Trabajo Fin de Grado

Escuela de Ingeniería Informática

Universidad de Las Palmas de Gran Canaria

Desarrollo del juego del Arkanoid en entorno web

Jonatan García Ferrera

Las Palmas de Gran Canaria

Diciembre 2013

Trabajo Fin de Grado realizado en la Escuela de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria, para la consecución del título de Graduado en Ingeniería Informática.

Título: Desarrollo del juego del Arkanoid en entorno web.

Alumno: Jonatan García Ferrera

Tutor: Javier Sánchez Pérez

Fecha: diciembre 2013

A mi familia.

Agradecimientos

Este documento es el final de un período, y no se habría escrito sin todas las personas que han ayudado, directa o indirectamente, a superar con éxito todas las materias que componen esta titulación.

En primer lugar, a mi familia, que me han apoyado incondicionalmente en los momentos más duros, dándome el aliento para seguir adelante. Y sobre todo a mis padres, Antonio y Mita, por darme la oportunidad de elegir mis estudios y mi profesión, y por animarme en mis momentos más pesimistas.

En segundo lugar, agradecer a mis amigos, Cristian, Jesús y Samuel, la paciencia que han tenido conmigo y las horas que han tenido que escucharme hablando sobre los problemas que tenía en mis prácticas, informes, y por último, en el desarrollo del Trabajo de Fin de Grado, y en este documento.

En tercer lugar, a mi tutor, Javier, por ayudarme a poner en práctica todos los conocimientos adquiridos durante mis estudios, y por acompañarme en los primeros pasos de mi vida profesional.

También me gustaría agradecer a mi compañero Josué, que participó en este trabajo en su período de prácticas de externas.

Por último, me gustaría agradecer a todos los que siempre han estado ahí, y que de una forma u otra me han apoyado para seguir adelante.

A todos, muchas gracias.

Índice

Prefacio

La historia de los videojuegos tiene su origen en la década de 1940 cuando, tras el fin de la Segunda Guerra Mundial, las potencias vencedoras construyeron las primeras supercomputadora programables como el ENIAC, de 1946. Los primeros intentos por implementar programas de carácter lúdico (inicialmente programas de ajedrez) no tardaron en aparecer, y se fueron repitiendo durante las siguientes décadas.

Los primeros videojuegos modernos aparecieron en la década de los 60, y desde entonces el mundo de los videojuegos no ha cesado de crecer y desarrollarse con el único límite que le ha impuesto la creatividad de los desarrolladores y la evolución de la tecnología.

El más inmediato reflejo de la popularidad que ha alcanzado el mundo de los videojuegos en las sociedades contemporáneas lo constituye una industria que da empleo a 120.000 personas y que genera unos beneficios multimillonarios que se incrementan año tras año. Al igual que ocurriera con el cine y la televisión, el videojuego ha logrado alcanzar en apenas medio siglo de historia el estatus de medio artístico, y semejante logro no ha tenido lugar sin una transformación y evolución constante del concepto mismo de videojuego y de su aceptación. Nacido como un experimento en el ámbito académico, logró establecerse como un producto de consumo de masas en tan sólo diez años, ejerciendo un formidable impacto en las nuevas generaciones que veían los videojuegos con un novedoso medio audiovisual que les permitiría protagonizar en adelante sus propias historias.

El trabajo expuesto en este documento refleja la realización de un proyecto de ingeniería del software, con el objetivo de realizar una aplicación web que implemente el clásico juego del arkanoid.

1.- Introducción

El Arkanoid es un videojuego arcade desarrollado por Taito en 1986, basado en los Breakout de Atari de los años 70. A causa de la popularidad del juego, cuatro versiones se desarrollaron, entre los años 1987 y 1997, para el mercado de las máquinas arcade.

En la actualidad, Arkanoid continúa siendo un juego muy popular, y es clonado frecuentemente. Y aunque la mayoría de empresas lo han clonado para sus máquinas, Arkanoid o sus secuelas no aparecen en ninguna de las recientes compilaciones de Taito Memories o Taito Legends, posiblemente debido a acciones legales de Atari.

El juego consiste en que el jugador controla una pequeña plataforma que impide que las bolas salgan de la zona de juego, haciéndolas rebotar. En la parte superior de la zona de juego hay ladrillos que desaparecen al ser tocados por la bola. Cuando no queda ningún ladrillo, el jugador pasa al siguiente nivel, donde aparece otro patrón de bloques. Existen distintas variaciones en cuanto a los ladrillos, y además existen cápsulas que modifican la plataforma que controla el usuario.

En este proyecto se ha desarrollado el clásico juego del arkanoid con una arquitectura cliente - servidor de dos capas y tres niveles. En la capa de presentación se encontrará la aplicación de juego, junto a las clases que conectarán con la capa de negocio. En la capa de negocio se encuentran las clases que interactúan con el disco duro, guardando y cargando objetos serializables.

El juego en sí constará de varios mundos, y cada mundo tendrá una serie de fases. Se prevé que el primer mundo sea de prueba para tratar de ganar adeptos al sistema, y que para poder acceder a los siguientes mundos, el usuario tenga que registrarse y pagar por cada uno.

Además, el juego tendrá la funcionalidad de guardar la partida actual, pero sólo de los usuarios registrados. Permitiendo reanudar la partida anterior, o iniciar una nueva al ejecutar la aplicación.

Por otro lado, y con el fin de explotar el concepto de “gamificación”, el sistema permitirá ver las mejores puntuaciones que han obtenido otros usuarios de la aplicación. El concepto “gamificación” es el uso del pensamiento y la mecánica de la jugabilidad en contextos ajenos a los juegos, con el fin de que las personas adopten cierto comportamiento.

En cuanto al modelo de negocio escogido, se pueden distinguir tres tipos de métodos para obtener ingresos con el sistema. En primer lugar, se crea un sistema de compra de mundos, en el que el primer mundo es gratuito, pero para jugar a más mundos, el usuario debe comprar cada mundo por separado. En segundo lugar, se pueden incluir banners de publicidad en la página web. Y por último, se pretende incluir

banners de publicidad que tengan enlaces a páginas que deseen publicitarse, a partir de lo que se genera un sistema de redirección de tráfico a las páginas de los clientes.

Para el proceso de desarrollo de este proyecto se ha elegido el Proceso Unificado de Desarrollo (PUD).

1.1.- Motivación y objetivos

Los objetivos principales de este trabajo son:

- Desarrollo del juego del Arkanoid en web.
- Registro de usuarios.
- Registro de puntuaciones.
- Descarga de contenidos del juego a través de la red.
- Aplicar una metodología de desarrollo de software para el desarrollo de la aplicación.

1.2.- Aportaciones

Con el desarrollo de este trabajo se pretende implementar un sistema novedoso de juego, en el que la aplicación es gratuita, y el usuario paga únicamente por los contenidos que descargue. Además, se pretende que el usuario pueda jugar en distintos equipos y recuperar el juego tal y cómo lo dejó la última vez que jugó, sin importar dónde lo hiciese.

1.3.- Organización del documento

A lo largo del contenido de este documento se describirá el proceso necesario para acometer el desarrollo de la aplicación resultante, desde las etapas iniciales de investigación y análisis de requisitos hasta las etapas finales de diseño e implementación.

El documento está dividido en seis capítulos centrados en el trabajo realizado en el proyecto, y posteriormente se detalla información más técnica y específica en un conjunto de apéndices.

El primer capítulo está dedicado a dar una introducción a los conceptos principales que se tratarán en este trabajo, y también se detalla el problema a resolver. También se incluyen los objetivos que se pretenden con la realización del trabajo, y las aportaciones del mismo.

El segundo capítulo está centrado en mostrar el estado actual del tema en cuestión, haciendo un recorrido por las diferentes versiones del juego del arkanoid que se han desarrollado a lo largo de la historia.

En el tercer capítulo se detallan los recursos empleados para la consecución de este trabajo.

El cuarto capítulo se centra en la planificación del trabajo, y engloba el plan de trabajo y el presupuesto necesario para construir la aplicación. Además, se expone la metodología de desarrollo empleada, para posteriormente realizar una descomposición temporal de las actividades que componen el trabajo. Por último, se hace una estimación económica del coste de desarrollo de la aplicación.

En el quinto capítulo se descomponen todas las etapas del desarrollo del trabajo siguiendo la metodología descrita en el capítulo anterior.

Por último, en el sexto capítulo, se encuentran las conclusiones obtenidas tras finalizar el trabajo, ya sea en cuanto al trabajo como a nivel personal. Además, se incluyen distintas opciones para ampliar el sistema en versiones posteriores.

El documento se cierra con los apéndices técnicos, apéndices legislativos, competencias y la bibliografía empleada en este proyecto.

2.- Estado del arte

Dentro del ambiente tecnológico industrial, se entiende como "estado del arte", "estado de la técnica" o "estado de la cuestión", todos aquellos desarrollos de última tecnología realizados a un producto, que han sido probados en la industria y han sido acogidos y aceptados por diferentes fabricantes.

En esta sección se expondrán los diferentes productos en los que se basa este proyecto, y además, algunos de los productos actuales similares al que dará como resultado la consecución del mismo.

2.1.- Videojuegos

Los videojuegos rodean e infunden su espíritu a todos los ámbitos de la vida; la cultura, la sociedad, la economía. Son una industria de diez mil millones de dólares, un modo de vida para jugadores profesionales y sus patrocinadores, un modo de expresión personal, una herramienta de aprendizaje, un entrenamiento para la guerra... Bruce Sterling dijo de la guerra del Golfo: "Es una guerra a lo Nintendo para la generación Nintendo". Y es verdad que en las imágenes televisadas de las bombas guiadas a distancia brillaba la estela catódica de Missile Command (Atari), Scramble (Taito) y X-Wing (Lucasarts). Este último título basado en la serie de películas La guerra de las galaxias (Star Wars, George Lucas, 1977).

Desde su nacimiento los videojuegos han inspirado películas (Tron, Steven Lisberger, 1982; Final Fantasy, Hironobu Sakaguchi, 2001), y las películas, videojuegos: Blade Runner, dirigida por Ridley Scott en 1982, se convirtió a mediados de los 90 en una aventura gráfica de notable éxito. El videojuego como fenómeno también ha propiciado reflexiones cinematográficas. Nirvana (Gabriele Salvatore, 1997) y Existenz (David Cronenberg, 1999) son películas en las que la pregunta "¿qué es un videojuego?" se transforma finalmente en "¿qué es la realidad?".

El videojuego está tan integrado en la cultura popular de fin de milenio que hay películas cuya estética es afín a la de los videojuegos, sin estar inspiradas en ningún título en particular. Es el caso de la batalla final de Blade (Stephen Norrington, 1999), cuyos tiros de cámara recuerdan a los de Tekken, o las 'vidas infinitas' en las que se centra Atrapado en el tiempo (Groundhog Day, Harold Ramis, 1993). Algunos visionarios son capaces de crear imágenes que prefiguran e inspiran nuevos lenguajes cinéticos. De este modo Matrix (Andy y Larry Wachowski, 1999) abre el camino con su brillante uso del bullet time para juegos como Max Payne (Remedy Entertainment, 2001).

Al igual que el gran arte popular del siglo XX ha sido el cine - una tecnología decimonónica -, el videojuego será el gran arte popular del siglo XXI. Entre los juegos

de este mismo año 2001 hay títulos para solteros, para madres de familia, para niñas y para adultos, para el gran público y para los connoisseurs más exquisitos.

El Arte con mayúscula, el de las galerías y los museos, también ha ido aceptando progresivamente este nuevo lenguaje. Los artistas que ahora tienen entre 20 y 35 años han crecido jugando con el Atari, la Nintendo, la Sega, la Playstation. De Space Invaders a Grand Prix, de Pong a Quake, el arte contemporáneo fagocita las referencias de la cultura pop y las devuelve recubiertas de una capa teórica, rellenas de significados herméticos, troceadas y recompuestas en collages que tratan el videojuego como texto fundacional de una estética renovada.

El videojuego, un medio de comunicación fundamentalmente comercial, está lleno de resquicios donde brilla la expresión personal de individuos excepcionales. Peter Molineux, Will Wright e Hideo Kojima tienen tanto de artistas como de ingenieros o productores, y su obra es tan personal como la de los directores cinematográficos elevados a la categoría de autores por la escuela crítica del 'Cahiers du Cinema'.

2.2.- Historia del Arkanoid

Arkanoid es un videojuego de arcade desarrollado por Taito en 1986. Está basado en los Breakout de Atari de los años 70. El jugador controla una pequeña plataforma, conocida como "Nave Espacial Vaus", que impide que una bola salga de la zona de juego, haciéndola rebotar. En la parte superior hay "ladrillos" o "bloques", que desaparecen al ser tocados por la bola. Cuando no queda ningún ladrillo, el jugador pasa al siguiente nivel, donde aparece otro patrón de bloques.

Existen distintas variaciones (ladrillos que hay que golpear varias veces para que desaparezcan, naves enemigas, etc) y cápsulas que mejoran a la Vaus (azul: expandiéndola, rojo: equipándola con un cañón láser, rosado: pasando directamente al siguiente nivel, celeste: aumentando el número de bolas, plomo: aumentando vidas, verde: atrapar pelota, naranja: más lenta la pelota) Todos los niveles son verdaderamente coloridos y tienen su propio estilo. Existen pequeñas figuras que pueden golpearse por algunos puntos e incluso son diferentes por cada nivel.

Se cuenta con 3 naves al principio, y después de que se pierden todas, se deberá comenzar de nuevo desde el inicio. En la pantalla número 33, el último nivel, el jugador se enfrenta al principal enemigo del juego, Doh. Una vez que el jugador llega a dicho nivel, debe vencer a Doh con el número de Vauses que tenga en reserva, si no, el juego termina y el jugador ha perdido.

A causa de la popularidad del juego, cuatro versiones se desarrollaron para el mercado de las máquinas arcade: Arkanoid, TournamentArkanoid y Revenge of Doh (Arkanoid II), ambas en 1987, y ArkanoidReturns en 1997. La mayoría de computadores de 8 bits (ZX Spectrum, Amstrad CPC 464, Commodore 64, MSX, Atari

8-bit, Apple II...) eran muy populares en Europa en los 80. Un puerto de consola en la NES también era popular, así que el juego fue portado para computadoras de 16 bits como Commodore Amiga, Atari ST, Apple IIGS o IBM-PC. Se desarrolló un puerto para el TRS-80 en 1989. Una versión para SNES, llamada Arkanoid: DohitAgain, se lanzó en 1997. ArkanoidReturns y su secuela, ArkanoidReturns 2000, se lanzaron en Japón para PlayStation. Las versiones de 16 bits tenían exactamente los mismos gráficos que el juego original. La conversión de Arkanoid para Commodore 64 es conocida por ser el primer juego para dicho sistema que incluía música que usaba samples digitalizados (compuestos por Martin Galway).

Los controles usados difieren entre las máquinas, y algunas conversiones permitían múltiples métodos de control. Los dos métodos básicos de control eran: digital y analógico. Los controles digitales (muchos joysticks y teclados) son considerados menos convenientes que los analógicos (como ratones y trackballs): mientras que los digitales limitan al jugador a una única velocidad, los analógicos permiten mover la Vaus casi a cualquier velocidad deseada a través del a pantalla. La versión de Arkanoid para NES era inicialmente empaquetada con el que se considera uno de los mandos menos comunes de dicha consola: el VausController, un pequeño mando con un único botón, una pequeña rueda (con ángulo de giro limitado), un puerto y el logo de Taito. Aunque podía jugarse con el mando digital estándar de NES, el juego óptimo se conseguía con el VausController.

Arkanoid continúa siendo un juego muy popular, y es clonado frecuentemente para títulos de freeware y shareware. La mayoría de empresas también lo han clonado habitualmente para sus máquinas. Sin embargo, Arkanoid o sus secuelas no aparecen en ninguna de las recientes recopilaciones TaitoMemories o TaitoLegends, posiblemente debido a acciones legales de Atari.

2.3.- Competencia

Tras el nacimiento de este juego a finales de los 80, han salido a la luz muchas versiones del mismo, y en múltiples formatos. Se han creado aplicaciones para plataformas como Nintendo, e incluso para Playstation. Y, en los últimos años, con el nacimiento de los smartphones, han salido al mercado aplicaciones que lo versionan.

Incluso el buscador Google, para celebrar el 37 aniversario de este juego, incorporó una versión en su servicio de búsqueda de imágenes.

Algunos ejemplos son:

- Arkanoid DS

Permite el juego en modo historia, con 140 niveles, o modo misión, en el que el jugador puede reglas para cada pantalla.

Este juego incluye algunas funcionalidades más, además del propio Arkanoid, como son desbloquear bloques, fondos de pantalla y efectos de sonido, para personalizar el juego. Permite también visualizar las clasificaciones de los usuarios a través de la conexión Wi-Fi de Nintendo.

- Block Buster

Versión del Arkanoid para la primera versión de la Play Station. En cuanto a funcionalidad, este juego no introduce demasiadas variaciones, a excepción de algunas ventajas nuevas, y de mejoras a nivel gráfico.

3.- Recursos utilizados

3.1.- Recursos software

3.1.1.- StarUML 5.0.2.1570

StarUML es una herramienta libre para el diseño de diagramas UML (Anexo C). Aunque esta herramienta dejó de usarse durante algún tiempo, se ha vuelto a poner de moda al pasar de Delphi a Java.

El objetivo principal de esta herramienta era sustituir aplicaciones comerciales mayores y complejas, como Rational Rose y Borland Together.

Soporta la mayoría de los tipos de diagramas especificados en UML 2.0.

3.1.2.- NetBeans 7.3

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

3.1.3.- Maven 3.0.4

Maven es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Es similar en funcionalidad a Apache Ant (y en menor medida a PEAR de PHP y CPAN de Perl), pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Estuvo integrado inicialmente dentro del proyecto Jakarta pero ahora ya es un proyecto de nivel superior de la Apache Software Foundation.

Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores. Este repositorio y su sucesor reorganizado, el repositorio Maven 2, pugnan por ser el mecanismo *de facto* de distribución de aplicaciones en Java, pero su adopción ha sido muy lenta. Maven provee soporte no sólo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios. Una caché local de artefactos actúa como la primera fuente para sincronizar la salida de los proyectos a un sistema local.

Maven está construido usando una arquitectura basada en plugins que permite que utilice cualquier aplicación controlable a través de la entrada estándar. En teoría, esto podría permitir a cualquiera escribir plugins para su interfaz con herramientas como compiladores, herramientas de pruebas unitarias, etcétera, para cualquier otro lenguaje. En realidad, el soporte y uso de lenguajes distintos de Java es mínimo. Actualmente existe un plugin para .Net Framework y es mantenido, y un plugin nativo para C/C++ fue alguna vez mantenido por Maven 1.

3.1.4.- Adobe Photoshop CS5

Adobe Photoshop (popularmente conocido sólo por su segundo nombre, Photoshop) es el nombre, o marca comercial oficial, que recibe uno de los programas más famosos de la casa Adobe junto con sus hermanos Adobe Illustrator y Adobe Flash, y que se trata esencialmente de una aplicación informática en forma de taller de pintura y fotografía que trabaja sobre un "lienzo" y que está destinado a la edición, retoque fotográfico y pintura a base de imágenes de mapa de bits. Su nombre en español significa literalmente "taller de fotos". Su capacidad de retoque y modificación de fotografías le ha dado el rubro de ser el programa de edición de imágenes más famoso del mundo.

Actualmente forma parte de la familia Adobe Creative Suite y es desarrollado y comercializado por Adobe Systems Incorporated inicialmente para computadores Apple pero posteriormente también para plataformas PC con sistema operativo Windows. Su distribución viene en diferentes presentaciones, que van desde su forma individual hasta como parte de un paquete siendo estos: Adobe Creative Suite

Design Premium y Versión Standard, Adobe Creative Suite Web Premium, Adobe Creative Suite Production Studio Premium y Adobe Creative Suite Master Collection.

3.1.5.- Microsoft Word 2007

Microsoft Word es un software destinado al procesamiento de textos, creado por la empresa Microsoft. Actualmente viene integrado en la *suite* ofimática Microsoft Office.

Originalmente fue desarrollado por Richard Brodie para el computador de IBM bajo sistema operativo DOS en 1983.Versiones subsecuentes fueron programadas para muchas otras plataformas, incluyendo, las computadoras IBM que corrían en MS-DOS (1983). Es un componente de la suite ofimática Microsoft Office; también es vendido de forma independiente e incluido en la Suite de Microsoft Works. Las versiones actuales son Microsoft Office Word 2013 para Windows y Microsoft Office Word 2011 para Mac. Ha llegado a ser el procesador de texto más popular del mundo.

En sus inicios, Word tardó más de 5 años en lograr el éxito en un mercado en el que se usaba comúnmente MS-DOS, y cuando otros programas, como Corel WordPerfect, eran mucho más utilizados y populares.

La primera versión de Microsoft Word fue un desarrollo realizado por Charles Simonyi y Richard Brodie, dos ex-programadores de Xerox contratados en 1981 por Bill Gates y Paul Allen. Estos programadores habían trabajado en Xerox Bravo, que fuera el primer procesador de textos desarrollado bajo la técnica WYSIWYG ("What You See Is What You Get"); es decir el usuario podía ver anticipadamente, en pantalla, el formato final que aparecería en el impreso del documento. Esta primera versión, Word 1.0, salió al mercado en octubre de 1983 para la plataforma Xenix MS-DOS; en principio fue rudimentario y le siguieron otras cuatro versiones muy similares que no produjeron casi impacto en las ventas a usuarios finales.

La primera versión de Word para Windows salió en el año 1989, que si bien en un entorno gráfico resultó bastante más fácil de operar, tampoco permitió que las ventas se incrementaran notablemente. Cuando se lanzó al mercado Windows 3.0, en 1990, se produjo el real despegue. A Word 1.0 le sucedieron Word 2.0 en 1991, Word 6.0 en 1993. El posterior salto en los números de versión se introdujo a fin de que coincidiera con la numeración del versionado de Windows, tal como fue Word 95 y Word 97. Con la salida del Windows 2000 (1999) también surgió la versión homóloga de Word. La versión Word 2002 emergió en la misma época que el paquete Microsoft Office XP, en el año 2001. Un año después le siguió la versión Microsoft Word 2003. Posteriormente se presentó Microsoft Word 2007 junto con el resto de aplicaciones del paquete Office 2007, en esta versión, Microsoft marcó un nuevo cambio en la historia de las aplicaciones office presentando la nueva interfaz Ribbons más sencilla e intuitiva que

las anteriores (aunque muy criticada por usuarios acostumbrados a las versiones anteriores). La versión más reciente lanzada al mercado es Microsoft Word 2013, en el mismo año en el que salió el sistema Microsoft Windows 8.

3.1.6.- GlassFish Server 3.1.2

GlassFish es un servidor de aplicaciones de software libre desarrollado por Sun Microsystems, compañía adquirida por Oracle Corporation, que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. Es gratuito, de código libre y se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL. La versión comercial es denominada Oracle GlassFish Enterprise Server (antes Sun GlassFish Enterprise Server).

GlassFish está basado en el código fuente donado por Sun y Oracle Corporation; este último proporcionó el módulo de persistencia *TopLink*. GlassFish tiene como base al servidor *Sun Java System Application Server* de Oracle Corporation, un derivado de Apache Tomcat, y que usa un componente adicional llamado Grizzly que usa Java NIO para escalabilidad y velocidad.

3.1.7.- Windows 7

Windows 7 es una versión de Microsoft Windows, línea de sistemas operativos producida por Microsoft Corporation. Esta versión está diseñada para uso en PC, incluyendo equipos de escritorio en hogares y oficinas, equipos portátiles, *tablet PC*, *netbooks* y equipos *media center*. El desarrollo de Windows 7 se completó el 22 de julio de 2009, siendo entonces confirmada su fecha de venta oficial para el 22 de octubre de 2009 junto a su equivalente para servidores Windows Server 2008 R2.

A diferencia del gran salto arquitectónico y de características que sufrió su antecesor Windows Vista con respecto a Windows XP, Windows 7 fue concebido como una actualización incremental y focalizada de Vista y su núcleo NT 6.0, lo que permitió mantener cierto grado de compatibilidad con aplicaciones y hardware en los que éste ya era compatible. Sin embargo, entre las metas de desarrollo para Windows 7 se dio importancia a mejorar su interfaz para volverla más accesible al usuario e incluir nuevas características que permitieran hacer tareas de una manera más fácil y rápida, al mismo tiempo que se realizarían esfuerzos para lograr un sistema más ligero, estable y rápido.

3.1.8.- GIT

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o front end como Cogito o StGIT. Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.

El mantenimiento del software Git está actualmente supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores.

3.2.- Recursos hardware

Para la realización de este trabajo se han utilizado tres equipos, dos sobremesas con dos pantallas, y un portátil. Uno de los equipos con dos pantallas fue proporcionado por el tutor, en el laboratorio del Centro de Tecnologías de la Imagen.

4.- Planificación del trabajo

4.1.- Metodología de desarrollo

Para el desarrollo de este proyecto se ha escogido la metodología PUD. El Proceso Unificado de Desarrollo Software o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El refinamiento más conocido y documentado del Proceso Unificado es el Proceso Unificado de Rational o simplemente RUP.

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. De la misma forma, el Proceso Unificado de Rational, también es un marco de trabajo extensible, por lo que muchas veces resulta imposible decir si un refinamiento particular del proceso ha sido derivado del Proceso Unificado o del RUP. Por dicho motivo, los dos nombres suelen utilizarse para referirse a un mismo concepto.

El nombre Proceso Unificado se usa para describir el proceso genérico que incluye aquellos elementos que son comunes a la mayoría de los refinamientos existentes. También permite evitar problemas legales ya que Proceso Unificado de Rational o RUP son marcas registradas por IBM(desde su compra de Rational Software Corporation en 2003). El primer libro sobre el tema se denominó, en su versión española, El Proceso Unificado de Desarrollo de Software (ISBN 84-7829-036-2) y fue publicado en 1999 por Ivar Jacobson, Grady Booch y James Rumbaugh, conocidos también por ser los desarrolladores del UML, el Lenguaje Unificado de Modelado. Desde entonces los autores que publican libros sobre el tema y que no están afiliados a Rational utilizan el término Proceso Unificado, mientras que los autores que pertenecen a Rational favorecen el nombre de Proceso Unificado de Rational.

Las características principales del Proceso Unificado de Desarrollo son:

- Iterativo e Incremental

El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio sólo consta de varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo

en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

- Dirigido por los casos de uso

En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc. el proceso dirigido por casos de uso es el rup. Nota: en UP se está Dirigido por requisitos y riesgos de acuerdo con el Libro UML 2 de ARLOW, Jim que menciona el tema.

- Centrado en la arquitectura

El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc.

- Enfocado en los riesgos

El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

4.1.1.- Fases del Proceso Unificado de Desarrollo

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Al final de cada uno de ellos se obtiene una versión final del producto, que no sólo satisface ciertos casos de uso, sino que está lista para ser entregada y puesta en producción. En caso de que fuese necesario publicar otra versión, deberían repetirse los mismos pasos a lo largo de otro ciclo.

Cada ciclo se compone de varias fases, y dentro de cada una de ellas, los directores o desarrolladores pueden descomponer adicionalmente el trabajo en iteraciones, con sus incrementos resultantes. Cada fase termina con un hito, determinado por la disponibilidad de un conjunto de artefactos, modelos o documentos.

Las iteraciones de cada fase se desarrollan a través de las actividades de identificación de requisitos, análisis, diseño, implementación, pruebas e integración.

4.1.1.1.- Fase de inicio

Durante la fase de inicio se desarrolla una descripción del producto final, y se presenta el análisis del negocio. Esta fase responde a las siguientes preguntas:

- ¿Cuáles son las principales funciones del sistema para los usuarios más importantes?

- ¿Cómo podría ser la mejor arquitectura del sistema?

- ¿Cuál es el plan del proyecto y cuánto costará desarrollar el producto?

En esta fase se identifican y priorizan los riesgos más importantes. El objetivo de esta fase es ayudar al equipo de proyecto a decidir cuáles son los verdaderos objetivos del proyecto. Las iteraciones exploran diferentes soluciones posibles, y diferentes arquitecturas posibles. Puede que todo el trabajo físico realizado en esta fase sea descartado. Lo único que normalmente sobrevive a la fase de inicio es el incremento del conocimiento del equipo.

Los artefactos que típicamente sobreviven a esta fase son:

- Un enunciado de los mayores requerimientos planteados generalmente como casos de uso.

- Un boceto inicial de la arquitectura.

- Una descripción de los objetivos del proyecto.

- Una versión muy preliminar del plan de proyecto.

- Un modelo del negocio.

La fase de inicio finaliza con el Hito de Objetivos del Ciclo de Vida. Este hito es alcanzado cuando el equipo de proyectos y las partes interesadas llegan a un acuerdo sobre:

- Cuál es el conjunto de necesidades del negocio, y qué conjunto de funciones satisfacen éstas necesidades.

- Una planificación preliminar de las iteraciones.

- Una arquitectura preliminar.

Debe poder responderse a las siguientes cuestiones:

- ¿Se ha determinado con claridad el ámbito del sistema? ¿Se ha determinado lo que va a estar dentro del sistema y fuera del sistema?

- ¿Se ha llegado a un acuerdo con todas las personas involucradas (las partes interesadas) sobre los requisitos funcionales del sistema?

- ¿Se vislumbra una arquitectura que pueda soportar éstas características?

- ¿Se identifican los riesgos críticos? ¿Se prevé una forma de mitigarlos?

- ¿El uso del producto justifica la relación costo-beneficio?

- ¿Es factible para la empresa llevar adelante el proyecto?

- ¿Están los inversores de acuerdo con los objetivos?

4.1.1.2.- Fase de elaboración

Durante la fase de elaboración se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura.

Las iteraciones de la fase de elaboración:

- Establecen una firme comprensión del problema a solucionar.

- Establece la fundación arquitectural para el software.

- Establece un plan detallado para las siguientes iteraciones.

- Elimina los mayores riesgos.

El resultado de esta fase es la línea base de la arquitectura. En esta fase se construyen típicamente los siguientes artefactos:

- El cuerpo básico del software en la forma de un prototipo arquitectural.

- Casos de prueba.

- La mayoría de los casos de uso (80%) que describen la funcionalidad del sistema.

- Un plan detallado para las siguientes iteraciones.

La fase de elaboración finaliza con el hito de la Arquitectura del Ciclo de Vida. Este hito se alcanza cuando el equipo de desarrollo y las partes interesadas llegan a un acuerdo sobre:

- Los casos de uso que describen la funcionalidad del sistema.

- La línea base de la arquitectura.

- Los mayores riesgos han sido mitigados.

- El plan del proyecto.

Al alcanzar este hito debe poder responderse a preguntas como:

- ¿Se ha creado una línea base de la arquitectura? ¿Es adaptable y robusta? ¿Puede evolucionar?

- ¿Se han identificado y mitigado los riesgos más graves?

- ¿Se ha desarrollado un plan de proyecto hasta el nivel necesario para respaldar una agenda, costes y calidad realistas?

- ¿Proporciona el proyecto una adecuada recuperación de la inversión?

- ¿Se ha obtenido la aprobación de los inversores?

4.1.1.3.- Fase de construcción

Durante la fase de construcción se crea el producto. La línea base de la arquitectura crece hasta convertirse en el sistema completo. Al final de esta fase el producto contiene todos los casos de uso implementados, sin embargo puede que no esté libre de defectos. Los artefactos producidos en esta fase son:

- El sistema software.

- Los casos de prueba.

- Los manuales de usuario.

La fase de construcción finaliza con el hito de Capacidad Operativa Inicial. Este hito se alcanza cuando el equipo de desarrollo y las partes interesadas llegan a un acuerdo sobre:

- El producto estable para ser usado.

- El producto provee alguna funcionalidad de valor.

- Todas las partes están listas para comenzar la transición.

4.1.1.4.- Fase de transición

La fase de transición cubre el período durante el cual el producto se convierte en la versión beta.

Las iteraciones en esta fase continúan agregando características al software. Sin embargo, las características se agregan a un sistema que el usuario se encuentra utilizando activamente.

Los artefactos construidos en esta fase son los mismos que en la fase de construcción. El equipo se encuentra ocupado fundamentalmente en corregir y extender la funcionalidad del sistema desarrollado en la fase anterior.

La fase de transición finaliza con el hito de Lanzamiento del Producto. Este hito se alcanza cuando el equipo de desarrollo y las partes interesadas llegan a un acuerdo sobre:

- Se han alcanzado los objetivos fijados en la fase de inicio.

- El usuario está satisfecho.

4.2.- Planificación y temporización

La planificación de este trabajo se ha dividido en siete actividades, a las cuales se les ha asignado proporcionalmente las trescientas horas que se deben dedicar al Trabajo de Fin de Grado.

Las actividades son:

- Elicitación de requisitos de usuario (15 horas).

- Análisis de los requisitos de usuario (15 horas).
- Elicitación de requisitos de software (20 horas).
- Diseño de la arquitectura de la aplicación (50 horas).
- Documentación de las fases de elicitación y diseño (20 horas).
- Implementación de la aplicación (170 horas).
- Pruebas (10 horas).

4.3.- Presupuesto

El presupuesto de desarrollo de éste producto viene dado por los gastos relacionados con el personal que se necesita para el desarrollo del mismo y su posterior mantenimiento, y además, los gastos relacionados con los productos software y hardware necesarios para la implementación del mismo.

Para calcular el coste de desarrollo se debe tener en cuenta el número de horas de trabajo efectivo de un trabajador, que ronda las 1575 horas laborables al año (descontando los días de vacaciones no laborables). Además, se debe diferenciar entre el salario bruto y lo que realmente le cuesta a la empresa el trabajador, ya que normalmente, un trabajador cuesta a la empresa alrededor de 1.5 su nómina.

Se toma como referencia que el alumno tiene un sueldo de 1100 euros al mes, con dos pagas extras, y que el tutor cobra 2100 euros. Por lo tanto:

- Alumno: 14 pagas * 1100 euros * 1.5 = 23100 €/año

- Profesor: 14 pagas * 2100 euros * 1.5 = 44100 €/año

Tomando por consideración que cada uno de ellos trabaja 1575 horas al año, el coste hora de cada uno es:

- Alumno: 14,67 €/h

- Profesor: 28 €/h

En cuanto a las horas de trabajo de cada uno, el alumno tendrá dedicación completa, por lo que trabajará las 300 horas que indica la planificación. Por otro lado, el tutor se reunirá con el alumno en algunos momentos, y prestará ayuda cuando éste lo necesite, por lo que se estima que dedicará un 30% del tiempo total del proyecto, por lo que resultan unas 90 horas.

Finalmente, los costes se calculan multiplicando el coste por hora de cada trabajador por el número de horas estimadas en la planificación.

- Total alumno: 300 h * 14,67 €/h = 4401 €

- Total tutor: 90 h * 28 €/h = 2520 €

El coste total de personal resulta 6921 €.

Para el coste de mantenimiento se tendrá en cuenta un período de mantenimiento de cinco años, y se estima que se dedicarán cuatro horas diarias. Teniendo en cuenta estos datos, durante los cinco años el alumno dedicará unas 3938 horas.

Atendiendo a los costes por hora calculados en el apartado anterior, los costes ascienden a:

- Total alumno: 3938 h * 14,67 €/h = 57770,46 €.

En cuanto a los costes de material, se tienen en cuenta los equipos utilizados. En primer lugar, el ordenador portátil ha tenido un coste de 800€. Por otro lado, el equipo con dos pantallas ha tenido un coste de 1300€.

Además, hay que incluir las licencias de los productos software que se utilizarán en el desarrollo:

- Windows 7 Home Edition. El coste aproximado de este producto es de 92,67 €.

- Adobe Photoshop CS6. El coste aproximado de este producto es de 286,59 €.

- Microsoft Word 2007. El paquete básico, que contiene los productos que se necesitan para el desarrollo de la memoria, cuesta aproximadamente unos 82,49 €.

En total, los costos de material ascenderían a 2561,75€.

Los costes totales de la ejecución de este proyecto ascenderían a 67253.21€.

5.- Desarrollo del trabajo

En este capítulo se ilustrarán varios de los artefactos que se han realizado hasta llegar a obtener el producto final, desde la recopilación de requisitos hasta la etapa de desarrollo y pruebas del sistema. La primera parte del desarrollo del proyecto consistirá en identificar los requisitos del sistema y los requisitos del software. Este estudio servirá como base para definir y delimitar el ámbito del proyecto, y para utilizarlo como guía para el desarrollo. Las herramientas que se utilizarán serán el modelo del dominio, la lista de características y los casos de uso. La segunda parte permite definir cualquier característica que fuese deseable incluir en la aplicación y la última perfila de forma algo más concreta las funciones que se esperan del mismo.

5.1.- Requisitos del sistema

En este apartado se describirán dos artefactos muy importantes a la hora de establecer el ámbito en el que se moverá la aplicación: el modelo del dominio y la lista de características. También ayudan a los desarrolladores a comprender el contexto del sistema, además de poder recopilar requisitos funcionales y no funcionales.

5.1.1.- Modelo del dominio

5.1.1.1.- Introducción

El modelo del dominio muestra clases conceptuales significativas en el dominio de un problema, es decir, es una representación de las clases conceptuales del mundo real, no de componentes software.

Muchos de los objetos del dominio o clases pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio. Las clases del dominio aparecen en tres formas típicas:

- Objetos del negocio que representan cosas que se manipulan en el negocio, como pedidos, cuentas y contratos.

- Objetos del mundo real y conceptos de los que el sistema de hacer un seguimiento, como la aviación enemiga, misiles y trayectorias.

- Sucesos que ocurrirán o han transcurrido, como la llegada de un avión, su salida y la hora de la comida.

El modelo del dominio se describe mediante diagramas UML. Estos diagramas muestran a los clientes, usuarios, revisiones y a otros desarrolladores las clases del dominio y cómo se relacionan unas cosas con otras mediante asociaciones.

5.1.1.2.- Desarrollo de un modelo del dominio

El modelado del dominio se realiza habitualmente en reuniones organizadas por los analistas del dominio, que utilizan UML y otros lenguajes de modelado para

documentar los resultados. Para formar un equipo eficaz, estas reuniones deberían incluir tanto a expertos del dominio como a gente con experiencia en modelado.

El objetivo del modelado del dominio es comprender y describir las clases más importantes dentro del contexto del sistema. Los dominios de tamaño moderado normalmente requieren entre 10 y 50 clases.

Los restantes cientos de clases candidatas que los analistas pueden extraer del dominio se guardan como definiciones en un glosario de términos, ya que, de otra manera, el modelo del dominio se haría demasiado grande y requeriría más esfuerzo del necesario para esta parte del proceso.

Algunas veces, como en los dominios de negocio muy pequeños, no es necesario desarrollar un modelo de dominio, en su lugar puede ser suficiente un glosario de términos.

El glosario y el modelo del dominio ayudan a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común. La terminología común es necesaria para compartir el conocimiento de los otros. Cuando abunda la confusión, el proceso de ingeniería se hace difícil, si no imposible. Para construir un sistema software de cualquier tamaño, los ingenieros de hoy en día deben fundir el lenguaje de todos los participantes en uno solo consistente.

5.1.1.3.- Uso del modelo de dominio

Las clases del dominio y el glosario de términos se utilizan en el desarrollo de los modelos de casos de uso y de análisis. Se utilizan:

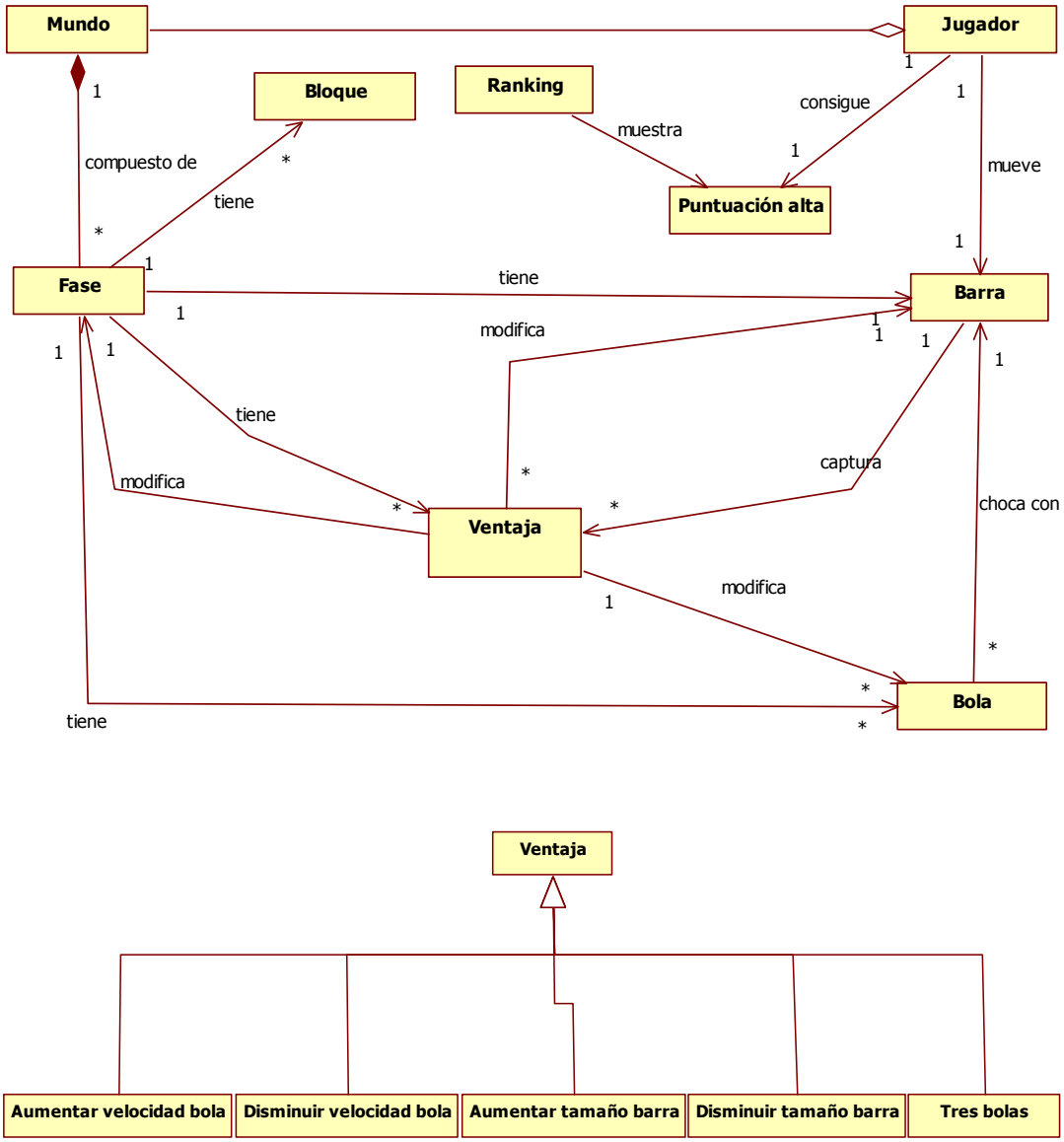
- Al describir los casos de uso y al diseñar la interfaz de usuario.

- Para sugerir clases internas al sistema en desarrollo durante el análisis.

5.1.1.4.- Diagramas

La identificación de los elementos o clases que componen los diversos diagramas del modelo de dominio se ha conseguido mediante varias iteraciones al proceso de recoger aquellos objetos y aspectos pertenecientes al mundo real asociado al contexto de la aplicación.

A continuación se presenta el modelo del dominio para el contexto en el que se engloba la aplicación.



5.1.2.- Lista de características

La lista de características sirve para contener las ideas de los clientes, usuarios, analistas y desarrolladores a modo de fichas sobre los posibles aspectos que se podrían incluir en la aplicación, y que, posteriormente, se podrán desarrollar en la versión actual del sistema o se podrán postergar a versiones futuras. Este artefacto sirve para gestionar el proyecto y sólo se utiliza para la planificación del trabajo. Podrá ir variando a medida que avance el proyecto, pudiéndose añadir y modificar las características que se crean oportunas, en cualquier momento del desarrollo.

A continuación se muestra la lista de características y funcionalidades que podría tener la aplicación. Cada característica tiene un atributo prioridad que especifica el orden que se seguiría para la implementación. Además, las características se han agrupado en base a los ámbitos a los que pertenece cada una.

ID	Nombre	Descripción	Prioridad
LC-A-1	**Jugar Arkanoid**	Permitir la carga de diferentes mundo y fases, así como de las bolas y la barra, y de la lógica de juego del arkanoid	1
LC-A-17	**Resumir partida**	Cuando el juego está pausado, el usuario vuelve a jugar en el estado en que quedó la partida cuando se pausó	1
LC-A-2	**Multijugador**	Varios jugadores puede conectarse al sistema para jugar juntos, ya sea en modo batalla, o cooperativo. En principio pueden ser 2 o 4 juadores, y cada uno tendrá una barra, colocada en uno de los lados del tablero, si son 2, una en la parte superior y otra en la parte inferior, y si son 4, además de las dos anteriores, una en el lado derecho, y otra en el lado izquierdo	1
LC-A-4	**Seleccionar fase**	Cuando ya se ha seleccionado un mundo, se puede seleccionar una fase de entre las desbloqueadas para ese usuario en el mundo seleccionado	1
LC-A-5	**Pausar**	Mientras el usuario está jugando, puede detener el juego, sin salir de éste	1
LC-A-6	**Guardar partida**	Cuando la partida está pausada, el usuario puede guardar el estado, y recuperarlo en otro momento	1
LC-A-7	**Continuar partida**	Cuando par partida está pausada, el usuario puede continuar la partida como ésta estaba cuando se pausó	1
LC-A-8	**Seleccionar mundo**	El usuario puede seleccionar uno de los mundos desbloqueados para él.	1
LC-B-1	**Registrar usuario**	Introducir los datos de un usuario en el sistema	1

LC-B-2	**Logear Usuario**	Cuando ya un usuario está registrado, se identifica ante el sistema	1
LC-B-3	**Logout**	El usuario cierra su sesión de cara al sistema	1
LC-A-14	**Bloquear mundo**	El sistema mantendrá los mundos bloqueados hasta que el usuario los compre, y además, hasta que no acabe todas las fases del mundo anterior, no se desbloqueará el siguiente	2
LC-A-15	**Desbloquear mundo**	El sistema desbloquerá los mundos que el usuario compre, y además, si ha acabado todas las fases del mundo anterior	2
LC-A-16	**Bloquear fase**	Mientras el usuario no termine la fase anterior, el sistema mantiene bloqueadas las fases consecutivas	2
LC-A-16	**Desbloquear fase**	Cuando el usuario termina una fase, el sistema desbloquea la siguiente	2
LC-A-3	**Ventajas**	Cuando se toca una caja con la bola, existe la posibilidad de que la caja contenga una ventaja, que caerá hacia el usuario que haya enviado la bola contra dicha caja. Esta ventaja podrá tener efectos negativos o positivos sobre cualquiera de los elementos de juego	2
LC-F-1	**Modificar configuración**	Modificar las opciones de sonido, resolución, configuración de teclado	2
LC-G-1	**Comprar mundo**	El usuario compra el mundo y se desbloquea la primera fase de dicho mundo, y las demás se irán desbloqueando a medida que el usuario termine las fases desbloqueadas anterioes	2
LC-A-13	**Añadir trampa**	El usuario compra códigos que después introduce en el sistema para tener ventajas	3

		adicionales, tales como vidas infinitas, armas durante toda la fase, etc	
LC-C-1	**Crear fase**	¿Un usuario crea un mundo, y si gusta se puede poner a la venta, y las ganancias se reparten entre ambos?	3
LC-E-4	**Enviar mensaje**	Los usuarios pueden enviar mensajes entre sí	3
LC-E-8	**Compartir video/record**	En youtube por ejemplo	3
LC-H-2	**Replays de partida**	Los usuarios tienen un historial de partidas, que pueden volver a ver en forma de video	3
LC-A-10	**Modos de juego**	Diferentes modos de juego. Historia, cooperativo, batalla, cooperativo por turnos, en el mismo o en equipos distintos	4
LC-A-11	**Tipos de partida**	privada o pública	4
LC-D-1	**Ver records**	Se pueden consultar las mayores puntuaciones y los tiempos record de cada fase/mundo	4
LC-D-2	**Ver mis puntuaciones**	El usuario puede consultar las puntuaciones obtenidas en cada partida	4
LC-E-2	**Invitar amigo**	El usuario puede invitar a jugar una partida multijugador a un amigo	4
LC-E-5	**Añadir amigo**	El usuario tiene una lista de amigos, y puede invitar a otros usuarios a formar parte de dicha lista	4
LC-E-6	**Invitar a partida privada**	Invitar a un amigo a ver/jugar una partida a la que sólo tienen acceso los usuarios que el usuario que la crea permita	4
LC-H-1	**Observar partida**	Cuando alguien está jugando, otros jugadores pueden observar la partida, si	4

		tienen acceso a ella	
LC-A-12	**Revancha**	Lista de jugadores con los que se ha jugado para jugar de nuevo	5
LC-A-9	**Búsqueda contrincante**	Cuando el usuario quiere jugar, busca otros usuarios con los que jugar una partida multijugador	5
LC-E-1	**Perfiles de usuario**	Información del usuario que los demás usuario puedan consultar	5
LC-E-3	**Comentar partida pública**	Los usuarios podrán hacer comentarios sobre el replay de cada partida	5
LC-E-7	**Chat de voz/video**	Mientras los usuarios están jugando, pueden hablar y verse	5
LC-E-9	**Bloquear/desbloquear mensajes**	Los usuarios pueden bloquear los mensajes entrantes de los usuarios de los que no deseen recibir mensajes	5
LC-F-2	**Activar sonido**	El sistema emite sonidos	5
LC-F-3	**Desactivar sonido**	El sistema no emite sonidos	5
LC-F-4	**Personalizar barra**	El usuario puede cambiar la apariencia de la barra en cuanto al color, texturas, etc	5

5.2.- Requisitos del software

5.2.1.- Actores

Los actores del sistema serán:

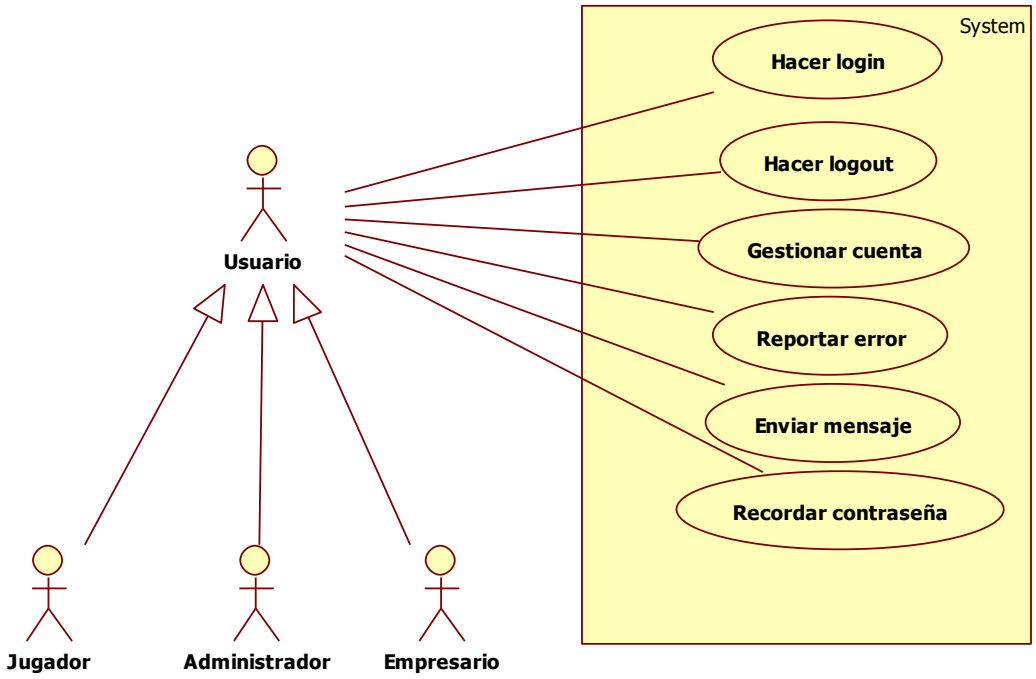
- Jugador: se entiende por jugador aquel usuario que registrado o no, entra en el sistema para jugar.

- Empresario: se entiende por empresario aquel usuario del sistema que entra en el sistema para ejecutar funcionalidades relacionadas con la consulta y gestión de los datos.

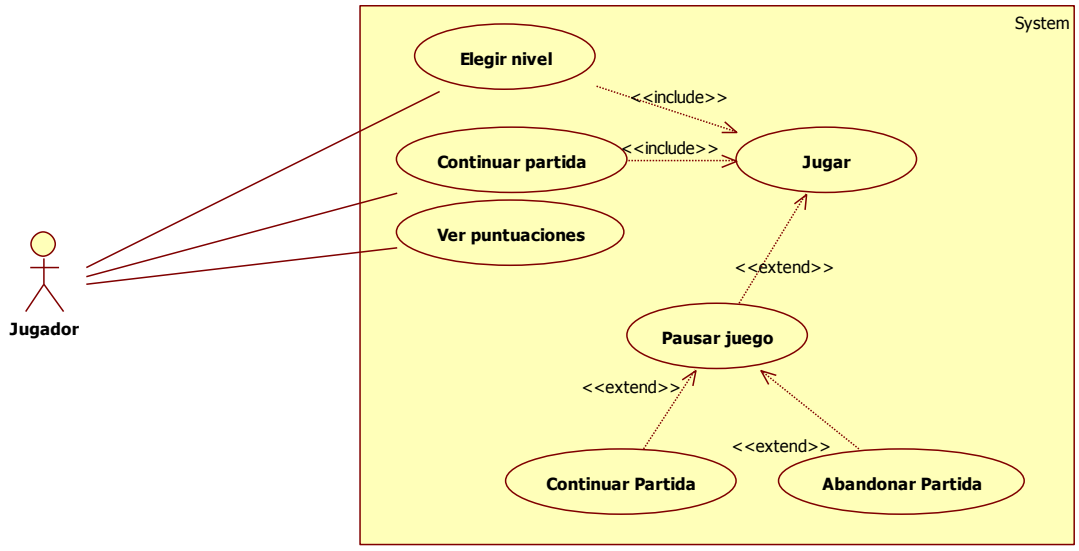
- Administrador/es del sistema: se entiende como administrador aquel usuario que entra en el sistema para configurar y administrar el sistema.

5.2.2.- Modelo de casos de uso

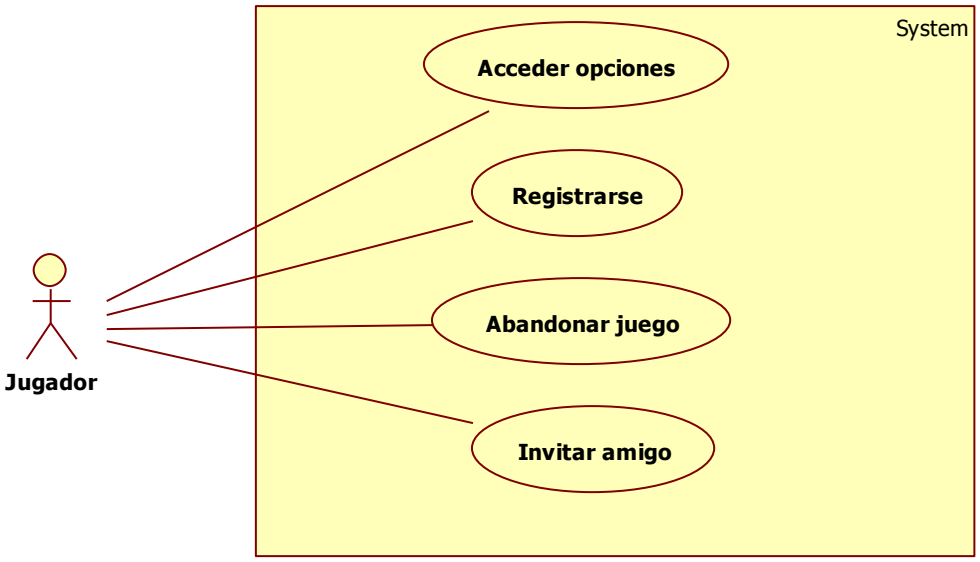
La aplicación tendrá varios tipos de usuario, pero todos ellos tendrán algunos casos de uso comunes, abstraídos a un usuario abstracto, del que heredarán los demás:



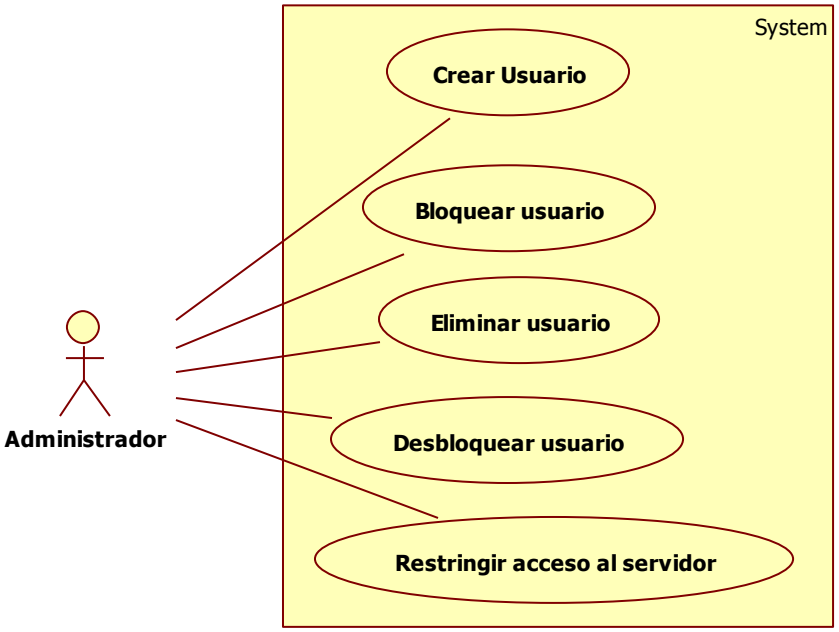
En siguiente lugar se muestran los casos de uso del actor Jugador relacionados con el juego y la partida:



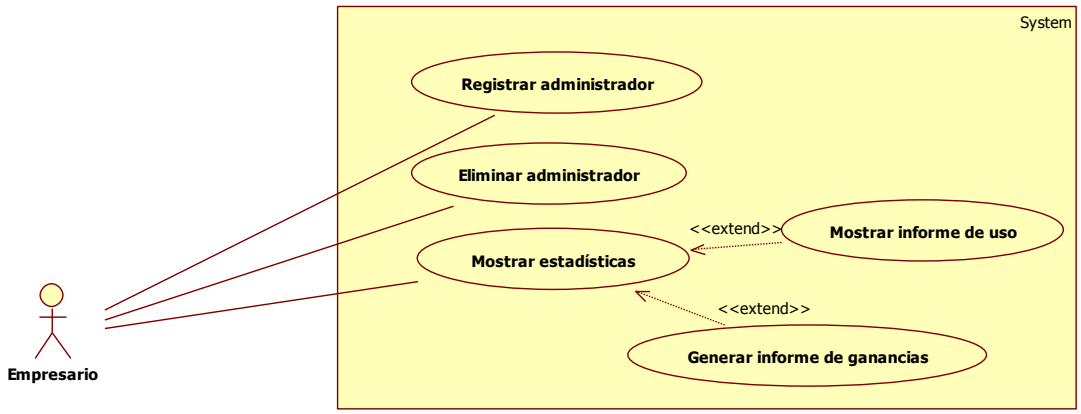
A continuación se muestran los casos de uso del actor Jugador relacionados con el registro de usuario:



En siguiente lugar se muestran los casos de uso del actor Administrador:



En último lugar, se muestran los casos de uso del actor Empresario:



De todos los casos de uso anteriores, se seleccionan como más prioritarios los relacionados con el juego y con el registro del usuario, que son:

- Elegir nivel.
- Continuar partida.
- Jugar.
- Registrarse.
- Hacer log in.

5.2.3.- Especificación de casos de uso

5.2.3.1.- C.U. Seleccionar Mundo

Actores	Jugador
Stakeholders	-
Precondiciones	-
Postcondiciones	Mundo seleccionado
Camino normal	**Camino alternativo**
1.- El sistema descarga una lista de mundos y la muestra al usuario.	1.1.- Si existen problemas de conexión al servidor 1.1.a.- Mensaje de error de conexión. 1.1.b.- Volver a la página principal.
2.- El usuario elige un mundo que esté desbloqueado.	2.1.- Si el usuario elige un mundo que está bloqueado.

	2.1.a.- Mensaje de mundo bloqueado y se ejecuta el caso de uso comprar mundo.
3.- El sistema ejecuta el caso de uso ‘Seleccionar Fase’.	
4.- Fin.	
Excepciones	

5.2.3.2.- C.U. Jugar

Actores	Jugador
Stakeholders	-
Precondiciones	Fase seleccionada
Postcondiciones	-
Camino normal	**Camino alternativo**
1.- El sistema descarga la siguiente fase y muestra la pantalla de juego.	
2.- El usuario juega.	2.1.- El usuario decide pausar. 2.1.a.- Se ejecuta el caso de uso ‘Pausar’. 2.2.- El usuario pierde 2.2.a.- Fin
3.- Si quedan fases, saltar a 1.	
4.- El sistema muestra la pantalla ‘Congratulations’.	
5.- Fin.	
Excepciones	

5.2.3.3.- C.U. Registrarse

Actores	Jugador
Stakeholders	-
Precondiciones	-
Postcondiciones	Usuario registrado en el sistema
Camino normal	**Camino alternativo**
1.- El sistema muestra un formulario de registro.	
2.- El usuario cumplimenta el formulario de registro, y selecciona “Ok”.	
3.- El usuario es añadido al sistema.	3.1.- El nombre de usuario o correo ya existe. 3.1.a.- Volver a punto 1
4.- Fin	
Excepciones	

5.2.3.4.- C.U. Invitar Amigo

Actores	Jugador
Stakeholders	Empresario
Precondiciones	El usuario deberá estar logueado en el sistema.
Postcondiciones	
Camino normal	**Camino alternativo**
1.- El usuario añade 1 o más correos de sus amigos en un formulario	
2.- El formulario es enviado	

3.- Fin.	
Excepciones	

5.2.3.5.- C.U. Seleccionar Fase

Actores	Jugador
Stakeholders	
Precondiciones	Mundo seleccionado
Postcondiciones	Fase seleccionada
Camino normal	**Camino alternativo**
1.- El sistema descarga una lista de fases y las muestra al usuario.	
2.- El usuario elige una fase que está desbloqueada.	2.- El usuario elige una fase que está bloqueada pero no ocurre nada.
3.- Se ejecuta el caso de uso ‘Jugar’.	
4.- Fin.	
Excepciones	

5.2.3.6.- C.U. Acceder Opciones

Actores	Jugador
Stakeholders	
Precondiciones	
Postcondiciones	
Camino normal	**Camino alternativo**
1.- El usuario pulsa la opción “Acceder Opciones”	

2.- El sistema muestra las opciones del sistema.	
3.- Fin.	
Excepciones	

5.2.3.7.- C.U. Abandonar Juego

Actores	Jugador
Stakeholders	-
Precondiciones	-
Postcondiciones	-
Camino normal	**Camino alternativo**
1.- El sistema detiene el juego y muestra el menú principal.	
2.- Fin.	
Excepciones	

5.2.3.8.- C.U. Ver Puntuaciones

Actores	Jugador
Stakeholders	
Precondiciones	
Postcondiciones	
Camino normal	**Camino alternativo**
1.- El sistema enseña una lista de usuarios con las mejores puntuaciones.	1.a.- Si el usuario está logueado se enseña una lista con sus mejores puntuaciones también.

2.- Fin	
Excepciones	

5.2.3.9.- C.U. Continuar Partida

Actores	Jugador
Stakeholders	-
Precondiciones	-
Postcondiciones	-
Camino normal	**Camino alternativo**
1.- El sistema pide al servidor el estado de la última partida de este usuario.	
2.- El sistema ejecuta al caso de uso Jugar.	
3.- Fin.	
Excepciones	

5.2.3.10.- C.U. Pausar Juego

Actores	Jugador
Stakeholders	-
Precondiciones	Caso de uso “Jugar” iniciado
Postcondiciones	-
Camino normal	**Camino alternativo**
1.- El sistema pausa el juego.	
Excepciones	

5.2.3.11.- C.U. Crear Usuario

Actores	Administrador
Stakeholders	-
Precondiciones	-
Postcondiciones	-
Camino normal	**Camino alternativo**
1.- El administrador cumplimenta un formulario de registro.	
2.- El administrador dota de un Nick y una contraseña a un nuevo usuario.	
3.- Fin.	
Excepciones – El nombre de usuario ya existe y no se puede crear	

5.2.3.12.- C.U. Bloquear Usuario

Actores	Administrador
Stakeholders	-
Precondiciones	Usuario no bloqueado
Postcondiciones	Usuario bloqueado
Camino normal	**Camino alternativo**
1.- El administrador deshabilita una cuenta de usuario para su uso.	
2.- Fin.	
Excepciones	

5.2.3.13.- C.U. Eliminar usuario

Actores	Administrador

Stakeholders	-
Precondiciones	-
Postcondiciones	-
Camino normal	**Camino alternativo**
1.- El administrador borra un usuario del sistema.	
2.- Sus puntuaciones se borran.	
3.- El estado de su partida se borra.	
4.- Fin.	
Excepciones	

5.2.3.14.- C.U. Desbloquear Usuario

Actores	Administrador
Stakeholders	
Precondiciones	Usuario bloqueado
Postcondiciones	Usuario no bloqueado
Camino normal	**Camino alternativo**
1.- El administrador habilita una cuenta de usuario para su uso.	
2.- Fin.	
Excepciones	

5.2.3.15.- C.U. Registrar Administrador

Actores	Empresario

Stakeholders	
Precondiciones	
Postcondiciones	
Camino normal	**Camino alternativo**
1.- El empresario habilita una cuenta de administrador para su uso.	
2.- Fin.	
Excepciones	

5.2.3.16.- C.U. Eliminar Administrador

Actores	Empresario
Stakeholders	
Precondiciones	
Postcondiciones	
Camino normal	**Camino alternativo**
1.- El empresario elimina una cuenta de administrador ya creada.	
2.- Fin.	
Excepciones	

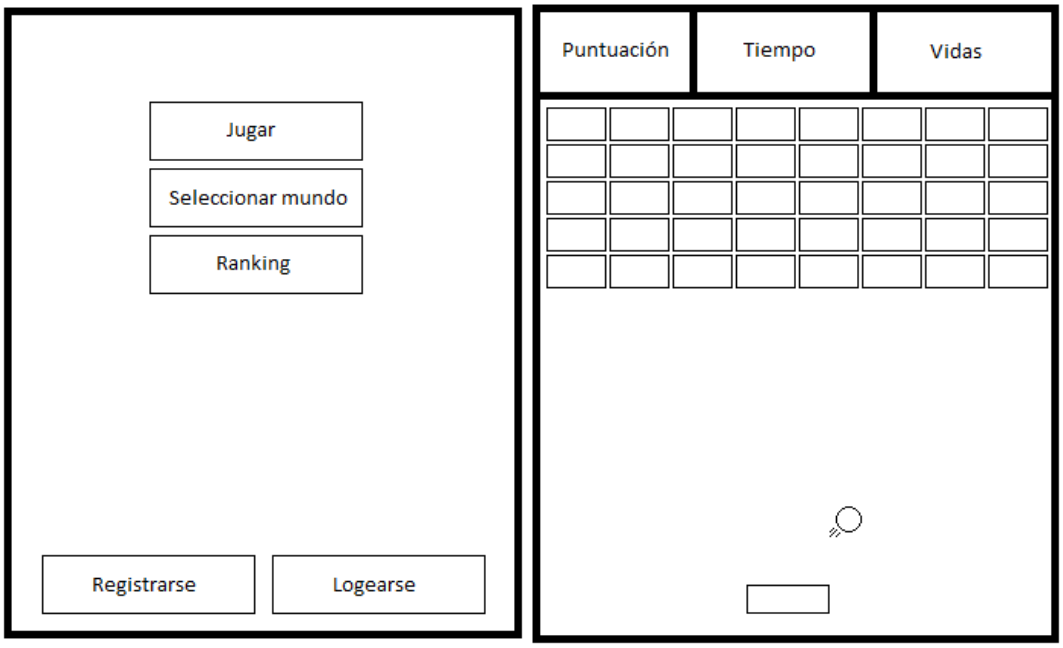
5.2.3.17.- C.U. Logearse

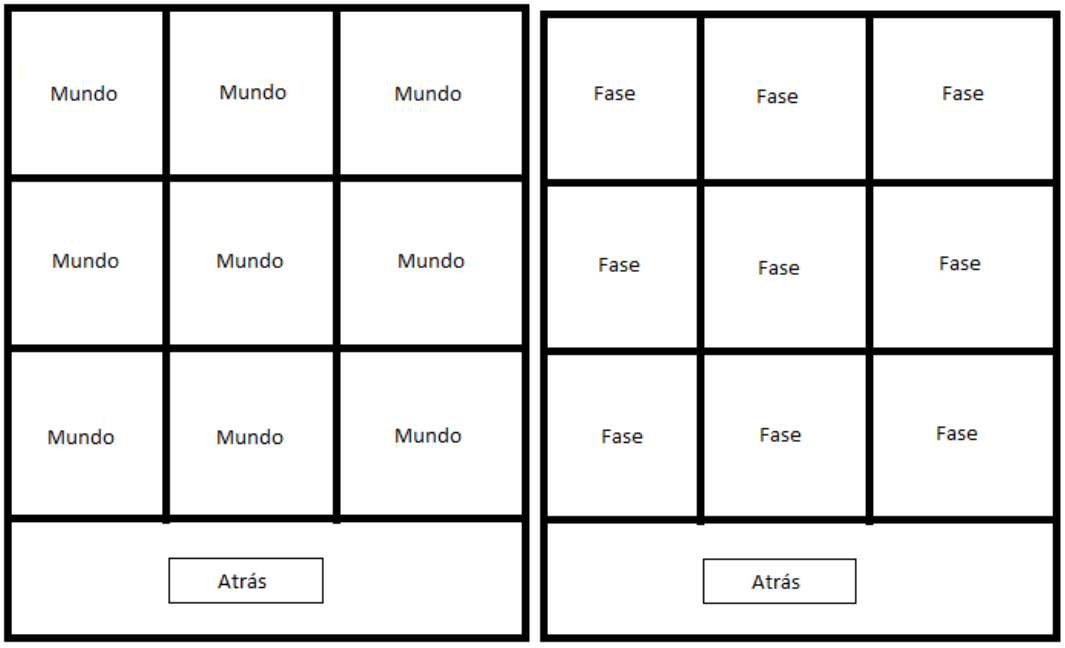
Actores	Usuario
Stakeholders	-
Precondiciones	-

Postcondiciones	-
Camino normal	**Camino alternativo**
1.- El usuario hace click en el botón Log In.	
2.- El sistema muestra un formulario en el que se solicita nombre de usuario y contraseña.	
3.- El usuario rellena el formulario y hace click en “Log in”.	3.1.- El usuario hace click en “Cancel” y el sistema vuelve al menú principal.
4.- El sistema muestra la pantalla principal y el usuario ya está logeado.	4.- Si la contraseña o el usuario no existen se vuelve al punto 2.
5.- Fin.	
Excepciones	

5.2.4.- Prototipo de interfaz de usuario

Las siguientes imágenes muestran cómo serán los diferentes menús de la aplicación.





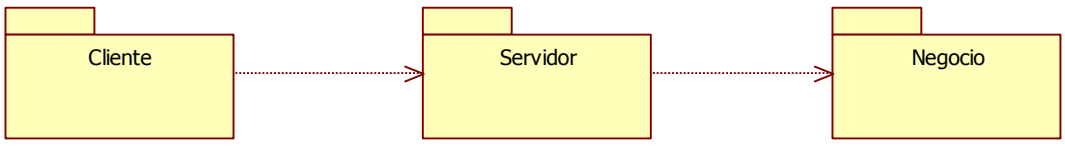
En la primera imagen se observa el menú principal del juego, desde el que se puede jugar, seleccionar mundo, ver el ranking, registrarse o hacer log in. En la segunda imagen se muestra cómo sería la ventana de juego, con los bloques, la barra, la bola, las puntuaciones, el tiempo y las vidas. En la tercera y cuarta imagen se muestra cómo serían los menús de selección de mundo y fase respectivamente.

5.3.- Modelo de análisis

El análisis se desarrollará sobre las tablas de flujo de sucesos descritas durante los requisitos. Primero se presentará una visión general de la organización de la aplicación, con los diferentes paquetes identificados. A continuación se realizarán los diagramas de clase para identificar los elementos estructurales que intervienen en el caso de uso especificado. Posteriormente se obtendrá el diagrama de colaboración, en el que queda reflejado cómo interactúan estas entidades, tanto en la forma de comunicarse como en la temporización de la interacción.

5.3.1.- Organización del modelo de análisis

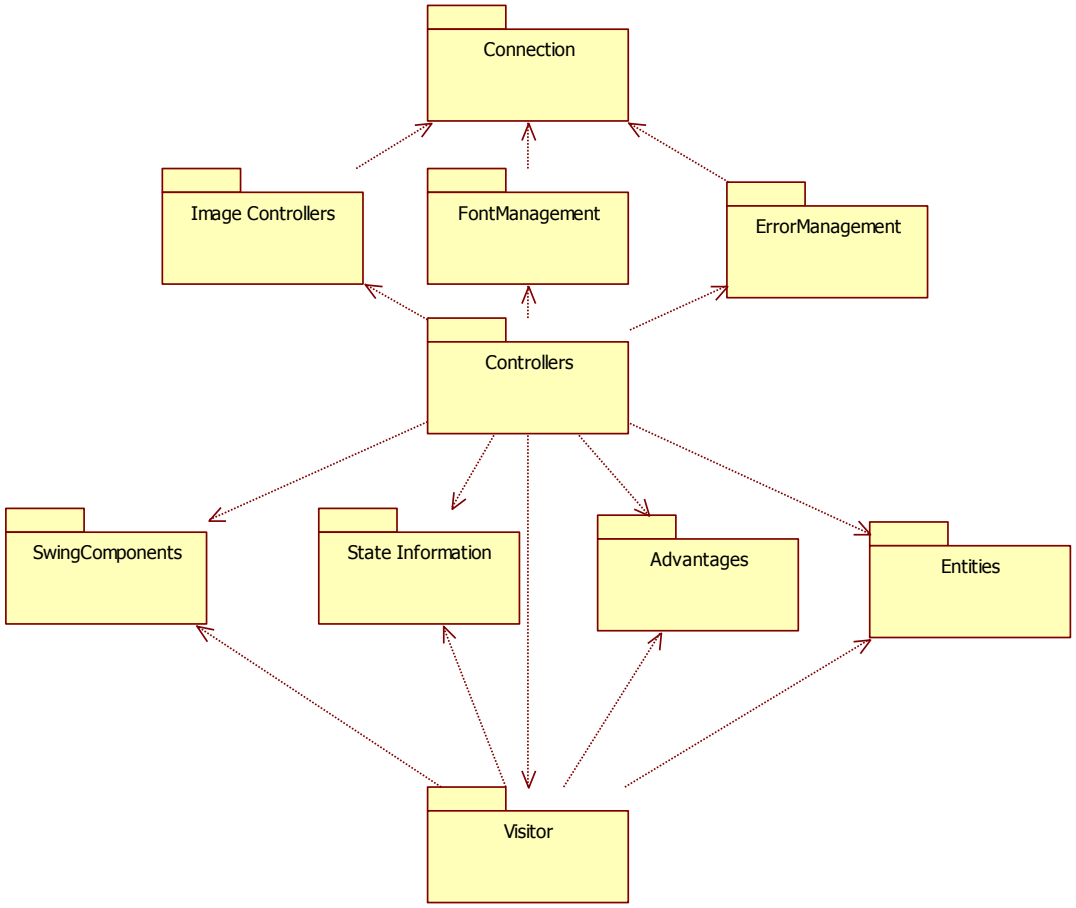
La organización de los paquetes del sistema y las dependencias entre ellos se muestra a continuación:



Cada uno de los paquetes que aparecen en la imagen anterior corresponde a una de las capas de la aplicación, y que en última instancia corresponden a una aplicación distinta para cada capa, pudiendo ejecutarse cada una en equipos distintos.

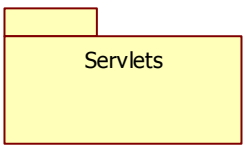
5.3.1.1.- Paquete cliente

En este paquete se encuentran los paquetes que contienen las clases que implementan toda la lógica del juego, además de los paquetes que proporcionan la conexión a la capa de servidor, para permitir la descarga de las fases, el registro y la identificación de los usuarios.



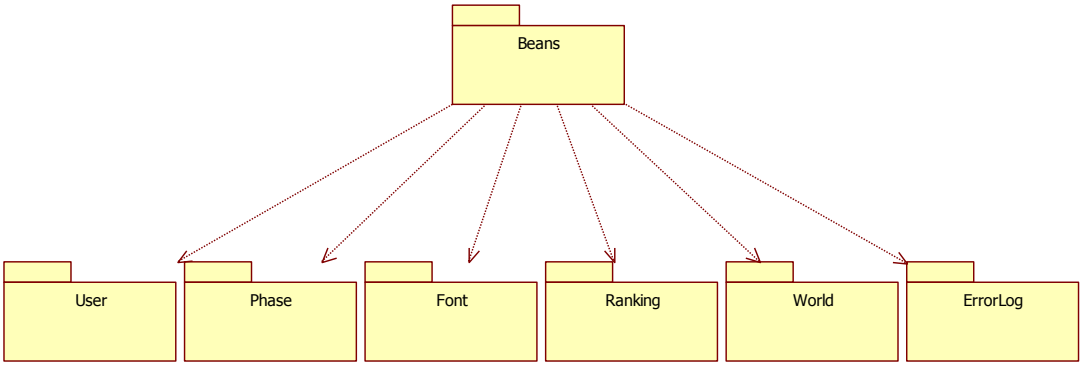
5.3.1.2.- Paquete servidor

En este paquete se encuentran las clases que comunican a los clientes con la capa de negocio.



5.3.1.3.- Paquete negocio

En este paquete se encuentran los paquetes que contienen las clases que gestionan todos los contenidos de la aplicación, que proporcionan las fases, los mundos, que gestionan las puntuaciones y los usuarios.



5.3.2.- Diagramas de clases

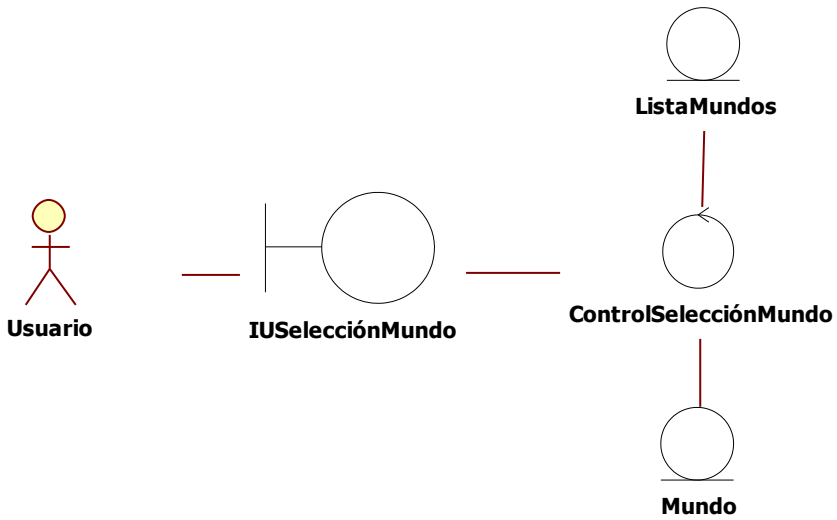
A la hora de modelar las clases de análisis se utilizan dos tipos de diagramas: estáticos o de colaboración. Para elaborar estos diagramas se utilizan tres tipos de clases:

- Interfaces: son clases que actúan de intermediarias en la comunicación entre usuario y sistema, entre sistemas.

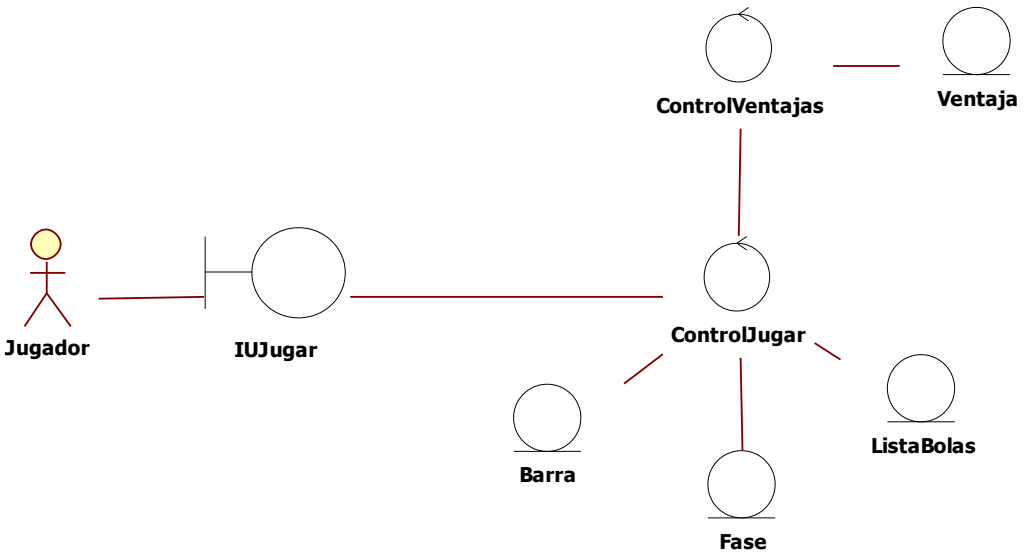
- Control: son las clases que realizan la lógica del modelo de negocio. Sirven para coordinar y comunicar clases (interfaces, datos y otras clases de control).

- Datos: representan un objeto de datos estático, normalmente una entidad del modelo de negocio. Generalmente son manipuladas por clases de control.

5.3.2.1.- Seleccionar mundo



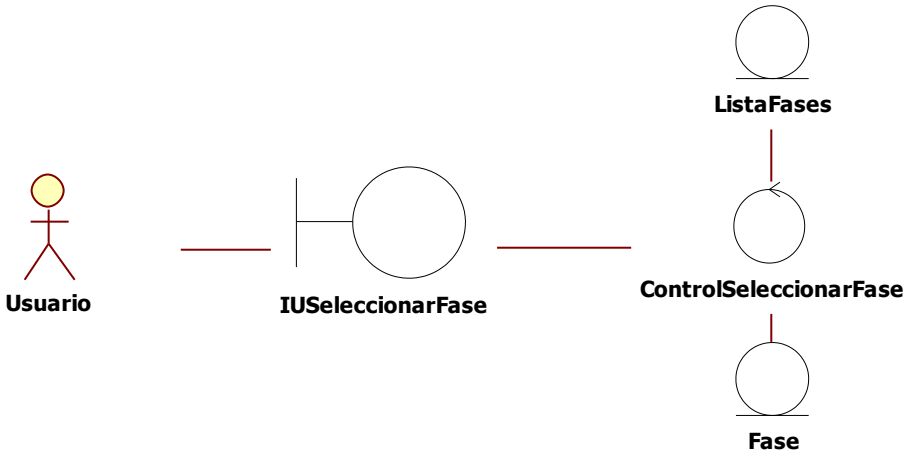
5.3.2.2.- Jugar



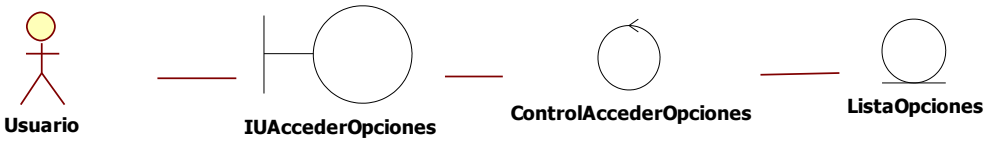
5.3.2.3.- Registrarse

5.3.2.4.- Invitar amigo

5.3.2.5.- Seleccionar fase

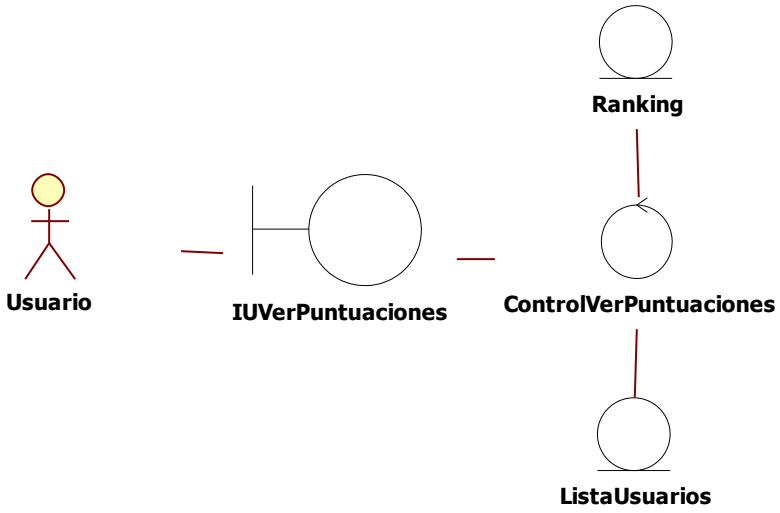


5.3.2.6.- Acceder opciones



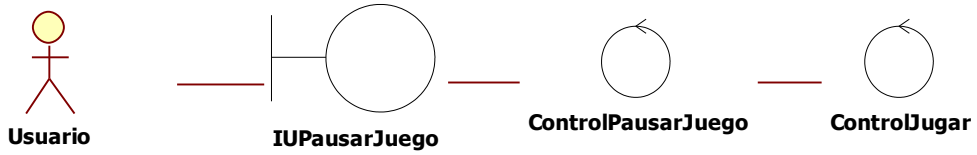
5.3.2.7.- Abandonar juego

5.3.2.8.- Ver puntuaciones

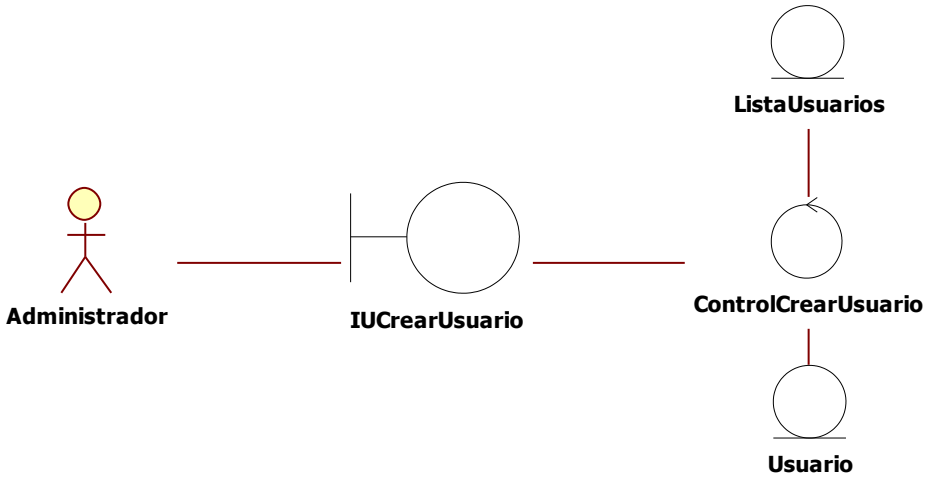


5.3.2.9.- Continuar partida

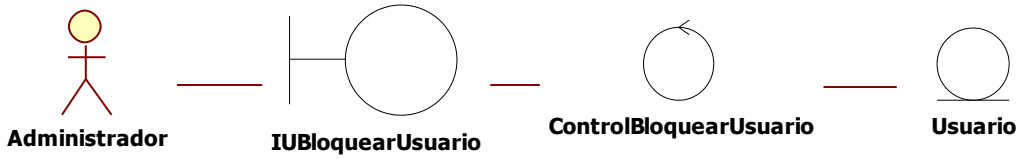
5.3.2.10.- Pausar juego



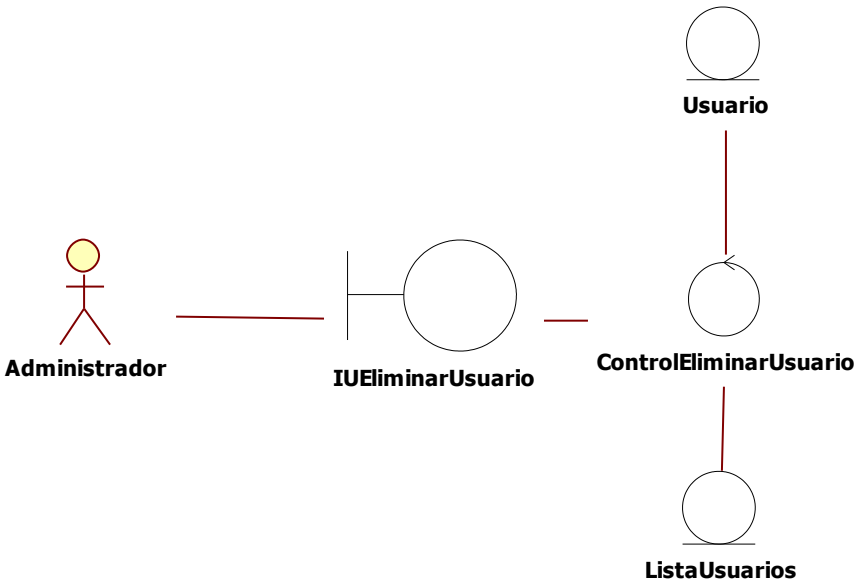
5.3.2.11.- Crear usuario



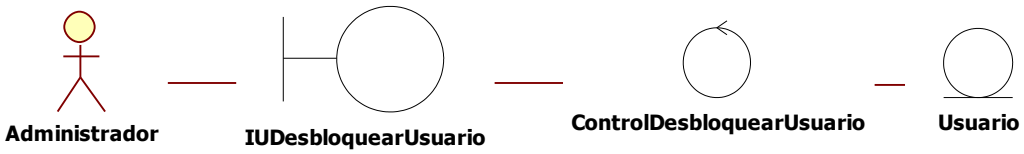
5.3.2.12.- Bloquear usuario



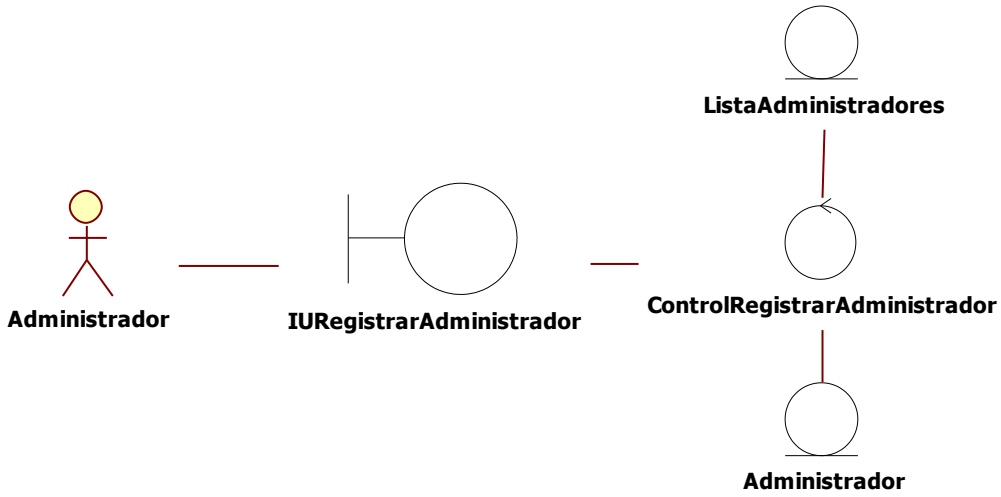
5.3.2.13.- Eliminar usuario



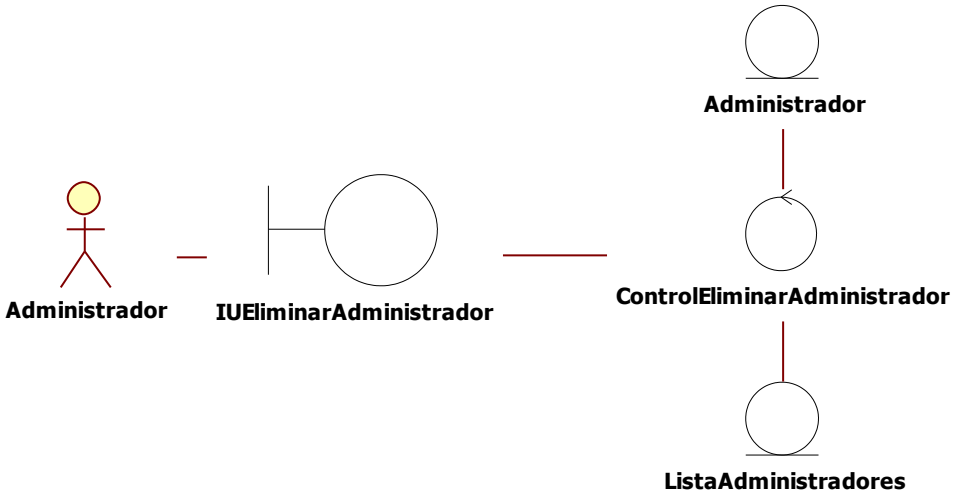
5.3.2.14.- Desbloquear usuario



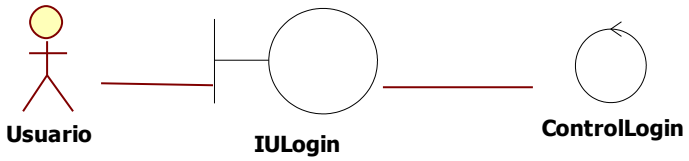
5.3.2.15.- Registrar administrador



5.3.2.16.- Eliminar administrador



5.3.2.17.- Logearse



5.4.- Modelo de diseño

5.4.1. Arquitectura del sistema

La arquitectura que se ha escogido para el desarrollo el sistema ha sido la arquitectura cliente-servidor, debido a las necesidades que se precisan sobre centralización de los datos y las fases, y debido a que pretende que se pueda jugar desde lugares remotos, a través de la página web en la que estará incrustado el applet.

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

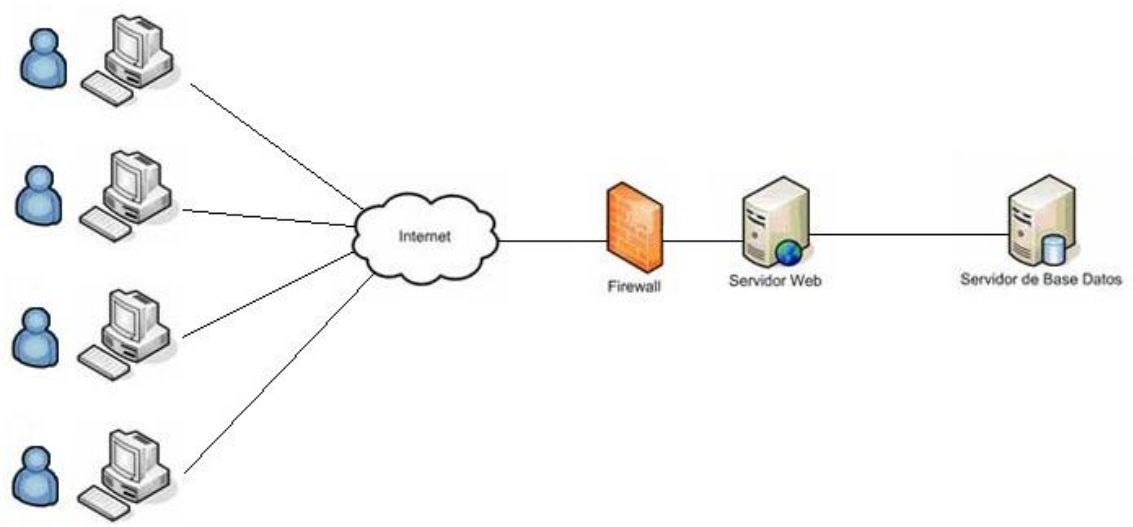
La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

La red cliente-servidor es aquella red de comunicaciones en la que todos los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se

concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta.

La vista de despliegue de un sistema contiene los nodos que forman la topología hardware sobre la que se ejecuta el sistema. Se preocupa principalmente de la distribución, entrega e instalación de las partes que constituyen el sistema. Los aspectos estáticos de esta vista se representan mediante los diagramas de despliegue y los aspectos dinámicos con diagramas de interacción, estados y actividades.

En este caso, existirán clientes que se conectarán a nuestro servidor web a través de internet. Hemos colocado un firewall para dar seguridad ante intrusos que puedan poner en peligro la integridad de nuestro sistema. A continuación, nuestro servidor web se conectará a un servidor de base de datos cuando sea necesario para consultas o actualizaciones de datos referentes a estados de partida, usuarios, mundos, fases, etc.



Es decir, nuestro sistema se dividirá en tres grandes componentes, que serán, el cliente, el servidor y la capa de negocio.

En cuanto a las tecnologías para el desarrollo de cada uno de los componentes se han escogido las tecnologías que proporciona Java, debido a que es un lenguaje de programación libre y a que proporciona múltiples herramientas, tanto para el desarrollo como para su uso con otras tecnologías.

5.4.2.- Organización del modelo de diseño

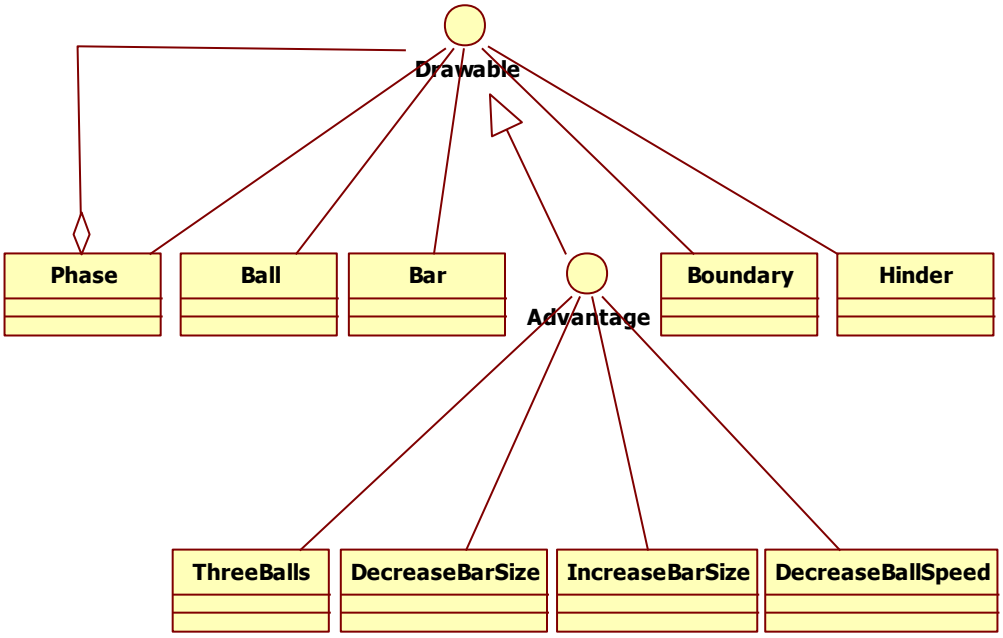
Al igual que en el apartado anterior, el modelo de diseño se divide en tres partes, una por cada capa del sistema.

5.4.3.- Diagramas de clases

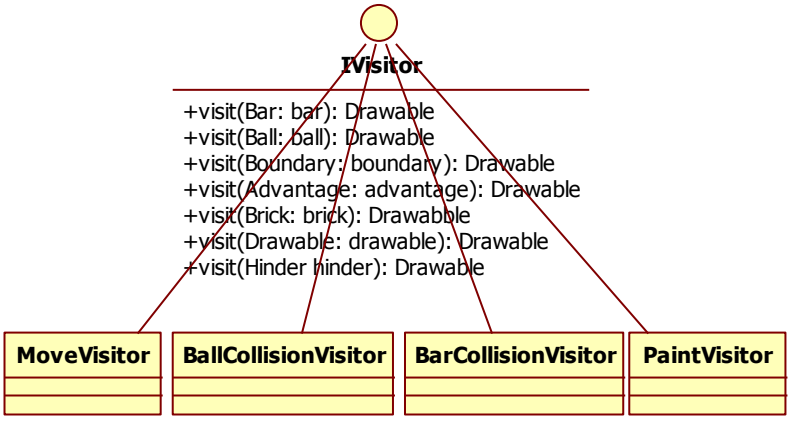
5.4.3.1.- Paquete cliente

En el cliente se definen las estructuras que controlan el desarrollo del juego. En primer lugar se definen los elementos que van a ser mostrados en la pantalla de juego.

En la imagen siguiente se define la estructura de clases que representa los elementos que forman parte de la lógica de juego. Para representar esta estructura se ha utilizado el patrón de diseño composite, como forma de representar que todos los objetos se pueden dibujar, y que existe un objeto que contiene una colección de objetos que se pueden dibujar, y que a la vez, el mismo también se puede dibujar. En general, el patrón composite sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

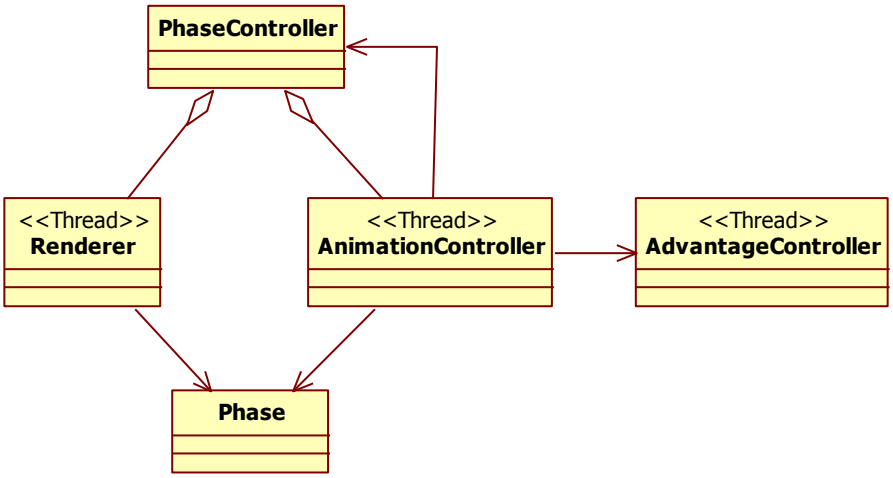


La utilización del patrón composite para representar la fase ofrece la posibilida de utilizar el patrón visitor para recorrer la fase y asignar un comportamiento a cada tipo de objeto. Este patrón nos permite separar el comportamiento de la estructura de un objeto. La idea básica es que se tiene un conjunto de clases que conforman la estructura de un objeto. Cada una de estas clases acepta a un visitante, que es capaz de diferenciar a qué objeto está visitando, y conoce el comportamiento de cada uno.

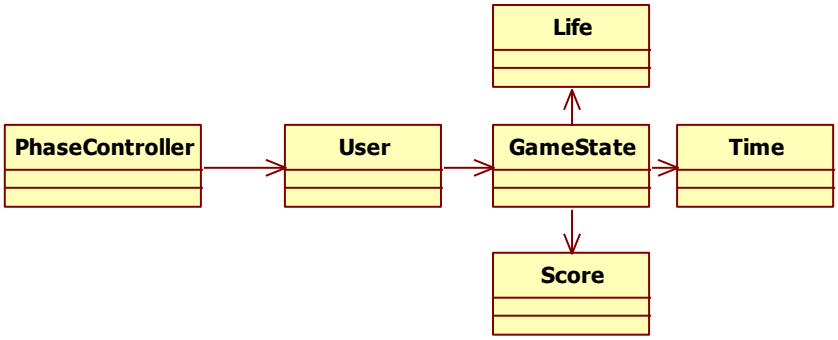


En este caso, se han utilizado visitors para el movimiento de todos los elementos de la fase, para el control de las colisiones de la bola con los elementos de la fase, para las colisiones de la barra con los elementos de la fase y para dibujar la fase al completo.

Para el control de la fase se crea la clase PhaseController, que activa dos hilos, uno para el control del movimiento (AnimationController) y otro para el control de la interfaz de usuario (Renderer). Cabe destacar que, debido a la complejidad de las ventajas, el control de las mismas se realiza en un hilo aparte que es controlado por el AnimationController.

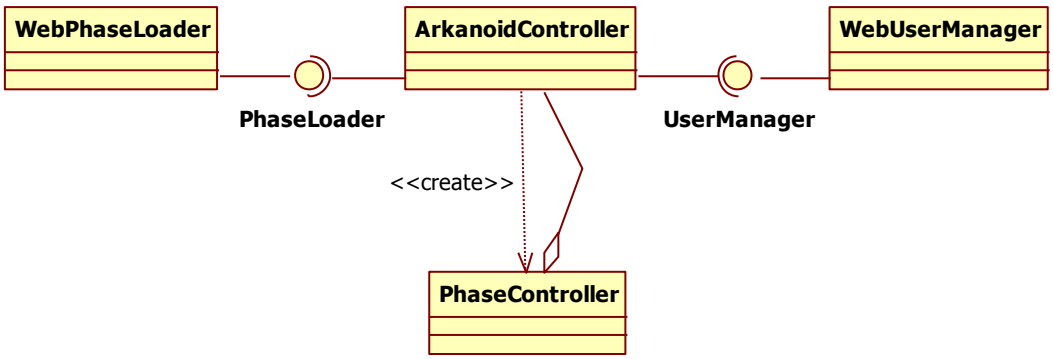


Además del control de los hilos anteriormente mencionados, la clase PhaseController se encarga de recoger los eventos de teclado que se suceden mientras los usuarios utilizan el sistema. A partir de estos eventos existe la posibilidad de que el estado del juego cambie, por lo que esta clase tiene la habilidad de detener los hilos que dan vida a la interfaz de usuario.

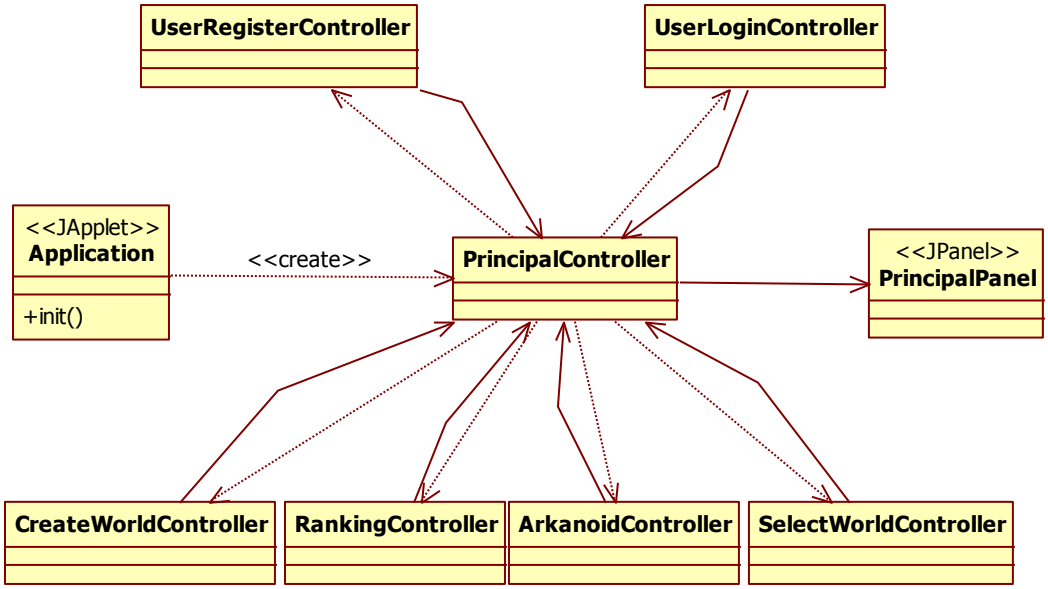


Por último, esta clase se encarga de revisar la fase que le ha inyectado el ArkanoidController, ya que si esta fase fue guardada anteriormente, es necesario restablecer la lista de bolas y la barra, y si éstos elementos no existen en la fase, el PhaseController es el encargado de crearlos, incluirlos en la misma y enviarla al Renderer y al AnimationController.

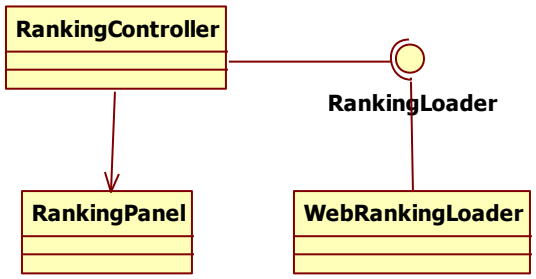
Como se especifica en el párrafo anterior, el ArkanoidController inyecta la fase al PhaseController, y para ello necesita descargar la fase de la capa de servidor. Esta clase además es la encargada de comunicarse con la capa de servidor para guardar las partidas del jugador.



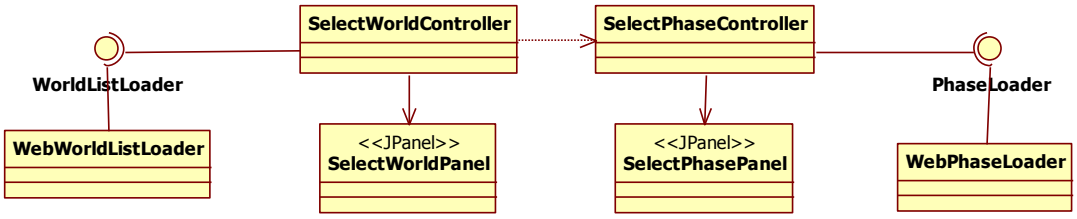
Como controlador principal de la aplicación se ha creado la clase PrincipalController, que se encarga de crear el menú principal, y activar el controlador necesario para llevar a cabo la operación que ha solicitado el usuario. Por lo que al inicio de la ejecución del cliente, el applet se despliega, y crea el PrincipalController, que recogerá las peticiones del usuario.



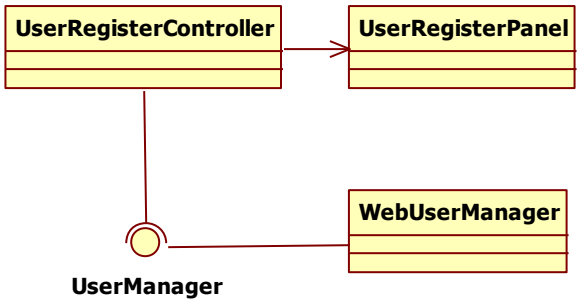
Para la consulta del ranking, el usuario hace click en el menú principal, en el botón "Ranking", y el PrincipalController, recoge la petición, y activa el RankingController. Éste se conecta a la capa de servidor, solicita el ranking, y rellena una tabla con los datos del mismo.



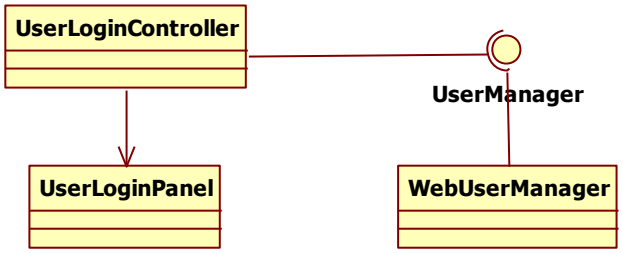
Cuando el usuario quiere seleccionar un mundo, hace click en el botón seleccionar mundo de la pantalla principal, y el PrincipalController activa el controlador SelectWorldController, que recogerá el mundo que el usuario desea jugar, y a continuación le mostrará las fases las que puede jugar de dicho mundo.



A la hora del registro, el usuario hace click en el botón “Register” y el PhaseController activa el controlador UserRegisterController, que muestra una ventana en la que se le solicitan al usuario los datos de registro.

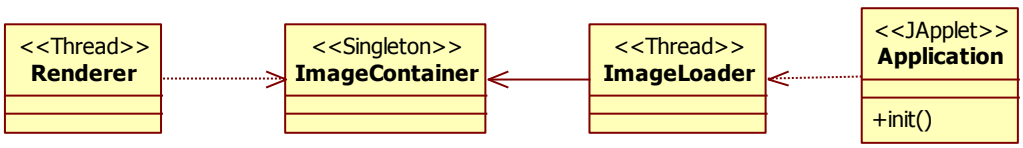


Para la identificación de los usuarios de cara al sistema, es necesario que el usuario haga click en el botón “Log in”, momento en el que el PrincipalController activa el UserLoginController, que recoge los datos de identificación.



Las clases anteriores, que incluyen la palabra “Web” en su nombre, son las que se conectan a la capa de servidor para interactuar y poder llevar a cabo sus operaciones.

Por último, cabe destacar un conjunto de clases que se han creado con el fin de optimizar la descarga de imágenes desde el servidor hacia el cliente. Se ha creado un hilo que comienza a funcionar desde que el applet se despliega, y desde ese momento comienza a descargar y almacenar las imágenes que se necesitarán para jugar.

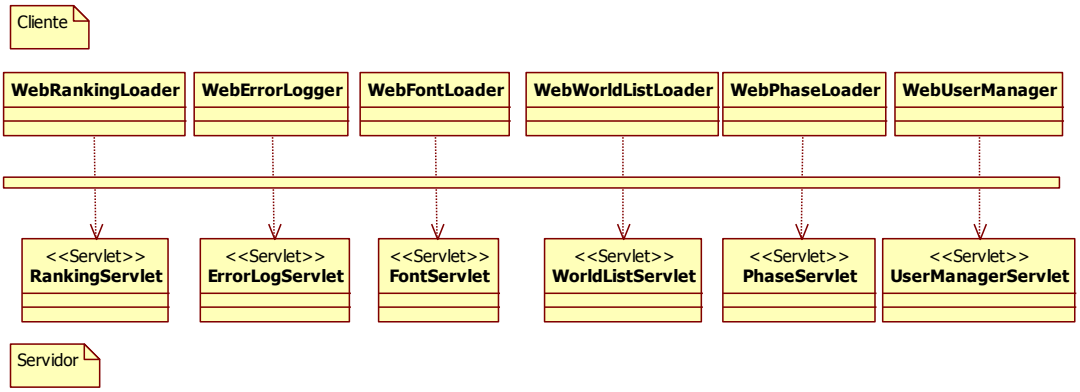


Como muestra la imagen anterior, el applet inicia el hilo ImageLoader, que utiliza el singleton ImageContainer para almacenar las imágenes. Posteriormente el hilo Renderer hará uso de esta clase para acceder a las imágenes ya descargadas. El patrón de diseño singleton está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto, es decir, garantizar que una clase sólo

tiene una instancia y proporcionar un punto de acceso global a ella. Para conseguir crear una única instancia se necesita que la propia clase sea la responsable de crear la única instancia, que exista un método de clase que permita el acceso a la instancia, y que el constructor de la clase sea privado para que no se puedan crear más instancias de forma externa a la clase.

5.4.3.2.- Paquete servidor

Esta capa sirve para hacer de pasarela entre el cliente y la capa de negocio, y además contiene la página web y despliega el applet en el navegador del usuario.

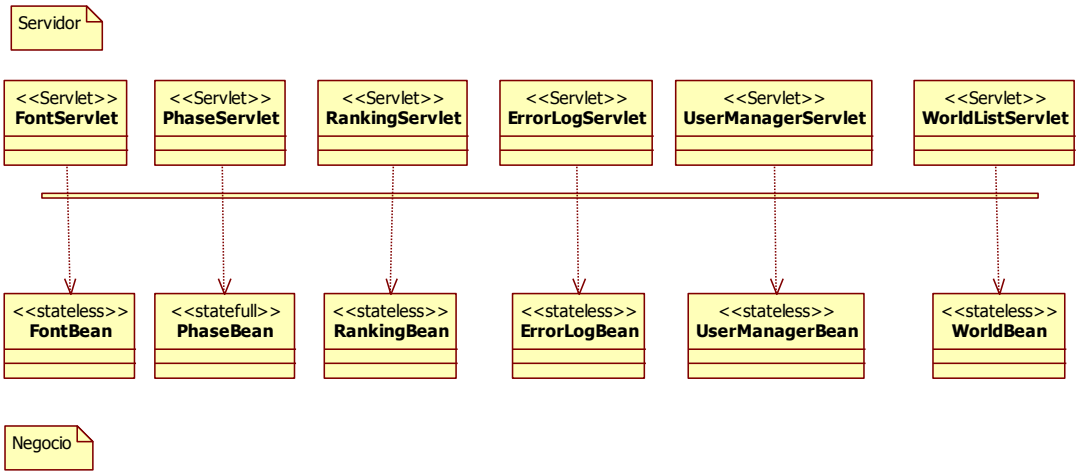


Como vemos en la imagen anterior, y como se comentó en el apartado anterior, las clases del cliente que contienen la palabra “Web” se conectan a los distintos servlets para solicitar datos al servidor, o para enviarlos al mismo.

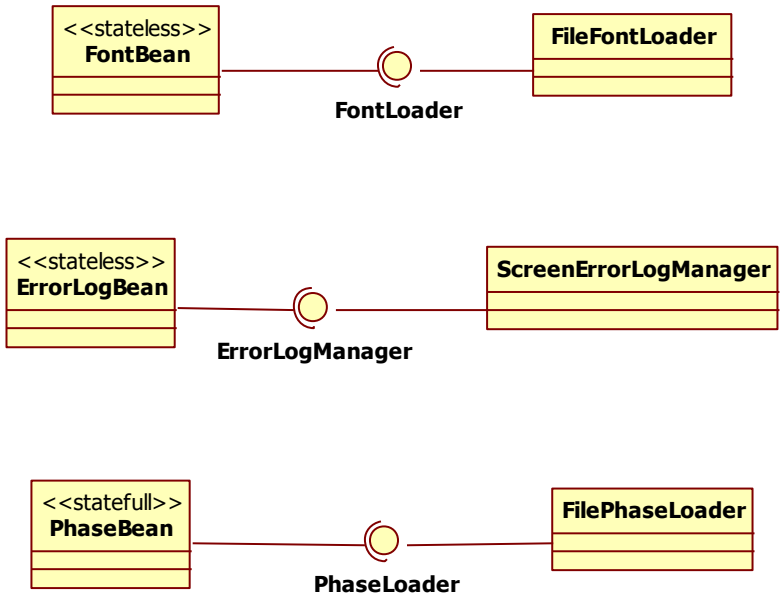
A través del RankingServlet se puede solicitar la lista de las mejores puntuaciones conseguidas en el juego. Por medio del ErrorLogServlet se puede informar al servidor sobre eventos o errores que se produzcan en el cliente. Utilizando el FontServlet se pueden descargar las fuentes que se utilizan en el cliente para crear la interfaz de usuario. A través del PhaseServlet se pueden cargar las fases y la lista de imágenes de cada mundo que representan a cada fase. El UserManagerServlet proporciona los métodos para identificarse ante el sistema, registrarse en el sistema, guardar el estado de la partida actual y cargar una partida guardada anteriormente. Por último, el WorldListServlet proporciona la lista de mundos del sistema.

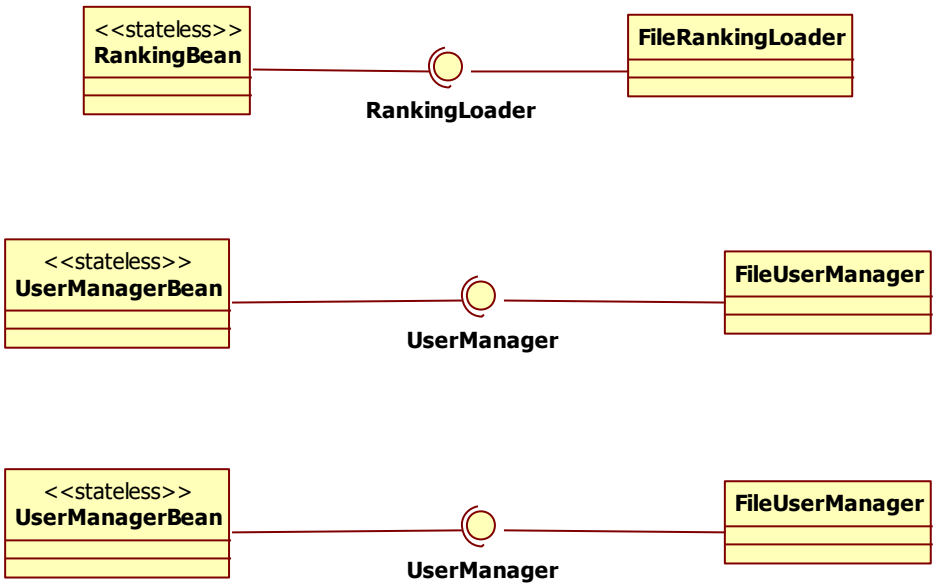
5.4.3.3.- Paquete negocio

En esta capa se encuentran las clases que proporcionan las funcionalidades a la capa anterior, implementadas a través de beans. Es decir, para que la capa de servidor sea capaz de satisfacer las peticiones de la capa de cliente, los servlets recogen las peticiones y las envían a la capa de negocio, por medio de los beans que existen en dicha capa.



Ya en la capa de negocio, los beans utilizan clases que permiten la carga de los datos que se soliciten desde cliente. Para disminuir el acoplamiento se han intercalado interfaces, con el objetivo de que los beans utilicen éstas interfaces, en vez de clases concretas. En un futuo, esto permitirá que sea más sencillo implementar diferentes métodos de almacenamiento de los datos, ya que en la actualidad se utilizan ficheros para el almacenamiento. Pero, por ejemplo, si se necesitase almacenar los usuarios en una base de datos, sería tan simple como crear una clase que implemente la interfaz UserManager, y dentro de ésta, implementar los métodos que se encarguen de manipular la base de datos.





5.4.4.- Diagramas de secuencia

5.5.- Implementación

En esta sección se detallan las decisiones técnicas tomadas en relación a diferentes aspectos de la fase de implementación. Evidentemente, esta fase del proyecto ha sido la que ha tomado la mayor parte de la planificación temporal del proyecto, y ha sido la de mayor complejidad.

5.5.1.- Applets

Para la implementación de la capa de cliente se ha elegido un applet, debido a que se quiere que el cliente se ejecute a través de un navegador web.

Un applet es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo, en un navegador web. El applet debe ejecutarse en un contenerdor, que le proporciona un programa anfitrión, mediante un plugin, o en aplicaciones para teléfonos móviles que soportan el modelo de programación por applets.

A diferencia de un programa, un applet no puede ejecutarse de manera independiente, ofrece información gráfica y a veces interactúa con el usuario, típicamente carece de sesión y tiene privilegios de seguridad restringidos. Un applet normalmente lleva a cabo una función muy específica que carece de uso independiente.

En nuestro caso, se ha utilizado un Java Applet, que se pueden ejecutar en un navegador web utilizando la Java Virtual Machine (JVM), o en el AppletViewer de Sun.

Entre sus características podemos mencionar que puede incrustarse en un documento HTML, es decir, en una página web. Cuando un navegador carga una página

web que contiene un applet, éste se descarga en el navegador web y comienza a ejecutarse. Esto permite crear programas que cualquier usuario puede ejecutar con tan solo cargar la web en su navegador.

La utilización de este tipo de applets tiene las siguientes ventajas:

- Son multiplataforma, es decir, funcionan en Linux, Windows Mac OS, y el cualquier sistema operativo para el cual exista una JVM.

- El mismo applet puede trabajar en todas las versiones de Java, y no sólo en la última versión del plugin. Sin embargo, si un applet requiere una versión posterior del Java Runtime Enviroment (JRE), el cliente se verá obligado a esperar durante la descarga de la nueva JRE.

- Es compatible con la mayoría de los navegadores web.

- Puede ser almacenado en la memoria caché de los navegadores web, de modo que se cargará rápidamente cuando se vuelva a cargar la página web, aunque puede quedar atascado en la caché, cuando se publican nuevas versiones.

- Puede tener acceso completo a la máquina en la que se está ejecutando, si el usuario lo permite.

- Puede ejecutarse a velocidades comparables a las de otros leguajes compilados, como C++.

- Puede trasladar el trabajo del servidor al cliente, haciendo una solución web más escalable, teniendo en cuenta el número de usuarios o clientes.

Pero no todo son ventajas, ya que la utilización de esta tecnología puede tener las siguientes desventajas:

- Requiere el plugin de java, que no está disponible por defecto para todos los navegadores web.

- No se puede iniciar la ejecución de la aplicación hasta que la JVM no esté en funcionamiento, y esto puede tomar tiempo la primera vez que se ejecuta un applet.

- Si no está firmado como confiable, tiene un acceso limitado al sistema del usuario, en particular no tiene acceso al disco duro del cliente o al portapapeles.

- Algunas organizaciones sólo permiten la instalación de software a los administradores. Como resultado, muchos usuarios sin privilegios para instalar el plugin en su navegador, no pueden ver los applets.

- Un applet podría exigir una versión específica del JRE.

- Puede tener vulnerabilidades que permitan ejecutar código malicioso.

5.5.2.- Movimiento de los elementos de la fase

De todos los elementos de la fase, sólo se mueven las ventajas, las bolas y la barra, pero todos los hacen con el mismo tipo de movimiento, el movimiento rectilíneo uniforme. Un movimiento es rectilíneo cuando el móvil describe una trayectoria recta, y es uniforme cuando su velocidad es constante en el tiempo, dado que su aceleración es nula.

En el caso de la bola, este tipo de movimiento se produce en los dos ejes, es decir, tanto en el eje x como en el eje y. En el caso de las ventajas el movimiento se produce sólo en el eje y, ya que no se mueven horizontalmente. Por último, en el caso de la barra, el movimiento se produce sólo en el eje x.

Las ventajas y las bolas se mueven a través de uno de los visitadores, que van sumando a la posición actual, el incremento de posición, mientras que la barra se mueve sólo cuando existen eventos de teclado que lo soliciten.

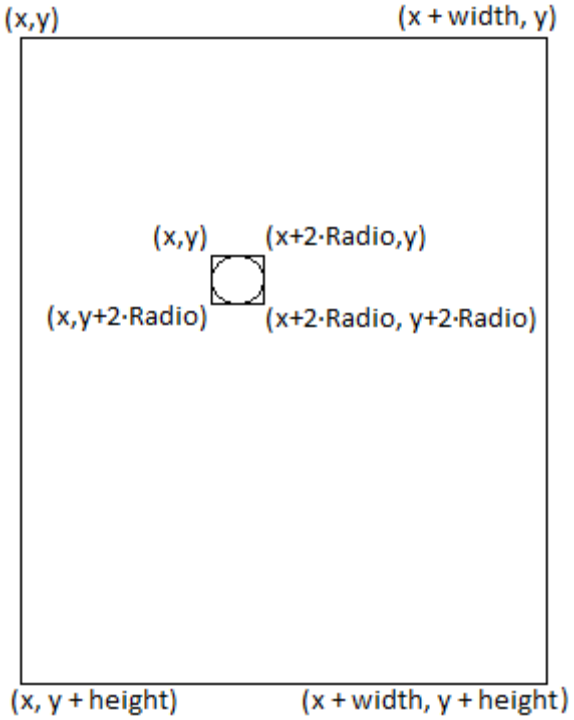
5.5.3.- Detección de colisiones

Las colisiones en este sistema se refieren a cuáles de los elementos de la fase deben interactuar con cuáles otros, es decir, qué elementos chocan entre sí y qué comportamiento deben tener después del choque.

5.5.3.1.- Detección de colisiones de la bola

En el caso de la bola, debe chocar con los bloques, con los límites del área de juego y con la barra. Las bolas no deben chocar entre sí, y tampoco deben chocar con las ventajas que estén cayendo.

En primer lugar, se va a tratar el estudio de las colisiones de la bola con los límites de la pantalla de juego. A continuación se muestra una imagen en la que aparecen los puntos necesarios para controlar que la bola permanezca en la pantalla de juego.



A la vista de la imagen anterior, se necesitará que la bola cambie su dirección cuando:

$$x_{bola} \leq x_{frontera}\ OR\ x_{bola} + 2 \cdot radio_{bola} \geq x_{frontera} + ancho_{frontera}$$

$$y_{bola} \leq y_{frontera}\ OR\ y_{bola} + 2 \cdot radio_{bola} \geq y_{frontera} + alto_{frontera}$$

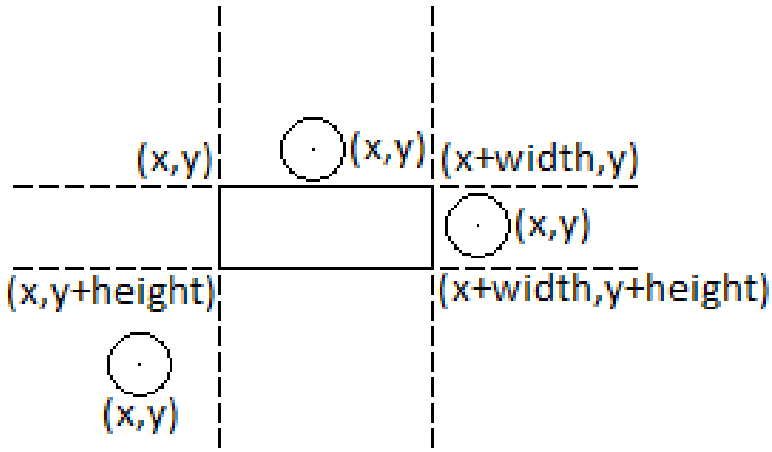
En el caso que se de cualquiera de las dos circunstancias descritas en las ecuaciones anteriores, será necesario cambiar el signo de la velocidad de la bola para hacer que ésta cambie de sentido. Si se cumple la primera ecuación, será necesario cambiar la velocidad del eje X, y si se cumple la segunda ecuación, será necesario cambiar la velocidad del eje Y.

En segundo lugar, se tratan las colisiones de la bola con los bloques. Para saber cuándo colisiona la bola con el bloque se deben tener en cuenta tres situaciones:

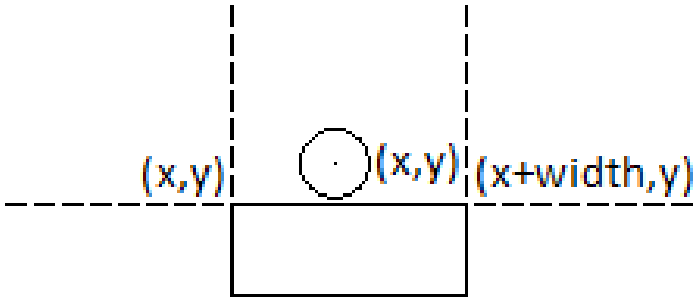
- La bola se encuentre por encima o por debajo del bloque. En este caso, la bola colisionará cuando la distancia desde el centro de la bola hasta la mitad de la altura del bloque sea menor que el radio de la bola más la mitad del alto del bloque. La colisión producirá un cambio de signo de velocidad en el eje y.

- La bola se encuentre a la izquierda o a la derecha del bloque. En este caso, la bola colisionará cuando la distancia desde el centro de la bola hasta la mitad del ancho del bloque sea menor que el radio de la bola más la mitad del ancho del bloque. La colisión producirá un cambio de signo de velocidad en el eje x.

- La bola se encuentre en cualquier otro lugar. En este caso, la bola colisionará cuando la distancia desde el centro de la bola hasta cualquiera de los vértices del bloque, sea menor que el radio de la bola.



Por último, para tratar las colisiones de la bola con la barra, hay que controlar que el centro de la bola esté encima de la barra, y que la distancia desde el centro de la bola hasta la parte superior de la barra sea menor que el radio de la bola.



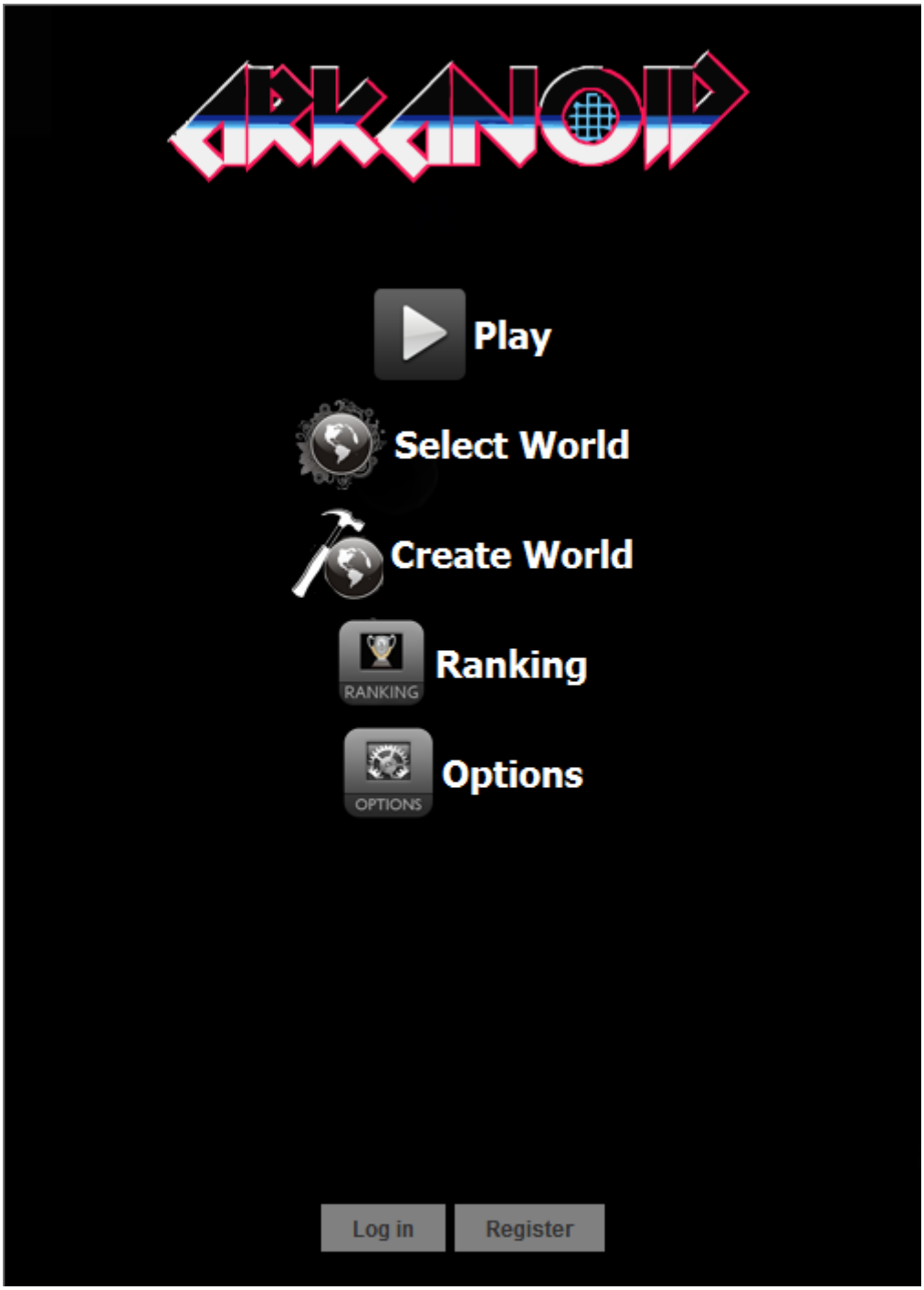
5.5.3.2.- Detección de colisiones de la barra

En cuanto a las colisiones con la barra, el único elemento que tendrá que cambiar su comportamiento cuando entre en contacto con ella serán las ventajas. Ya que las colisiones de la bola con la barra se tratan en el apartado anterior.

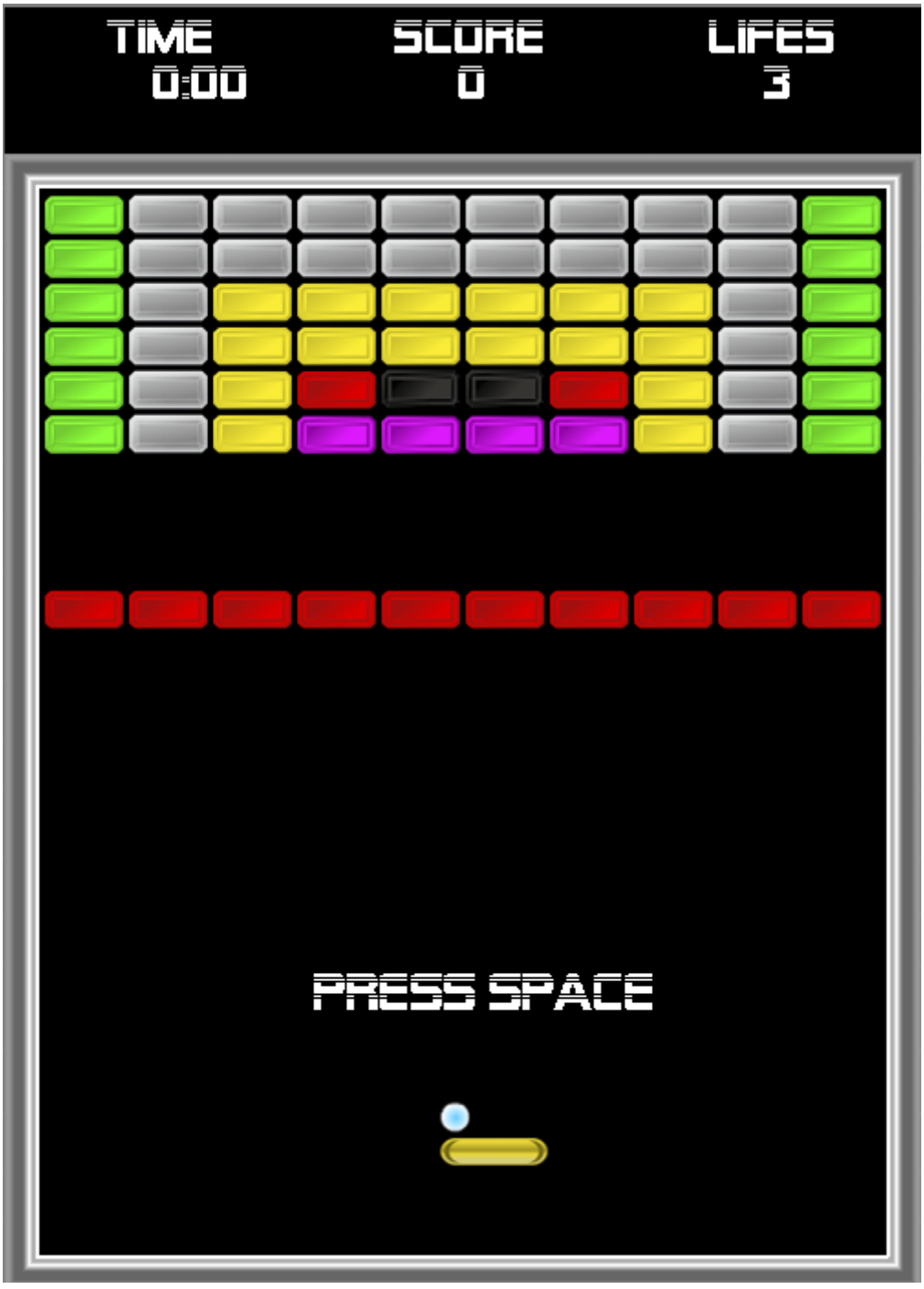
En la siguiente imagen se muestran los puntos críticos necesarios para conocer cuándo colisiona una ventaja con la barra.

5.5.4.- Diseño de la interfaz de usuario

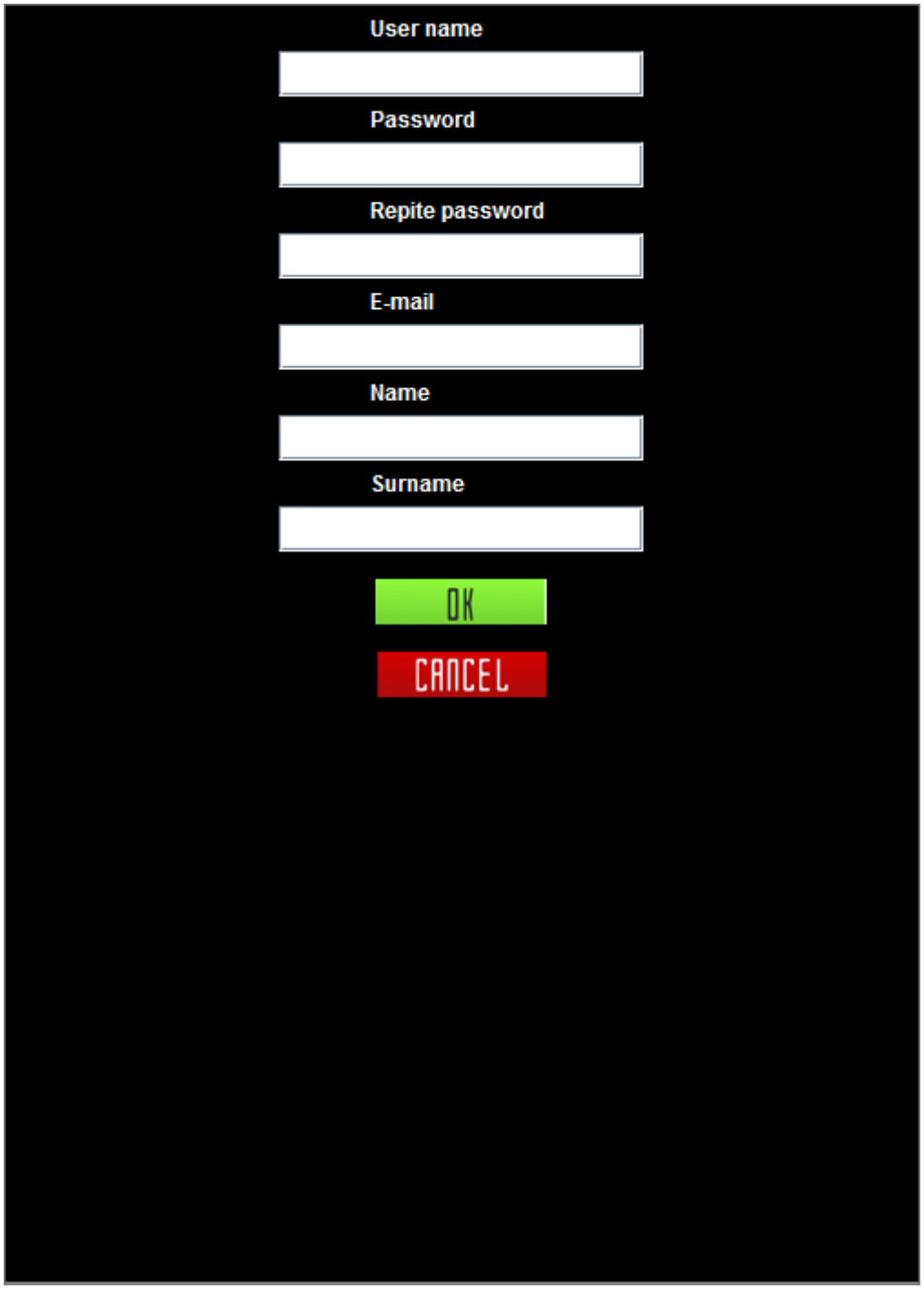
5.5.4.1.- Menú principal



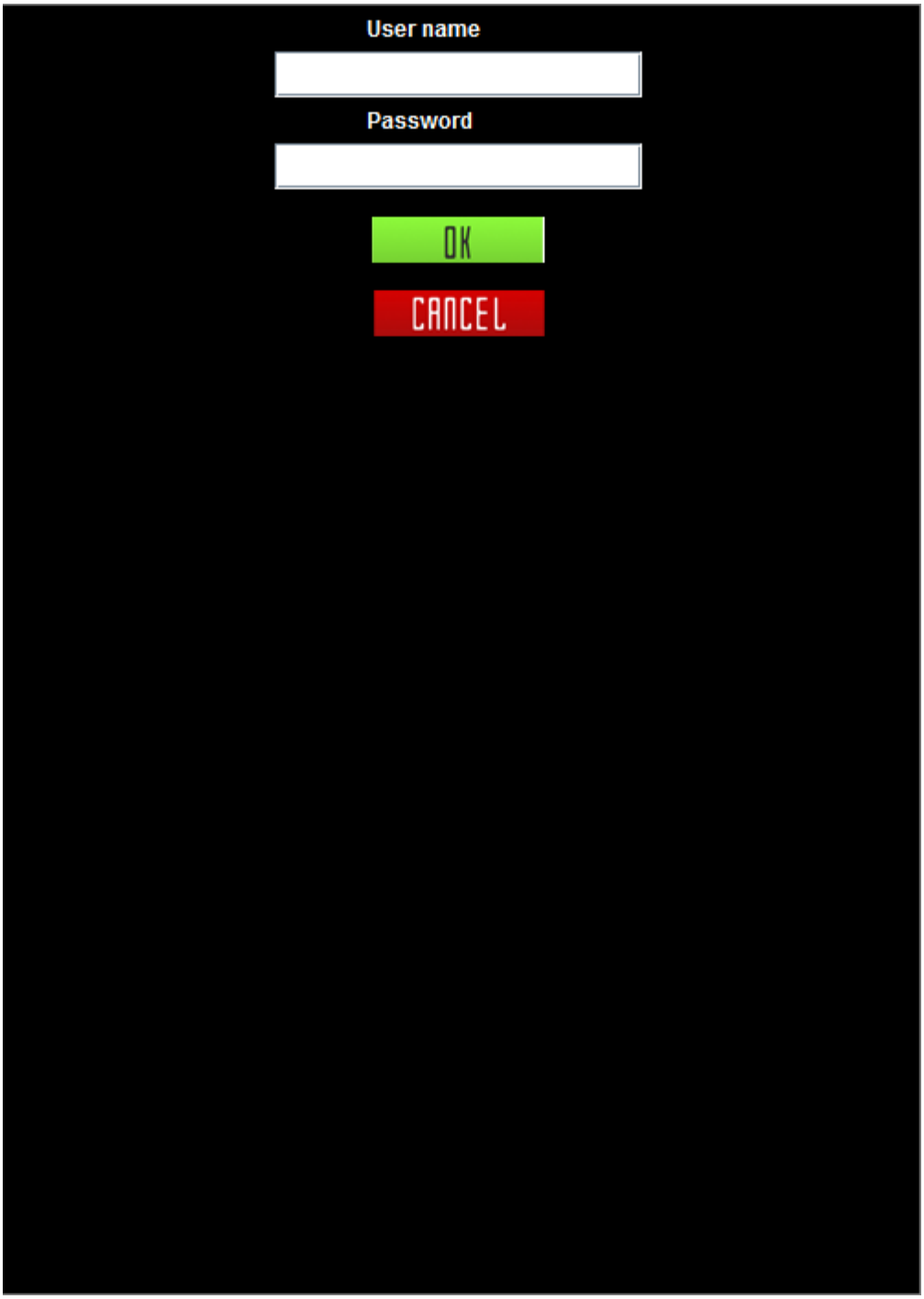
5.5.4.2.- Pantalla de juego



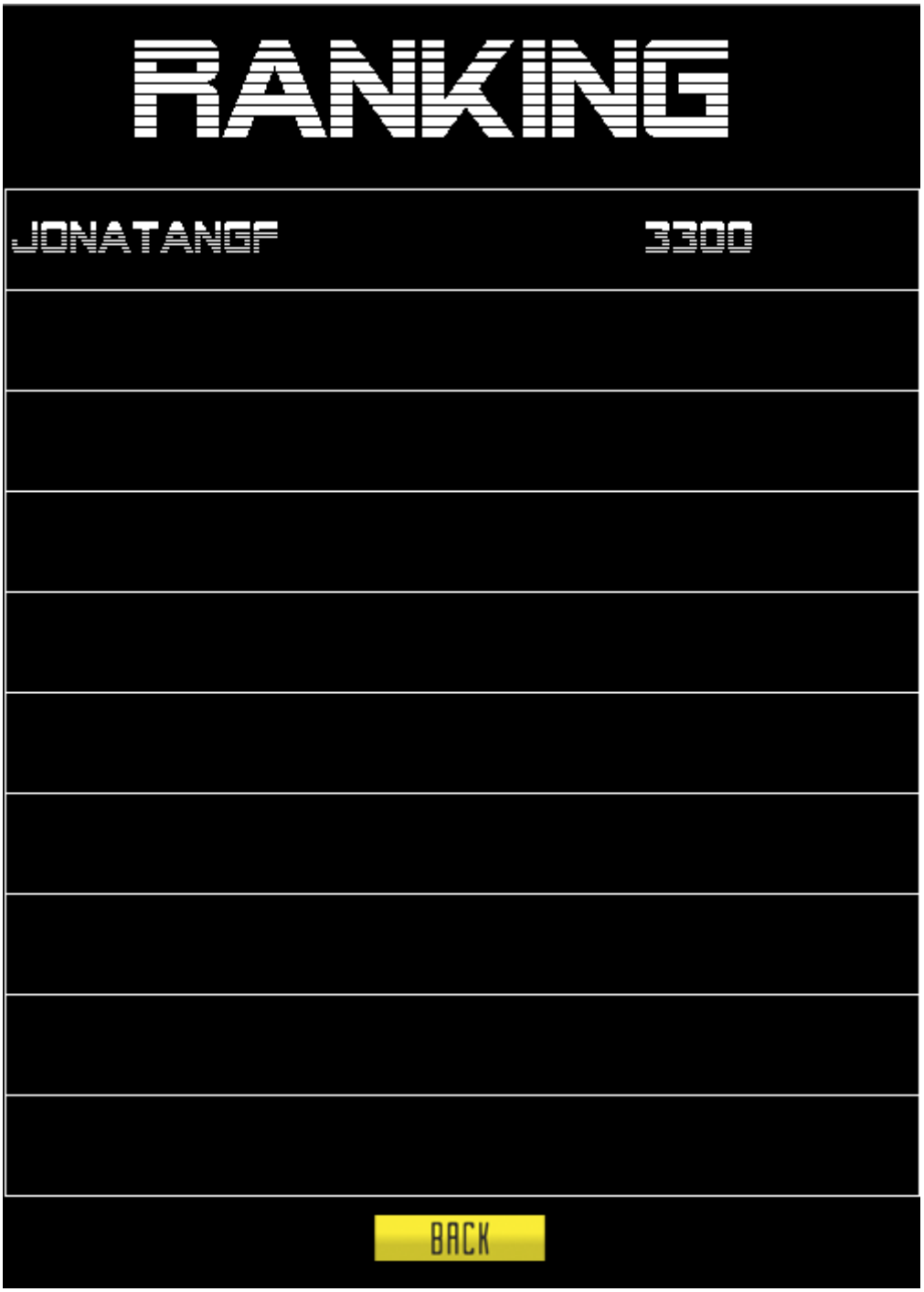
5.5.4.3.- Menú de registro



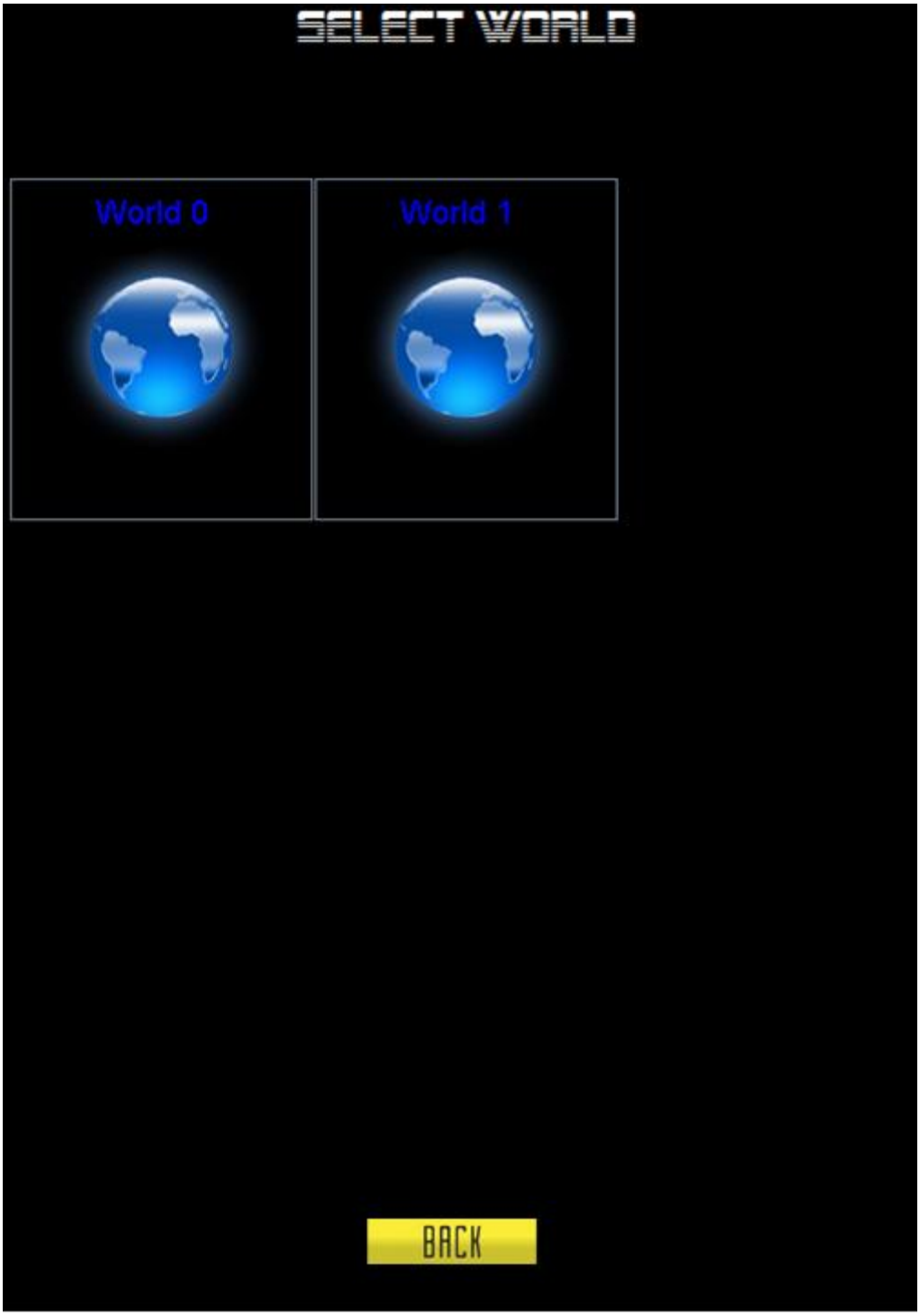
5.5.4.4.- Menú de log in



5.5.4.5.- Ranking



5.5.4.6.- Pantalla seleccionar mundo



5.5.4.7.- Pantalla seleccionar fase

6.- Conclusiones y trabajo futuro

6.1.- Conclusiones

Durante el desarrollo de este proyecto se ha desarrollado un sistema de tres capas, implementado en tres aplicaciones distintas, y dependientes entre ellas. Esto permite crear un sistema distribuido que funcione en tres equipos distintos, y con posibilidad de atender a múltiples usuarios. Además, queda abierto a introducir una cuarta capa, en la que se podría encontrar la base de datos, y que podría funcionar en otro equipo.

Una vez concluido el desarrollo del proyecto y habiendo realizado las pruebas necesarias sobre el mismo, se puede concluir que se han cumplido los objetivos marcados el inicio del proyecto.

Como conclusión a este trabajo cabe destacar la importancia de las fases de planificación, y la importancia de la fase de planificación para que cualquier proyecto salga adelante. Además, es de vital importancia crear pequeños prototipos que permitan a los desarrolladores encontrarse con los problemas propios de la tecnología o del producto, para tomar decisiones antes de entrar en el desarrollo del producto final.

Finalmente, y como valoración personal del proyecto, he de decir que ha sido muy gratificante el hecho de realizar un trabajo de fin de grado de estas características, ya que las aplicaciones distribuidas y de arquitectura cliente-servidor están en auge. Además, este trabajo me ha permitido conocer en profundidad ciertas tecnologías, proporcionándome nuevas herramientas y métodos de resolución de problemas, que estoy seguro, me serán muy útiles en mi vida profesional.

6.2.- Trabajo futuro

Como trabajo futuro, y para completar este producto se cree necesaria la implementación de varios aspectos, con el objetivo de obtener recursos financieros del mismo.

En primer lugar, la implementación de un sistema de compra de mundos, que permita vender nuevos mundos con un número determinado de fases cada uno. En relación a esto, implementar una herramienta de creación de mundos para los usuarios, con el objetivo de que puedan crear sus propios mundos, e incluso venderlos para obtener beneficios de ello.

En segundo lugar, implementación de un sistema de gestión de la aplicación, que permita conocer datos sobre las ventas, número de usuarios activos, número de usuarios registrados, y demás indicadores que permitan conocer el estado del sistema y la

aceptación del mismo. Esto es necesario para reaccionar a tiempo y cambiar aspectos del sistema, para que sea más atractivo a los usuarios.

Por último, se considera necesaria una refactorización del código, y un refinado de la arquitectura, debido a que en la fase de implementación se han encontrado ciertos problemas que han provocado que algunas clases tengan más de una responsabilidad. En sí mismo este hecho no provoca ningún problema, pero este desarrollo va en contra del principio de responsabilidad única, y en un futuro, si el sistema creciese, podría provocar que los cambios que fuese necesario realizar, fueran más costosos.

Anexo I: Competencias

G1. Poseer y comprender conocimientos en un área de estudio (Ingeniería Informática) que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.

G2. Aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.

G3. Reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.

G4. Transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.

G5. Desarrollar aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

N1. Comunicarse de forma adecuada y respetuosa con diferentes audiencias (clientes, colaboradores, promotores, agentes sociales, etc.), utilizando los soportes y vías de comunicación más apropiados (especialmente las nuevas tecnologías de la información y la comunicación) de modo que pueda llegar a comprender los intereses, necesidades y preocupaciones de las personas y organizaciones, así como expresar claramente el sentido de la misión que tiene encomendada y la forma en que puede contribuir, con sus competencias y conocimientos profesionales, a la satisfacción de esos intereses, necesidades y preocupaciones.

N2. Cooperar con otras personas y organizaciones en la realización eficaz de funciones y tareas propias de su perfil profesional, desarrollando una actitud reflexiva sobre sus propias competencias y conocimientos profesionales y una actitud comprensiva y empática hacia las competencias y conocimientos de otros profesionales.

N3. Contribuir a la mejora continua de su profesión así como de las organizaciones en las que desarrolla sus prácticas a través de la participación activa en procesos de investigación, desarrollo e innovación.

N4. Comprometerse activamente en el desarrollo de prácticas profesionales respetuosas con los derechos humanos así como con las normas éticas propias de su ámbito profesional para generar confianza en los beneficiarios de su profesión y obtener la legitimidad y la autoridad que la sociedad le reconoce.

N5. Participar activamente en la integración multicultural que favorezca el pleno desarrollo humano, la convivencia y la justicia social.

T1. Capacidad para concebir, redactar, organizar, planificar, desarrollar y firmar proyectos en el ámbito de la ingeniería en informática que tengan por objeto, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada, la concepción, el desarrollo o la explotación de sistemas, servicios y aplicaciones informáticas. (G1, G2)

T2. Capacidad para dirigir las actividades objeto de los proyectos del ámbito de la informática, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada. (G1, G2)

T3. Capacidad para diseñar, desarrollar, evaluar y asegurar la accesibilidad, ergonomía, usabilidad y seguridad de los sistemas, servicios y aplicaciones informáticas, así como de la información que gestionan. (G1, G2)

T5. Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería del software como instrumento para el aseguramiento de su calidad, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada. (G1, G2)

T6. Capacidad para concebir y desarrollar sistemas o arquitecturas informáticas centralizadas o distribuidas integrando hardware, software y redes, de acuerdo con los conocimientos adquiridos Página 2 de 6según lo establecido en apartado 5 de la resolución indicada. (G1, G2)

T7. Capacidad para conocer, comprender y aplicar la legislación necesaria durante el desarrollo de la profesión de Ingeniero Técnico en Informática y manejar especificaciones, reglamentos y normas de obligado cumplimiento. (N4)

T8. Conocimiento de las materias básicas y tecnologías, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran versatilidad para adaptarse a nuevas situaciones.

T11. Capacidad para analizar y valorar el impacto social y medioambiental de las soluciones técnicas, comprendiendo la responsabilidad ética y profesional de la actividad del Ingeniero Técnico en Informática.

CII01. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

CII02. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

CII04. Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.

CII18. Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

TFG01. Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

Anexo II: Legislación vigente

La Ley Orgánica de Protección de Datos tiene por objetivo garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar.

Para cumplir con esta ley se deben tener en cuenta, al menos, los conceptos que se definen a continuación:

- Inscripción de ficheros

Todos y cada uno de los ficheros que contengan datos personales deben estar inscritos en el Registro General de Protección de Datos, para que sean públicos y accesibles para su consulta por cualquier interesado.

- Calidad de los datos

En cumplimiento del principio de calidad de los datos regulado en el artículo 4 de la LOPD, se deben tratar los datos de carácter personal aplicando las siguientes reglas:

- Solamente se podrán recoger los datos de carácter personal utilizando medios que no sean fraudulentos, desleales o ilícitos.

- Deberá recoger aquellos datos personales que sean adecuados, pertinentes y no excesivos en relación a la finalidad que persiga con su tratamiento.

- Solamente utilizará los datos para finalidades compatibles con la finalidad que originó su recogida.

- Deberá mantener los datos exactos y actualizados de forma que respondan con veracidad a la situación actual del titular.

- Deberá cancelar los datos cuando hayan dejado de ser necesarios para la finalidad que originó su recogida.

- Se deberán almacenar los datos personales de manera que el titular de los datos pueda ejercer su derecho de acceso cuando lo considere oportuno.

- Deber de información

Se tiene el deber de informar al titular de los datos de modo expreso, preciso e inequívoco de todos los extremes regulados en el artículo 5 de la LOPD. El deber de información surge desde el mismo momento en que se recogen los datos, pero se cumple de diferente manera en función de si los datos se obtienen directamente de su titular o si provienen de otras fuentes.

- Consentimiento del afectado

Sólo se podrán tratar los datos de carácter personal si se dispone del consentimiento del titular, salvo que el tratamiento esté fundamentado en una de las excepciones legalmente previstas. Para cumplir con este principio, se debe tener claro cuando se necesita el consentimiento, y, en caso de necesitarlo, se deberá obtener legalmente teniendo en cuenta que, para que sea válido, debe tratarse de una manifestación de voluntad libre, inequívoca, específica e informada.

- Datos especialmente protegidos

Si se utilizan datos de carácter personal que revelen la ideología, afiliación sindical, religión y creencias o datos que hagan referencia al origen racial, a la salud y a la vida sexual, se debe tener en cuenta que se trata de datos especialmente protegidos y que deben ser tratados con la máxima cautela. En relación al tratamiento de este tipo de datos, y teniendo en cuenta lo previsto en el artículo 7 de la LOPD, se deben tener en cuenta los siguientes aspectos:

- No se puede obligar al ciudadano a declarar sobre su ideología, religión o creencias, y en caso de pedirle el consentimiento para tratar este tipo de datos, debe advertirle acerca de su derecho a no prestarlos.

- Sólo se podrán utilizar los datos de carácter personal que revelen la ideología, afiliación sindical, religión y creencias si se dispone del consentimiento expreso y por escrito del afectado.

- Sólo se podrán utilizar los datos de carácter personal que hagan referencia al origen racial, a la salud y a la vida sexual si se dispone del consentimiento expreso.

- Seguridad de los datos

En cumplimiento del artículo 9 de la LOPD, se deben adoptar medidas de índole técnica y organizativa, que garanticen la seguridad de los datos de carácter personal y eviten su alteración, pérdida, tratamiento o acceso no autorizado, habida cuenta del estado de la tecnología, la naturaleza de los datos almacenados y los riesgos a que estén expuestos, ya provengan de la acción humana o del medio físico o natural.

- Deber de secreto

En cumplimiento del artículo 10 de la LOPD, se está obligado a guardar secreto profesional sobre los datos tratados y a mantener la confidencialidad de los mismos dentro de su organización, debiendo tener en cuenta que la ley extiende el deber de secreto a cualquier persona o entidad que intervenga en cualquiera de las fases de tratamiento, aun después de finalizar sus relaciones con el titular del fichero o, en su caso, con el responsable del mismo.

- Comunicación de datos

Cuando se pretenda entregar los datos de carácter personal almacenados a otra persona, empresa o entidad para que los trate por su cuenta y bajo su propia responsabilidad, se debe hacer aplicando lo dispuesto en el artículo 11 de la LOPD, teniendo en cuenta que, de manera general, los datos de carácter personal objeto de tratamiento sólo podrán ser comunicados a un tercero para el cumplimiento de fines directamente relacionados con las funciones legítimas del cedente y del cesionario, con el previo consentimiento del interesado, salvo que la comunicación esté fundamentada en alguna de las excepciones previstas en el artículo 11.2.

- Acceso a los datos por cuenta de terceros

Cuando se contrate a otra persona, empresa o entidad para que preste algún servicio utilizando los datos de carácter personal almacenados se produce una relación jurídica denominada "acceso a los datos por cuenta de terceros" en la que quien presta el servicio adquiere la condición de "encargado del tratamiento" y se limita a tratar los datos por cuenta del responsable del fichero, siguiendo estrictamente sus instrucciones y devolviendo o destruyendo los datos una vez haya finalizado el servicio contratado.

- Ejercicio de los derechos ARCO

Se debe facilitar a los ciudadanos el ejercicio de los denominados derechos ARCO (acceso, rectificación, cancelación y oposición). Para ello se deben diseñar procedimientos y formularios adecuados que, por un lado, faciliten al ciudadano solicitar el ejercicio de sus derechos, y por otro lado, permitan a la empresa contestar a las solicitudes en los plazos legalmente establecidos.

- Transferencia internacional de datos

Cuando se tenga la intención de transmitir los datos de carácter personal a países que no formen parte del Espacio Económico Europeo se deberán tener en cuenta, al menos, los siguientes aspectos:

- Notificación. Las transferencias internacionales de datos se deben notificar a la Agencia Española de Protección de Datos para su inscripción en el Registro General de Protección de Datos.

- Autorización previa. La transmisión de datos fuera del Espacio Económico Europeo requiere de autorización previa del Director de la Agencia Española de Protección de Datos y sólo podrá otorgarla si se obtienen las garantías adecuadas. No será necesaria dicha autorización si la transferencia se realiza bajo alguna de las excepciones previstas en el artículo 34 de la LOPD.

- Requisitos específicos para la comunicación o cesión de datos. Cuando la transferencia internacional de datos se realice con el objeto de entregar los datos a un tercero para que este los trate por su cuenta, además de los requisitos generales de notificación y autorización previa, será necesario aplicar las garantías descritas en el punto "Comunicación de datos".

- Requisitos específicos para el acceso a los datos por cuenta de terceros. Cuando la transferencia internacional de datos se realice para que un tercero preste un servicio al responsable del fichero, además de los requisitos generales de notificación y autorización previa, será necesario aplicar lo legalmente previsto en el punto "Acceso a los datos por cuenta de terceros".

Anexo III: Manual de usuario

Con el objetivo de que el usuario tenga todas las funcionalidades activadas, la mejor opción sería empezar por el registro. Para ello se debe hacer click en el botón "Register". A continuación aparece una pantalla en la que se solicitan algunos datos al usuario para completar el registro y, después de completar estos datos, se debe hacer click en "Ok".

Una vez registrado, para identificarse ante el sistema, el usuario debe hacer click en el botón "Log in", momento en el que el sistema mostrará una pantalla en la que el usuario deberá introducir su nombre de usuario y su contraseña, y hacer click en "Ok".

Una vez identificado ante el sistema, el usuario puede acceder al juego de dos formas. La primera forma es pulsando jugar, en la que se carga una fase que estuviera guardada por el usuario, o si no hay un juego guardado, carga la primera fase. La segunda forma es pulsando en seleccionar mundo, elegir un mundo, elegir una fase y el sistema cargaría esta fase.

Estando ya en la pantalla de juego, los controles básicos son la flecha derecha, la flecha izquierda y la barra espaciadora. La barra espaciadora se utiliza sólo para iniciar el juego, y las flechas mueven a la derecha o izquierda la barra de juego.

Mientras el juego esté en curso, el usuario puede pulsar la tecla "P", para pausar el juego, y se mostrará el menú de pausa, en el que se pueden hacer tres cosas. En primer lugar, pulsando la tecla "R", se puede resumir el juego y volver a jugar. En segundo lugar, pulsando la tecla "S", se puede guardar el juego, para seguir jugando en otro momento. En tercer lugar, pulsando la tecla "E", se sale al menú principal del sistema, sin guardar el juego.

Por último, para ver las mejores puntuaciones registradas en el sistema, el usuario debe hacer click en el botón "Ranking" que aparece en el menú principal, y se mostrará una pantalla con una tabla en la que aparecen los diez mejores marcadores.

Bibliografía

[1] Ivar Jacobson, Grady Booch y James Rumbaugh. *El Proceso Unificado de Desarrollo de Software*. Pearson Addison-Wesley 1999.

[2] Vaughn, Larry T. *Client/Server system design and implementation*. McGraw-Hill 1994.

[3] Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. *Patrones de Diseño: Elementos de software orientado a objetos reutilizable*. Addison-Wesley 2003.

[4] Christian W. Dawson, Gregorio Martín Quetglás. *El proyecto fin de carrera en ingeniería informática: una guía para el estudiante*. Prentice Hall 2002.

[5] Gregory M. Horine. *Manual imprescindible de gestión de proyectos*. Anaya Multimedia 2009.

[6] Pérez, Santos. *Cómo elaborar y presentar un trabajo escrito*. Deusto 1993.

[7] Fundación Wikimedia, *Wikipedia: The Free Encyclopedia*, http://www.wikipedia.org/, 2013.