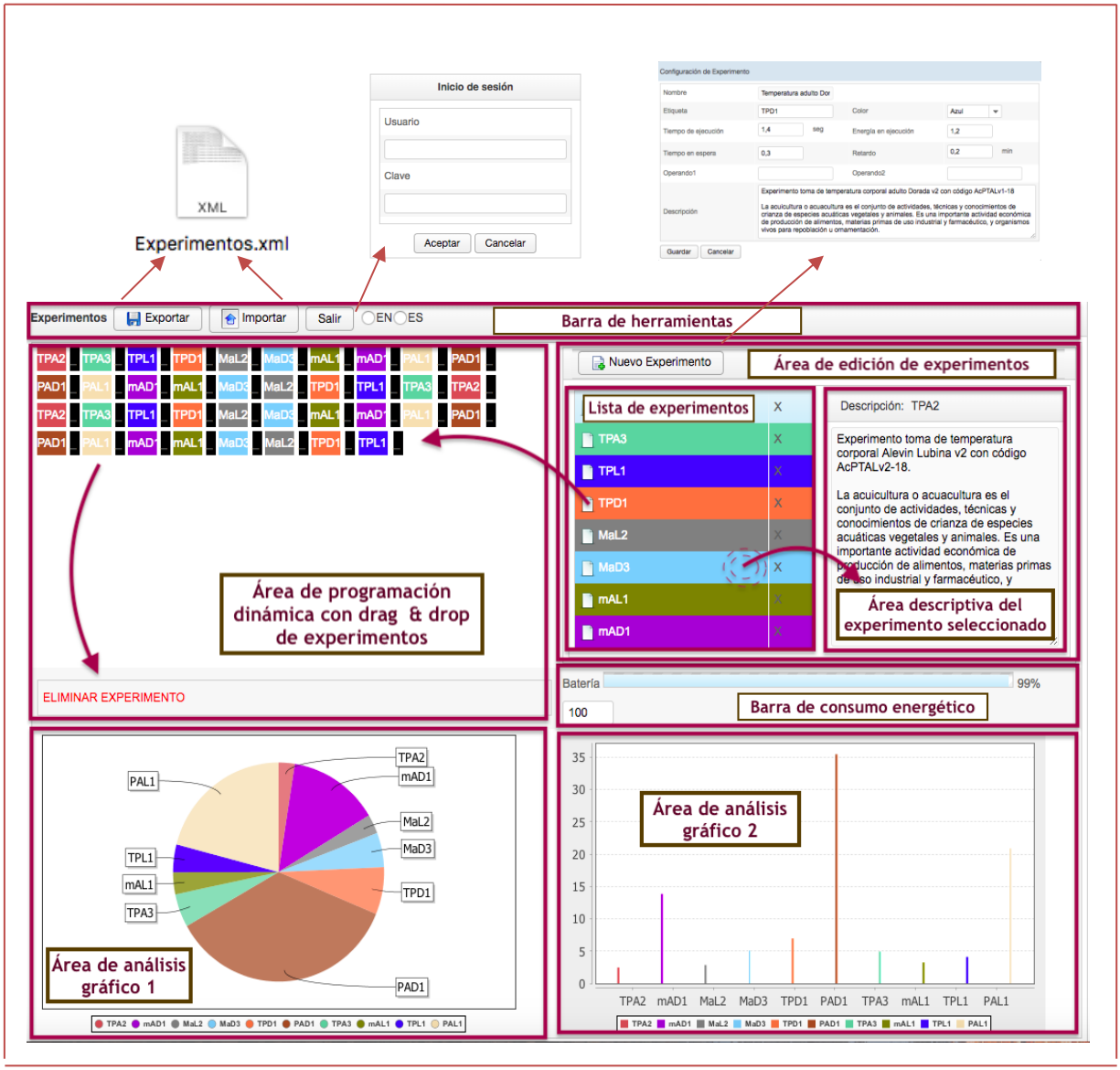


Figura 58. Dinámica de experimentación usuario Administrador Acuicultura

- **Barra de herramientas:** Se muestra en la parte superior de la aplicación conteniendo un grupo de botones que proporcionan funcionalidades principales como exportar, importar, salir y seleccionar idioma. Se puede exportar un proyecto que haya sido confeccionado en el área de edición de experimentos por el usuario Administrador Acuicultura, obteniendo un archivo con extensión .xml en una ubicación determinada del ordenador. Así mismo con un archivo previamente exportado, se pude importar a un nuevo proyecto de trabajo para reutilizarlo o editarlo teniendo acceso a cada una de sus características y propiedades en el área de edición de experimentos. El botón de salir es un acceso directo para cerrar el área de trabajo o sesión de usuario y nos redirige a la interfaz de inicio de sesión; es recomendable para si queremos cambiar de tipo de usuario, sin necesidad de cerrar la aplicación. Con el concepto de internacionalizar la herramienta

software, se tendrá en la próxima versión la posibilidad de seleccionar el idioma con el que deseamos trabajar tanto en español (idioma por defecto) como en inglés.

- **Área de programación dinámica:** Ubicada por debajo de la barra de herramientas y en la parte superior izquierda. Es el área donde el usuario en cuestión deposita los experimentos seleccionados con un clic desde la lista de experimentos, empleando el recurso de interfaz de usuario de arrastrar y soltar (Drag & Drop), y organizando los experimentos por su orden, tipo y cantidad, en función del consumo de energía de la batería. En esta área se encuentra en la parte inferior el deposito denominado "Eliminar experimento", que como su nombre indica si arrastramos los experimentos hasta allí mediante Drag & Drop, podemos eliminar los experimentos independientemente de su organización por fila o columna.

- **Área de edición de experimentos:** Se encuentra debajo de la barra de herramientas y en la parte superior derecha. Esta a su vez comprende la lista de experimentos, la descripción del experimento seleccionado y el botón de acceso directo a la interfaz de configuración de experimentos.

• **Interfaz de configuración de experimentos:** Si pulsamos en el botón "Nuevo experimento" accedemos a la interfaz de configuración o creación de nuevos experimentos. En esta interfaz conformamos cada experimento con sus propiedades y características como por ejemplo su nombre que puede contener caracteres de letras y números, etiqueta de hasta cuatro caracteres, añadir colores a las etiquetas (rojo, verde, azul, coral, gris, agua, aceituna, violeta, beige, siena). Tiempo de ejecución en segundos, este es el tiempo que se tarda ejecutando un determinado experimento. La energía de ejecución es adimensional, depende del valor en autonomía de la batería, es el consumo en la ejecución del experimento. La energía en espera es adimensional, es el consumo de la batería cuando no se realiza ningún experimento o en la transición de un experimento a otro. El tiempo de espera o retardo en minutos, es representado con un color negro junto a cada etiqueta de experimento en el área de programación dinámica, y es el tiempo o retardo que transcurre de un experimento a otro, o cuando el dispositivo esta en modo "sleep". Además, cada experimento va acompañado de una descripción con toda la información detallada de sus objetivos, frecuencia de ejecución, fechas, códigos y otros. En la interfaz se encuentran dos operandos que por el momento no se tienen en cuanta, pero que serán utilizado en el futuro como propiedades añadidas a próximos experimentos. En la siguiente imagen se muestra un ejemplo de la interfaz.

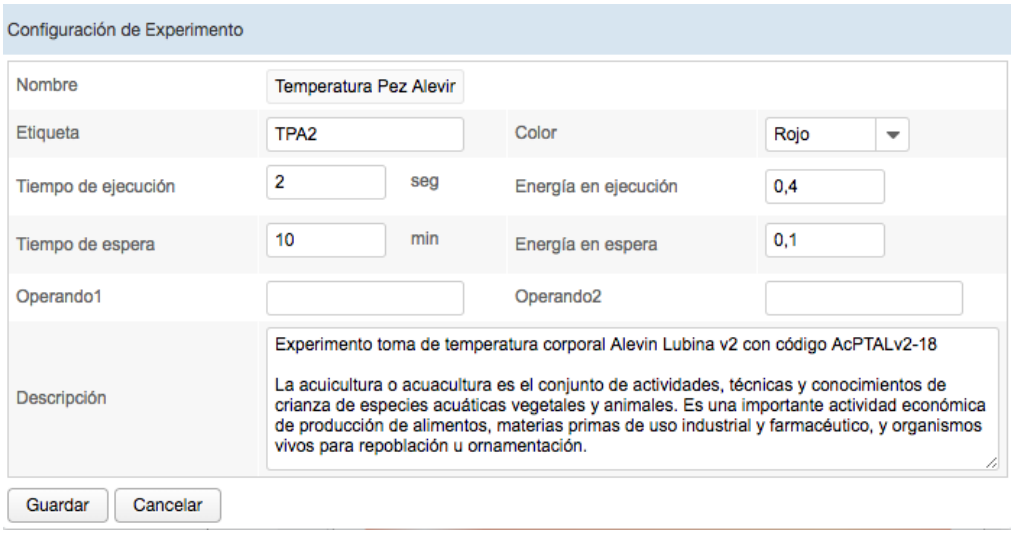


Figura 59. Interfaz de configuración de experimentos.

- **Lista de experimentos:** Está compuesta por todos los experimentos de un proyecto, teniendo acceso a cada experimento desplazándonos por la misma. En ella se muestran los experimentos ordenados de arriba hacia abajo, con su etiqueta, color y un botón por cada uno de ellos con la posibilidad de desecharlo de la lista. Tenemos varias funcionalidades que podemos realizar desde la lista, si hacemos doble clic sobre un experimento accedemos a la interfaz de configuración de experimento y podemos editar cada una de las características del mismo. Con un clic en un determinado experimento observamos su información en el área descriptiva de experimento. Desde la lista podemos seleccionar los experimentos, arrastrándolos y soltándolos en el área de programación dinámica para gestionar y programar nuestro proyecto.

- **Área descriptiva del experimento:** Una vez seleccionado un experimento sólo con un clic poseemos toda la información al respecto del mismo. Es la descripción realizada desde la interfaz de configuración de experimentos.

- **Barra de consumo energético:** La formulación algorítmica de los parámetros de estimación de consumo energético contenido en cada experimento y que conforman un proyecto, denotan el consumo energético total de la batería del dispositivo autónomo de experimentación en acuicultura para dicho proyecto. Por defecto está delimitada en cien por ciento, pero puede ser escalable en el recuadro de número que se encuentra debajo de la barra de batería.

- **Área de análisis gráfico 1:** Ubicado en la esquina inferior izquierda, el gráfico circular permite analizar con facilidad todas las porciones de cantidad de cada uno de los experimentos programados con su color y etiqueta correspondiente respecto al total de un proyecto. Las porciones por experimentos se obtienen y actualizan de la formulación algorítmica desarrollada en la aplicación para los parámetros de estimación de consumo energético.

- **Área de análisis gráfico 2:** Se observa en la esquina inferior derecha el gráfico diagrama de barras con los experimentos programados en función de su consumo energético. Esta gráfica permite comparar cada una de las cantidades de experimentos en cuanto a su valor de consumo. Cada cantidad de experimento representado con su color y etiqueta posee en número la suma de los valores provenientes de la formulación algorítmica de los parámetros de estimación de consumo energético.

6.2.2 Usuario Acuicultura

Una vez iniciado su sesión correspondiente, el Usuario Acuicultura accede a su módulo de interfaz gráfica, la cual es muy similar a la de Administrador Acuicultura, solo con la diferencia gráfica de no poseer el botón "Nuevo experimento". Este usuario no tiene la capacidad funcional de crear un nuevo experimento, ni de editar aquellos parámetros y características de los experimentos que pudiera importar a un proyecto propio. El usuario acuicultura está restringido en comparación con el administrador acuicultura, ya que solo está enfocado en utilizar y programar los proyectos y experimentos creados por el administrador, teniendo solo la posibilidad de modificar de manera limitada los proyectos y exportarlos en un nuevo archivo .xml como un proyecto propio. El resto de funcionalidades y herramientas anteriormente descritas en el apéndice 6.2.1 correspondientes al usuario Administrador Acuicultura, están disponibles para este usuario como se observan en la imagen a continuación.

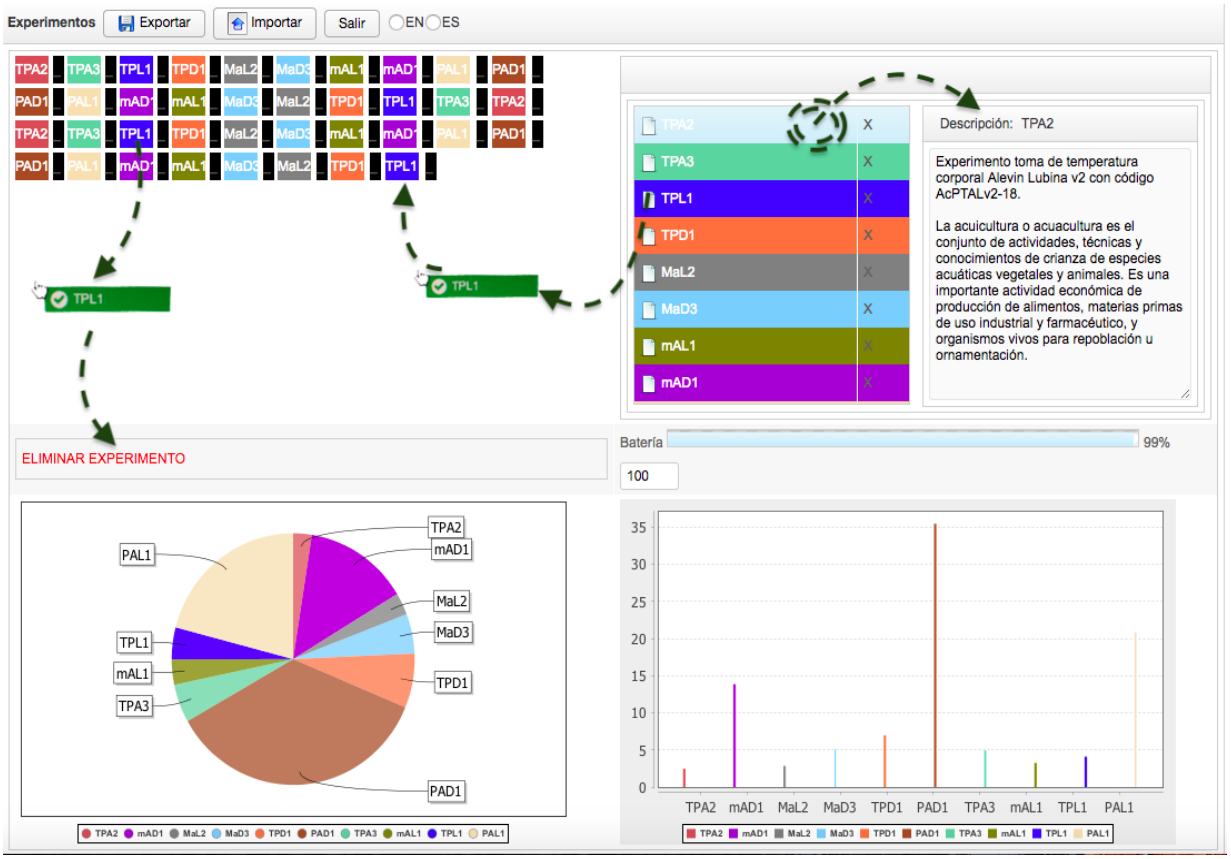


Figura 60. Interfaz gráfica de Usuario Acuicultura

El objetivo principal de cada usuario, ya sea Usuario Acuicultura o Administrador Acuicultura es aprovechar al máximo las funcionalidades de esta aplicación software en la programación, gestión y estimación de la autonomía del DAE en función de la planificación del conjunto de experimentos a realizar. Por tanto, su enfoque va dirigido a investigar y controlar la capacidad de la batería para las sumatoria de los parámetros de los experimentos. Esta herramienta permite ir adicionando cantidades de experimentos con sus parámetros de consumo, mostrándonos su progreso y límite de la autonomía del DAE. Una vez agotada la batería el sistema software nos notifica un mensaje como se observa a continuación.

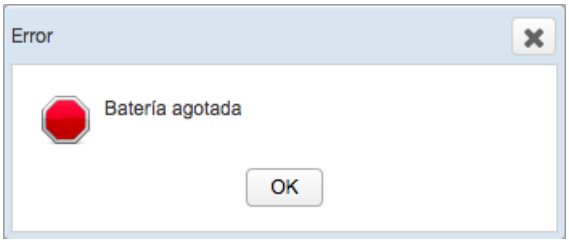


Figura 61. Alerta de consumo total de la batería del DAE

Luego si deseamos restablecer nuestro proyecto con los mismos experimentos importados y con el área dinámica de trabajo limpia, pulsamos el botón F5 de nuestro teclado o el botón del navegador web de restablecer o cargar la página nuevamente. Se ilustra en la siguiente imagen lo antes mencionado.

Figura 62. Restablecimiento inicial del proyecto

6.3 Validación

Se han desarrollado un conjunto de pruebas con motivo de encontrar errores y validar la herramienta software para el futuro uso de sus interfaces gráficas en la programación del DAE. Los usuarios Administrador Acuicultura y Usuario Acuicultura emplearan de forma intuitiva la aplicación, sin conocimientos electrónicos o del funcionamiento del dispositivo, y con una instrucción mínima de los datos del experimento y de programación algorítmica.

Se efectuó un análisis crítico de un grupo de elementos principales como:

- **Botones:** Se comprobó que cada uno de los botones nos dirigiera a los destinos deseados y a las funcionalidades establecidas de manera intuitiva.

- **Redireccionamiento:** Igual que el punto anterior, fue necesario verificar que aquellas acciones que generan un redireccionamiento como el inicio de sesión, nos lleven al lugar adecuado.

- **Interfaces:** Se verificó exhaustivamente que el contenido introducido por el usuario fuera exactamente igual al que se almacena en el sistema, se comprobó que el desplegable de color concordara su nombre con dicho color, se estudió que todos los campos obligatorios incluyendo los que tienen un formato específico fueran correctos y con contenido adecuado. Además, se comparó el comportamiento de cada interfaz de usuario en los diferentes navegadores disponibles en la actualidad.

- **Drag & Drop:** Se realizaron iterativamente todos los posibles casos de usos de este recurso con los experimentos en cuanto a su introducción al área dinámica de trabajo y a la eliminación de los mismos.

- **Gráficas:** Se estudió profundamente cada una de las gráficas de resultados, analizando la fórmula algorítmica empleada en la representación de los datos de cada experimento y del proyecto en general, en concordancia con los datos introducidos. Se compararon las porciones gráficas por color y etiqueta de la suma de cantidades de cada clase de experimentos respecto al total del proyecto y de los valores en número de consumo por cada clase de experimento.

- **Barra de batería:** El progreso de la batería y su límite se verificaron continuamente a lo largo de todo el proyecto, analizando la formulación algorítmica de sus parámetros de consumo en función de la suma de todas las clases de experimentos de un proyecto, respecto a su límite o autonomía prefijada en medida de porcentaje.

En la validación de la herramienta desarrollada participaron alrededor de 20 personas con y sin conocimientos de las materias utilizadas. Con un marcado interés se analizó cuanto de intuitiva resulta la herramienta software para los diferentes tipos de usuarios, en cada uno de los cuatro muestreos por persona que se efectuó a la aplicación. Estos muestreos comprendían dos ciclos de usos de las funcionalidades de cada módulo de usuario por cada una de estas personas, en días indistintos. Se emplearon varios sistemas operativos (Windows, GNU/Linux y macOs), navegadores (Google Chrome, Safari e Internet explorer) y diferentes entornos como la universidad de Las Palmas de Gran Canaria, hogares y centros de trabajos. También se utilizó el proyecto ejemplo de los peces con su conjunto de 10 experimentos previamente definidos.

Este sistema software de programación y estimación del consumo energético para el DAE, es sin duda una herramienta de gran utilidad que posibilita un mayor estudio e investigación en el entorno de la acuicultura sostenible.

CAPITULO 7. CONCLUSIONES Y FUTURO

Se concluye el presente documento de Trabajo Fin de Master con este último capítulo, luego de cumplir los objetivos propuestos de manera global y parcial, en el desarrollo de la herramienta de programación y estimación del consumo energético para el DAE en Acuicultura. Se reflejan las conclusiones a las que se ha llegado tras la realización de este trabajo, además de las mejoras futuras que podría implementarse en la herramienta.

7.1 Conclusiones

A continuación, se enumeran las siguientes conclusiones obtenidas:

- Se cumplió el principal objetivo, construyéndose la aplicación software que permite la descripción gráfica de los experimentos a desarrollar por los usuarios, en base a la configuración de sus parámetros y temporización.
- Se desarrolló una interfaz XML permitiendo abstraer los detalles de implementación de las primitivas de medida y/o cómputo y su temporización del nivel de usuario.
- Se implementó una interfaz gráfica de usuario (GUI) posibilitando programar el dispositivo de experimentación autónoma de forma intuitiva y con conocimientos mínimos de programación.
- Se creó un algoritmo que permite estimar tanto el consumo energético del DAE como el horizonte temporal de ejecución de los experimentos.
- Con la realización de este trabajo se ha podido mejorar las habilidades para el diseño y desarrollo de aplicaciones web a través del lenguaje de programación Java y los lenguajes de marcado ZUML, XML y otros. Además, el framework ZK nos facilitó el aprendizaje para la realización de aplicaciones web haciendo uso del moderno patrón Model-View-ViewModel (MVVM).
- La metodología empleada RUP fue de vital importancia para conseguir los requisitos exigidos. Esta nos posibilitó la guía, organización y control durante todas las fases del proceso de desarrollo del software.
- Este TFM ha permitido interiorizar la importancia social, ecológica y económica que tiene el estudio y la investigación para el desarrollo del entorno de la acuicultura sostenible.

7.2 Trabajos futuro

Las funcionalidades del sistema cumplen con los objetivos propuestos, pero se podrían incluir otras, que quedaron pendientes o que surgieron al terminar la etapa de desarrollo. A continuación, se especifican las aportaciones que podrían ofrecer:

- La funcionalidad de seleccionar varios idiomas quedo pendiente de cumplimentar, incluyendo el inglés. De esta forma cubriríamos las necesidades idiomáticas de algunos usuarios de otras nacionalidades y las normas de internacionalización del software.

- Otra funcionalidad interesante a incorporar en el sistema sería la opción de colocar un calendario, para la planificación temporal de los experimentos gestionados por los

usuarios.

- Incorporar alertas por días y horas del comienzo y finalización de los conjuntos de experimentos programados, a través del recibo de correos electrónicos.
- Implementar una base de datos que posibilite entre otras funcionalidades, personalizar el inicio de sesión de los diferentes usuarios como autogestión de las contraseñas.
- Implementar los parámetros de configuración de experimentos Operando 1 y 2 para ampliar la información y propiedades en los experimentos realizados en acuicultura.

Bibliografía

[1] "European research infrastructures (including e-Infrastructures)", HORIZON 2020, Work Programe 2014 – 2015.

[2] "Selection and Breeding Programs in Aquaculture", ISBN: 978-1-4020-3341-4, Springer, 2015.

[3] http://www.aquaexcel2020.eu/about/overview, último acceso 2019/01/07.

[4] https://es.wikipedia.org/wiki/Interfaz_gráfica_de_usuario, último acceso 2018/08/04.

[5] Vartan Piroumian "Java Gui Development" August 1999.

[6] https://www.techopedia.com/definition/5435/graphical-user-interface-gui, último acceso 2018/08/05.

[7] https://dictionary.cambridge.org/es/diccionario/ingles/graphical-user-interface, último acceso 2018/08/04.

[8] https://proyectoidis.org/memex/, último acceso 2018/09/03.

[9] https://es.wikipedia.org/wiki/Memex, último acceso 2018/09/03.

[10] https://www.xatakaciencia.com/tecnologia/el-memex-de-bush-la-inspiracion-de-los-actuales-ordenadores-y-de-la-www, ultimo acceso 2018/09/03.

[11] https://es.wikipedia.org/wiki/Hipertexto, último acceso 2018/09/04.

[12] https://es.wikipedia.org/wiki/Hiperenlace, último acceso 2018/09/04.

[13] https://en.wikipedia.org/wiki/Ted_Nelson, último acceso 2018/09/04.

[14] https://en.wikipedia.org/wiki/Hypertext_Editing_System, último acceso 2018/09/04.

[15] https://www.w3c.es/, último acceso 2018/09/04.

[16] https://www.wikipedia.org/, último acceso 2018/09/04.

[17] https://es.wikipedia.org/wiki/Sketchpad, último acceso 2018/09/05.

[18] https://www.britannica.com/technology/Sketchpad, último acceso 2018/09/05.

[19] https://en.wikipedia.org/wiki/TX-2, último acceso 2018/09/05.

[20] https://en.wikipedia.org/wiki/Ivan_Sutherland, último acceso 2018/09/05.

[21] https://es.wikipedia.org/wiki/L%C3%A1piz_%C3%B3ptico, último acceso 2018/09/05.

[22] https://es.wikipedia.org/wiki/Dise%C3%B1o_asistido_por_computadora, último acceso 2018/09/05.

[23] http://www.mit.edu/, último acceso 2018/09/05.

[24] https://www.xataka.com/otros/sketchpad-cumple-50-anos-cuando-sutherland-sono-con-el-diseno-asistido-por-ordenador, último acceso 2018/09/05.

[25] https://www.youtube.com/watch?v=6orsmFndx_o&t=743s, último acceso 2018/09/05.

[26] https://en.wikipedia.org/wiki/Douglas_Engelbart, último acceso 2018/09/06.

[27] https://en.wikipedia.org/wiki/NLS_(computer_system), último acceso 2018/09/06.

[28] https://web.stanford.edu/dept/SUL/library/extra4/sloan/mousesite/Archive/ResearchCenter1968/ResearchCenter1968.html, último acceso 2018/09/06.

[29] https://www.computerworld.com/article/2468414/operating-systems/131320-the-mother-of-all-demos-a-retrospective.html#slide2, último acceso 2018/09/06.

[30] https://en.wikipedia.org/wiki/Intelligence_amplification, último acceso 2018/09/06.

[31] https://www.youtube.com/watch?v=yJDv-zdhzMY, último acceso 2018/09/06.

[32] https://en.wikipedia.org/wiki/The_Mother_of_All_Demos, último acceso 2018/09/06.

[33] http://www.dougengelbart.org/content/view/243/253/#Firsts, último acceso 2018/09/06.

[34] https://www.darpa.mil/, último acceso 2018/09/06.

[35] https://www.nasa.gov/, último acceso 2018/09/06.

[36] https://en.wikipedia.org/wiki/SDS_940, último acceso 2018/09/06.

[37] https://es.wikipedia.org/wiki/Hipermedia, último acceso 2018/09/06.

[38] https://en.wikipedia.org/wiki/Bill_English_(computer_engineer), último acceso 2018/09/06.

[39] https://www.youtube.com/watch?v=L1oNBImSX0M&t=393s, último acceso 2018/09/06.

[40] http://wcsa.world/news/world-almanac-event-academy/wcsa-old-events-april-27-2018-xerox-parc-introduces-the-computer-mouse-in-1981, último acceso 2018/09/08.

[41] https://wiredstories.wordpress.com/2014/05/14/la-historia-del-mouse/, último acceso 2018/09/08.

[42] https://en.wikipedia.org/wiki/Trackball, último acceso 2018/09/08.

[43] https://en.wikipedia.org/wiki/Ralph_Benjamin, último acceso 2018/09/08.

[44] https://en.wikipedia.org/wiki/Comprehensive_Display_System, último acceso 2018/09/08.

[45] https://en.wikipedia.org/wiki/DATAR, último acceso 2018/09/08.

[46] https://tecnologia-informatica.com/mouse-raton-historia-futuro/, último acceso 2018/09/08.

[47] https://es.wikipedia.org/wiki/Telefunken, último acceso 2018/09/08.

[48] https://en.wikipedia.org/wiki/Computer_mouse#Rollkugel, último acceso 2018/09/08.

[49] https://www.computerhistory.org/revolution/input-output/14/350/1794, último acceso 2018/09/08.

[50] http://www.computinghistory.org.uk/det/720/bill-english/, último acceso 2018/09/08.

[51] https://www.oldmouse.com/mouse/xerox/alto.shtml, último acceso 2018/09/08.

[52] https://es.wikipedia.org/wiki/Xerox_Alto, último acceso 2018/09/08.

[53] https://es.wikipedia.org/wiki/Xerox_Star, último acceso 2018/09/08.

[54] https://es.wikipedia.org/wiki/Rat%C3%B3n_(inform%C3%A1tica), último acceso 2018/09/08.

[55] https://www.xataka.com/tag/trackball, último acceso 2018/09/08.

[56] https://www.bbc.com/mundo/noticias-37278755, último acceso 2018/09/08.

[57] https://www.microsoft.com/accessories/es-xl/mice, último acceso 2018/09/08.

[58] https://www.apple.com/es/shop/product/MLA02ZM/A/rat%C3%B3n-magic-mouse-2-plata, último acceso 2018/09/08.
http://itechfuture.com/tag/computer-mouse/, último acceso 2018/09/08.
[59] https://en.wikipedia.org/wiki/PARC_(company), último acceso 2018/09/10.
[60] https://www.parc.com/, último acceso 2018/09/10.
[61] https://www.xataka.com/historia-tecnologica/la-fotocopiadora-que-cambio-el-mundo-asi-fue-la-xerox-914, último acceso 2018/09/10.
[62] https://en.wikipedia.org/wiki/WYSIWYG, último acceso 2018/09/10.
[63] https://blog.ensalza.com/diccionario/que-es-wysiwyg/, último acceso 2018/09/10.
[64] https://www.youtube.com/watch?v=pQocN_c2uLI, último acceso 2018/09/10.
[65] https://www.youtube.com/watch?v=Cn4vC80Pv6Q&t=3s, último acceso 2018/09/10.
[66] https://www.youtube.com/watch?v=9H79_kKzmFs, último acceso 2018/09/10.
[67] https://www.youtube.com/watch?v=495nCzxM9PI, último acceso 2018/09/10.
[68] https://es.wikipedia.org/wiki/Alan_Kay, último acceso 2018/09/10.
[69] http://historiadelatabet.blogspot.com/p/blog-page.html, último acceso 2018/09/11.
[70] https://www.britannica.com/biography/Alan-Kay, último acceso 2018/09/11.
[71] https://www.computerhistory.org/fellowawards/hall/alan-kay/, último acceso 2018/09/11.
[72] https://en.wikipedia.org/wiki/Smalltalk, último acceso 2018/09/11.
[73] https://www.um.es/web/universidad/doctores-honoris-causa/alan-kay, último acceso 2018/09/11.
[74] https://en.wikipedia.org/wiki/Dynabook, último acceso 2018/09/11.
[75] http://wiki.laptop.org/go/Alan_Kay, último acceso 2018/09/11.
[76] https://www.pinterest.es/pin/536772849316191402/, último acceso 2018/09/11.
[77] https://www.linuxadictos.com/analisis-entornos-escritorio-2015.html, último acceso 2018/09/11.
[78] https://www.computerhistory.org/revolution/input-output/14/347, último acceso 2018/09/11.
[79] https://www.computerhistory.org/collections/catalog/X1552.98A, último acceso 2018/09/11.
[80] https://commons.wikimedia.org/wiki/File:Apple_first_logo.png, último acceso 2018/09/12.
[81] https://en.wikipedia.org/wiki/Apple_Inc., último acceso 2018/09/12.
[82] https://www.apple.com/es/, último acceso 2018/09/12.
[83] https://prezi.com/6xfyfaszuojp/sistemas-operativos-de-apple/, último acceso 2018/09/12.
[84] https://es.wikipedia.org/wiki/Apple_I, último acceso 2018/09/12.
[85] https://es.wikipedia.org/wiki/Apple_II, último acceso 2018/09/12.
[86] https://es.wikipedia.org/wiki/Apple_DOS, último acceso 2018/09/12.
[87] https://en.wikipedia.org/wiki/Apple_SOS, último acceso 2018/09/12.
[88] https://en.wikipedia.org/wiki/Apple_ProDOS, último acceso 2018/09/12.
https://en.wikipedia.org/wiki/Apple_GS/OS, último acceso 2018/09/12.
[89] https://es.wikipedia.org/wiki/Apple_Lisa, último acceso 2018/09/12.

[90] https://en.wikipedia.org/wiki/Apple_Lisa#Lisa_2, último acceso 2018/09/12.
[91] https://de.wikipedia.org/wiki/Lisa_OS, último acceso 2018/09/12.
[92] https://es.wikipedia.org/wiki/Macintosh, último acceso 2018/09/12.
[93] https://en.wikipedia.org/wiki/Macintosh, último acceso 2018/09/12.
[94] https://en.wikipedia.org/wiki/System_1, último acceso 2018/09/12.
[95] http://applemuseum.bott.org/sections/os.html, último acceso 2018/09/12.
[96] https://es.wikipedia.org/wiki/Mac_OS, último acceso 2018/09/12.
[97] https://es.wikipedia.org/wiki/MacOS, último acceso 2018/09/12.
[98] https://es.wikipedia.org/wiki/Historia_de_Mac_OS_X, último acceso 2018/09/12.
[99] https://www.apple.com/la/macos/what-is/, último acceso 2018/09/12.
[100] http://pcmuseum.tripod.com/lisadsk.htm, último acceso 2018/09/12.
[101] http://www.maestrosdelweb.com/historia-y-evolucion-del-sistema-operativo-mac-os/, último acceso 2018/12/14.
[102] https://miescapedigital.com/como-instalar-macos-mojave-en-macs-no-compatibles/, último acceso 2018/09/12.
[103] https://es.wikipedia.org/wiki/IOS, último acceso 2018/09/12.
[104] https://www.apple.com/es/watchos/watchos-5/, último acceso 2018/09/12.
[105] https://developer.apple.com/tvos/, último acceso 2018/09/12.
[106] https://www.apple.com/es/ios/ios-12/, último acceso 2018/09/12.
[107] http://apple.wikia.com/wiki/File:IOS_12_Homescreen_iPhone_X.png, último acceso 2018/09/12.
[108] https://9to5mac.com/2015/03/24/how-to-hide-rearrange-apple-tv-channels/, último acceso 2018/09/12.
https://www.wareable.com/apple/apple-watch-series-4-release-date-price-specs, último acceso 2018/09/12.
[109] http://apple.wikia.com/wiki/File:IOS_12_Homescreen_iPhone_X.png, último acceso 2018/09/12.
[110] https://9to5mac.com/2015/03/24/how-to-hide-rearrange-apple-tv-channels/, último acceso 2018/09/12.
[111] https://www.wareable.com/apple/apple-watch-series-4-release-date-price-specs, último acceso 2018/09/12.
[112] https://en.wikipedia.org/wiki/File:Microsoft_logo_(1975).svg, último acceso 2018/09/14.
[113] https://www.microsoft.com/es-es, último acceso 2018/09/14.
[114] https://es.wikipedia.org/wiki/Microsoft, último acceso 2018/09/14.
[115] https://en.wikipedia.org/wiki/Microsoft, último acceso 2018/09/14.
[116] https://en.wikipedia.org/wiki/Xenix, último acceso 2018/09/14.
[117] https://es.wikipedia.org/wiki/DOS, último acceso 2018/09/14.
[118] https://es.wikipedia.org/wiki/IBM_PC, último acceso 2018/09/14.
[119] https://es.wikipedia.org/wiki/DOS, último acceso 2018/09/14.
[120] https://en.wikipedia.org/wiki/OS/2, último acceso 2018/09/14.
[121] https://es.wikipedia.org/wiki/Windows_1.0, último acceso 2018/09/14.
[122] https://es.wikipedia.org/wiki/Windows_2.0, último acceso 2018/09/14.

[123] https://en.wikipedia.org/wiki/Windows_3.0, último acceso 2018/09/14.
[124] https://es.wikipedia.org/wiki/Windows_NT_4.0, último acceso 2018/09/14.
[125] https://es.wikipedia.org/wiki/Windows_95, último acceso 2018/09/14.
[126] https://es.wikipedia.org/wiki/Windows_98, último acceso 2018/09/14.
[127] https://es.wikipedia.org/wiki/Windows_XP, último acceso 2018/09/14.
[128] https://en.wikipedia.org/wiki/Windows_Vista, último acceso 2018/09/14.
[129] https://es.wikipedia.org/wiki/Windows_7, último acceso 2018/09/14.
[130] https://es.wikipedia.org/wiki/Windows_8, último acceso 2018/09/14.
[131] https://es.wikipedia.org/wiki/Windows_10, último acceso 2018/09/14.
[132] https://es.wikipedia.org/wiki/Microsoft_Edge, último acceso 2018/09/14.
[133] https://www.office.com/?omkt=es-es, último acceso 2018/09/14.
[134] http://home.bt.com/tech-gadgets/computing/15-things-you-didnt-know-about-microsoft-windows-10-11363944811339, último acceso 2018/09/14.
[135] http://es.windows.wikia.com/wiki/Archivo:Windows_95.gif, último acceso 2018/09/14.
[136] https://miescapedigital.com/novedades-windows-10-spring-creators-update/, último acceso 2018/09/14.
[137] https://en.wikipedia.org/wiki/Xbox, último acceso 2018/09/14.
[138] https://www.microsoft.com/es-es/surface, último acceso 2018/09/14.
[139] https://es.wikipedia.org/wiki/Windows_Mobile, último acceso 2018/09/14.
[140] https://es.wikipedia.org/wiki/Sistema_Xbox_One, último acceso 2018/09/14.
[141] https://www.amazon.es/Nokia-Lumia-520-Smartphone-Dual-Core/dp/B00C4U4FNC, último acceso 2018/09/12.
[142] https://www.softonic.com/articulos/2014-03-01-xbox-one-actualizacion-marzo, último acceso 2018/09/12.
[143] https://blogs.technet.microsoft.com/jonjor/2014/10/16/dual-boot-surface-3-with-windows-10/, último acceso 2018/09/12.
[144] https://linuxforallsite.wordpress.com/2015/12/03/que-es-una-distribucion-de-gnu-linux/, último acceso 2018/09/15.
[145] https://www.gnu.org/distros/free-distros.html, último acceso 2018/09/15.
[146] https://en.wikipedia.org/wiki/Linux, último acceso 2018/09/15.
[147] https://es.wikipedia.org/wiki/GNU, último acceso 2018/09/15.
[148] https://es.wikipedia.org/wiki/GNU/Linux, último acceso 2018/09/15.
[149] https://www.linux.org/, último acceso 2018/09/15.
[150] https://es.wikipedia.org/wiki/Distribuci%C3%B3n_Linux, último acceso 2018/09/15.
[151] https://es.wikipedia.org/wiki/Linux-VServer, último acceso 2018/09/15.
[152] https://es.m.wikipedia.org/wiki/Archivo:GNOME-escritorio-1.x.png, último acceso 2018/09/15.
[153] https://fossbytes.com/red-hat-enterprise-linux-6-8-rhel-features/, último acceso 2018/09/15.
[154] http://www.ubuntubuzz.com/2018/03/how-to-install-ubuntu-1804-lts-bionic-beaver.html, último acceso 2018/09/15.
[155] https://es.wikipedia.org/wiki/Android, último acceso 2018/09/15.

[156] https://www.android.com/, último acceso 2018/09/15.

[157] https://wearos.google.com/#hands-free-help, último acceso 2018/09/15.

[158] https://es.wikipedia.org/wiki/Wear_OS, último acceso 2018/09/15.

[159] https://es.wikipedia.org/wiki/WebOS, último acceso 2018/09/15.

[160] https://www.lg.com/es/posventa/guias-y-soluciones/television/como-saber-que-version-de-webos-tengo, último acceso 2018/09/15.

[161] https://www.android.com/phones/, último acceso 2018/09/15.

[162] https://www.lg.com/pa/tvs/lg-43LH6000, último acceso 2018/09/15.

[163] https://topesdegama.com/noticias/apps-software/diseno-android-wear-actualizado, último acceso 2018/09/15.

[164] https://arstechnica.com/gadgets/2018/09/review-googles-wear-os-2-0-cant-fix-its-obsolete-smartwatch-hardware/, último acceso 2018/09/15.

[165] https://www.muylinux.com/2018/07/30/canonical-aplicaciones-electron-snap/, último acceso 2018/09/17.

[166] https://electronjs.org/, último acceso 2018/09/17.

[167] https://thermodynamicprocess.wordpress.com/2018/01/14/wxwidgets-library/, último acceso 2018/12/17

[168] https://www.wxwidgets.org/, último acceso 2018/12/17.

[169] https://blog.desdelinux.net/llega-gtk-con-mejoras-en-su-nueva-version-3-24-1/, último acceso 2018/09/17.

[170] https://www.gtk.org/, último acceso 2018/09/17.

[171] https://www.digiajay.com/labview-logo/, último acceso 2018/09/18.

[172] http://www.ni.com/es-es/shop/labview.html, último acceso 2018/09/18.

[173] https://es.wikipedia.org/wiki/Archivo:Tcl-Tk_universal_scripting.svg, último acceso 2018/09/18.

[174] https://www.tcl.tk/, último acceso 2018/09/18.

[175] https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_m%C3%B3vil, último acceso 2018/09/19.

[176] https://www.masdigital.net/servicios-mas-digital/aplicaciones-moviles, último acceso 2018/09/19.

[177] https://play.google.com/store, último acceso 2018/09/19.

[178] https://developer.android.com/studio/?hl=es-419, último acceso 2018/09/19.

[179] https://es.wikipedia.org/wiki/Android_Studio, último acceso 2018/09/19.

[180] https://www.apple.com/es/ios/app-store/, último acceso 2018/09/19.

[181] https://docs.swift.org/swift-book/index.html, último acceso 2018/09/19.

[182] https://developers.google.com/web/progressive-web-apps/, último acceso 2018/09/19.

[183] https://en.wikipedia.org/wiki/Progressive_web_applications, último acceso 2018/09/19.

[184] https://www.genbeta.com/desarrollo/los-10-frameworks-librerias-mas-populares-de-javascript, último acceso 2018/09/19.

[185] https://kotlinlang.org/, último acceso 2018/09/19.

[186] http://www.cplusplus.com/, último acceso 2018/09/19.

[187] https://www.java.com/es/download/, último acceso 2019/01/07.
[188] https://www.eclipse.org/ide/, último acceso 2019/01/o7.
[189] https://en.wikipedia.org/wiki/Objective-C, último acceso 2018/09/20.
[190] https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/ObjectiveC.html, último acceso 2018/09/20.
[191] https://developer.apple.com/xcode/ide/, último acceso 2018/09/20.
[192] http://www.reactnative.com/, último acceso 2018/09/20.
[193] https://reactjs.org/, último acceso 2018/09/20.
[194] https://www.javascript.com/, último acceso 2018/09/20.
[195] https://www.w3.org/TR/html5/, último acceso 2018/09/21.
[196] https://www.w3.org/Style/CSS/, último acceso 2018/09/21.
[197] https://drafts.csswg.org/, último acceso 2018/09/21.
[198] https://ionicframework.com/, último acceso 2018/09/21.
[199] https://visualstudio.microsoft.com/es/xamarin/, último acceso 2018/09/21.
[200] https://visualstudio.microsoft.com/es/vs/, último acceso 2018/09/21.
[201] https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto, último acceso 2018/09/22.
[202] https://es.wikipedia.org/wiki/Front-end_y_back-end, último acceso 2018/09/22.
[203] https://devcode.la/blog/frontend-y-backend/, último acceso 2018/09/22.
[204] https://www.strato.es/wordpress-hosting/wordpress-frontend/, último acceso 2018/09/12.
[205] https://es.wikipedia.org/wiki/HTML5, último acceso 2018/09/22.
[206] https://en.wikipedia.org/wiki/Cascading_Style_Sheets, último acceso 2018/09/22.
[207] https://es.wikipedia.org/wiki/JavaScript, último acceso 2018/09/22.
[208] https://www.apache.org/, último acceso 2018/09/24.
[209] https://www.nginx.com/, último acceso 2018/09/24.
[210] https://www.eclipse.org/jetty/download.html, último acceso 2018/09/24.
[211] https://www.oracle.com/technetwork/es/middleware/glassfish/overview/index.html, último acceso 2018/09/24.
[212] http://www.wildfly.org/, último acceso 2018/09/24.
[213] http://php.net/manual/es/intro-whatis.php, último acceso 2018/09/24.
[214] https://symfony.com/, último acceso 2018/09/24.
[215] https://www.ruby-lang.org/es/, último acceso 2018/09/24.
[216] https://rubyonrails.org/, último acceso 2018/09/24.
[217] https://spring.io/, último acceso 2018/09/24.
[218] https://www.python.org/, último acceso 2018/09/24.
[219] https://www.djangoproject.com/ último acceso 2018/09/24.
[220] https://nodejs.org/es/, último acceso 2018/09/24.
[221] https://www.mysql.com/, último acceso 2018/09/28.
[222] https://www.mongodb.com/es, último acceso 2018/09/28.
[223] https://es.wikipedia.org/wiki/Base_de_datos, último acceso 2018/09/28.

[224] https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos, , último acceso 2018/09/28.

[225] https://www.zkoss.org/, último acceso 2019/01/08.

[226] https://es.wikipedia.org/wiki/AJAX, último acceso 2018/09/28.

[227] https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n), último acceso 2018/09/28.

[228] Vartan Piroumian "Java Gui Development" August 1999

[229] https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html, último acceso 2018/09/28.

[230] https://en.wikipedia.org/wiki/Java_Development_Kit, último acceso 2018/09/28.

[231] https://en.wikipedia.org/wiki/ZK_(framework), último acceso 2018/09/28.

[232] https://www.tiobe.com/tiobe-index/, último acceso 2018/10/02.

[233] https://insights.stackoverflow.com/survey/2018/, último acceso 2018/10/02.

[234] http://www.uml.org/, último acceso 2018/09/12

[235] https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado, last acces: 2018/12/14.

[236] Ahmad K. Shuja and Jochen Krebs, "IBM Rational Unified Process Reference and Certification Guide: Solutions Designer", 2008.

[237] https://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational, last acces: 2018/11/04.

[238] https://en.wikipedia.org/wiki/Rational_Software, last acces: 2018/12/15.

[239] http://www.argentuminc.com/?p=1994, último acceso: 2018/12/15.

[240] https://www.researchgate.net/publication/299506242_METODOLOGIAS_TRADICIONALES_VS_METODOLOGIAS_AGILES, último acceso: 2018/12/15.

[241] https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software, último acceso: 2018/12/15.

[242] https://es.wikipedia.org/wiki/Proceso_del_desarrollo_del_software, último acceso: 2018/12/15.

[243] https://www.apple.com/es/macbook-air/, último acceso: 2018/7/4.

[244] https://support.apple.com/kb/DL1966?locale=es_ES, último acceso: 2018/18/20.

[245] https://www.google.com/intl/es_ALL/chrome/, último acceso: 2018/10/03.

[246] https://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java, último acceso: 2018/08/03.

[247] https://es.wikipedia.org/wiki/WildFly, último acceso: 2018/08/03.

[248] https://www.w3.org/XML/, último acceso: 2018/10/08.

[249] http://www.mundolinux.info/que-es-xml.htm, último acceso: 2018/10/08.

[250] https://www.xml.com/, último acceso: 2018/10/08.

[251] https://es.wikipedia.org/wiki/Eclipse_(software), último acceso: 2018/10/08.

[252] https://products.office.com/es-es/word, último acceso: 2018/08/22.

[253] https://es.wikipedia.org/wiki/Microsoft_Word, último acceso: 2018/08/22.

[254] https://support.apple.com/es-es/guide/preview/welcome/mac, último acceso: 2018/08/22.

[255] http://staruml.io/, último acceso: 2018/12/26.

[256] https://en.wikipedia.org/wiki/StarUML, último acceso: 2018/08/22.

[257] https://www.invisionapp.com/, último acceso: 2018/08/22.

[258] https://blog.gfi.es/mvvm/, último acceso: 2018/09/05.

[259] https://www.zkoss.org/wiki/ZK_Developer%27s_Reference/MVVM/Data_Binding, último acceso: 2018/09/05.