**FrozenWars**

**Arquitectura**

**Fecha: 15/01/2013**

**Autor: María Díaz Calvo, Ricardo Javier Carrión Beltrán, Diego Arranz García**

**Versión: 1.0**



**TABLA DE VERSIONES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor/es** |
| 15/01/2013 | v1.0 | Creación del documento | María Díaz Calvo, Ricardo Javier Carrión Beltrán, Diego Arranz García |

**Descripción del documento**

El propósito de este documento es proporcionar un único punto de referencia sobre el apartado de la arquitectura del juego FrozenWars.

**Propósito**

El Documento de Arquitectura del Software proporciona una visión arquitectónica general del sistema, utilizando diferentes vistas arquitecturales para representar diferentes aspectos del sistema. Su objetivo es captar y transmitir las decisiones importantes de arquitectura que se han hecho en el sistema

**Ámbito**

El ámbito de este documento es definir la arquitectura, el diseño general y de cada uno de los módulos y paquetes que lo componen. La arquitectura servirá de base y como referencia para la implementación de la lógica de juego.

**Referencias**

[Catálogo de Requisitos](https://docs.google.com/document/d/1PZ-SD95kaErf6ukcXG0xRvkerYHbBDGeyUH-KbkdWMY/edit)

En él se detallan los requisitos necesarios para que la aplicación pueda cumplir la garantía de calidad.

[Librería libgdx](https://docs.google.com/document/d/13wQm4gu44DJBlhtOPz3c5CrJ0V3AjHKs55PLH8fF3Rw/edit)

En este documento se ofrecen detalles de la potencia de la librería libgdx a la hora de desarrollar aplicaciones para android

[SmartFoxServer](https://docs.google.com/document/d/11n_2en9W4RmHpGvxbJM-LsS1pCTlWiCkZwPrjEBiX_c/edit)

En este documento se ofrecen detalles de la librería SmartFoxServer la cual gestiona todo tipo de información sobre el servidor.

[Casos de uso](https://drive.google.com/?tab=wo&authuser=0#folders/0BwRUj9DWfmFmSk1xSFFEbUdSUGs)

En esta carpeta se encuentran todas las descripciones detalladas sobre los casos de uso.

**Descripción General**

Con el objetivo de documentar todos los aspectos de la arquitectura, el documento de arquitectura del software está dividido en las siguientes secciones:

* **Representación de la Arquitectura:** Describe el enfoque de cada una de las vistas.
* **Objetivos y restricciones de la arquitectura:** Describe las restricciones de las arquitectura del juego y de las librerías.
* **Vista de casos de uso:** Representación los distintos casos de uso de la aplicación y su descripción.
* **Vista Lógica:** Descomposición de los paquetes de UML y descripción de los diversos paquetes, funcionalidades y clases que lo componen.
* **Vista de procesos:** Descomposición del sistema en procesos ligeros o hilos.
* **Vista de despliegue:** Representación de las redes que participan en la aplicación.
* **Vista de implementación:** Describe el tipo de ejecutable que se genera, y qué contiene el mismo.
* **Calidad:** Describe los aspectos de calidad.

**Representación de la Arquitectura**

Para representar la arquitectura a lo largo del documento, se emplearán las siguientes vistas:

* **Vista de casos de uso:** representa los casos de uso que implementa esta arquitectura.
* **Vista lógica:** muestra cómo se distribuye la funcionalidad del sistema en la arquitectura, su división en paquetes y subpaquetes, las clases significativas y sus funcionalidades.
* **Vista de proceso:** muestra el sistema desde el punto de vista de sus procesos ligeros o hilos.
* **Vista de despliegue:** se representan los diferentes nodos de red en el que el software se ejecuta, así como los nodos de computación físicos que ejecutan el software y sus conexiones.
* **Vista de implementación:** diagramas que muestran cada elemento significativo del sistema y cómo se relacionan entre ellos.

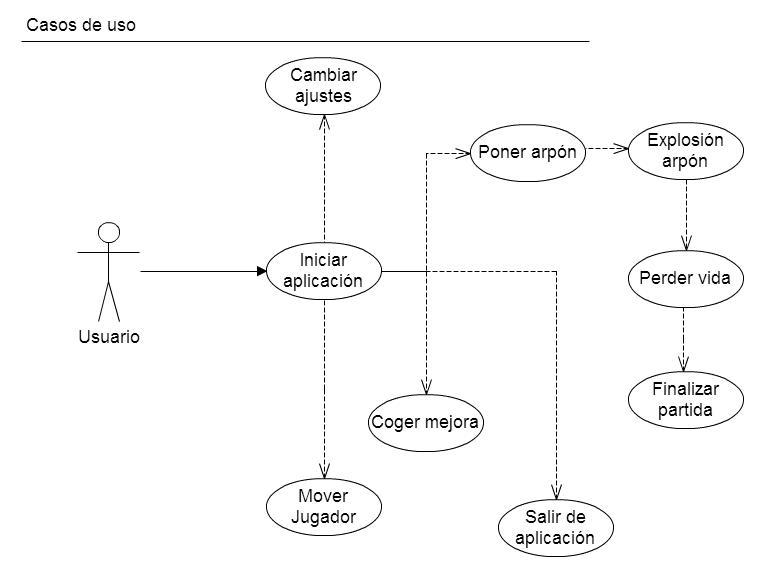
**Objetivos y restricciones de la arquitectura**

Para el diseño de la arquitectura, se tendrán en cuenta los siguientes objetivos:

* **Portabilidad en Android:** FrozenWars podrá ser ejecutada en la mayoría de versiones de Android(2.3.3 Gingerbread- 4.2.2 Jelly Bean).
* **Uso eficiente de la red:** ver requisitos funcionales en el documento Catálogo de requisitos[1]
* **Extensibilidad:** se pretende que los desarrolladores puedan ser capaces de ampliar el contenido del juego de forma sencilla.
* **Modularidad:** El proyecto ha de estar formado por módulos independientes y autocontenidos.
* **Reutilización:** Al tratarse de un juego, es más que probable que haya muchas tareas repetitivas asociadas con cálculos y tareas gráficas. Se procurará agrupar esas funciones de manera que puedan ser reutilizadas desde cualquier punto del sistema.

**Vista de Casos de uso [3]**

En esta sección del documento se realiza una especificación de los casos de uso y los escenarios que componen el videojuego FrozenWars y tienen un impacto directo en la arquitectura del mismo.



**Realizaciones de casos de uso**

A continuación se hará una breve descripción de cada uno de los casos de uso:

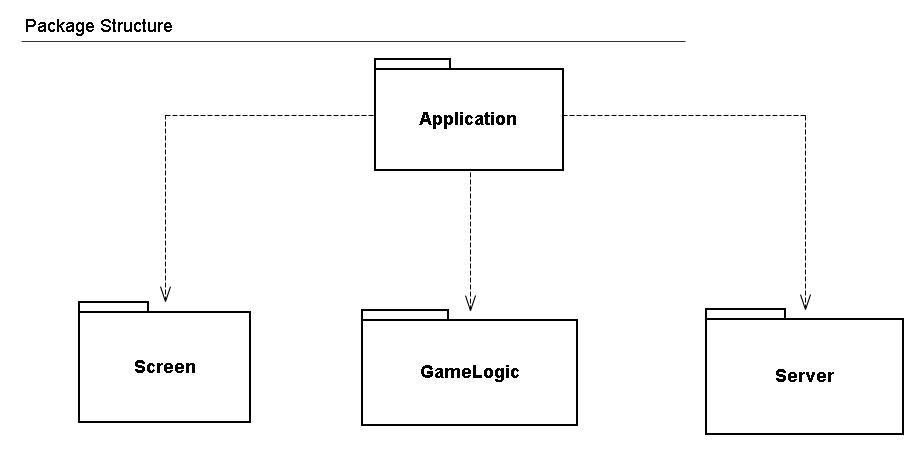
* **Iniciar aplicación:** Al abrir la aplicación aparece una pantalla con una imagen de la aplicación y después lleva al menú principal.
* **Mover jugador:** Al pulsar cualquiera de los botones de dirección, si las reglas del juego lo permiten, el jugador se mueve en la dirección correspondiente.
* **Cambiar ajustes:** Ver y cambiar los ajustes de la aplicación (vibración, sonido, confirmación al salir y borrar información de usuario y contraseña).
* **Coger mejora:** Cuando un jugador se sitúa encima de una mejora, ésta desaparece del mapa y se añade a las mejoras del jugador.
* **Poner bomba:**Al pulsar el botón correspondiente, si las reglas del juego lo permiten, se coloca un arpón en la posición actual del jugador.
* **Explosión bomba:**Después de 3 segundos de poner la bomba, la bomba explota causando diversos efectos en el juego o esta explota debido a un encadenamiento
* **Perder vida:** Si un personaje es alcanzado por el rango del hundimiento del hielo éste pierde una vida
* **Finalizar partida:**La partida actual es finalizada
* **Salir de aplicación:** El usuario decide salir de la aplicación y se desconecta del servidor y libera recursos del móvil

**Vista Lógica**

En esta sección se describen las partes principales del Modelo de Diseño: subsistemas, paquetes y su descomposición de cada paquete en clases, indicando sus responsabilidades y funcionalidad principal.

**Resumen**

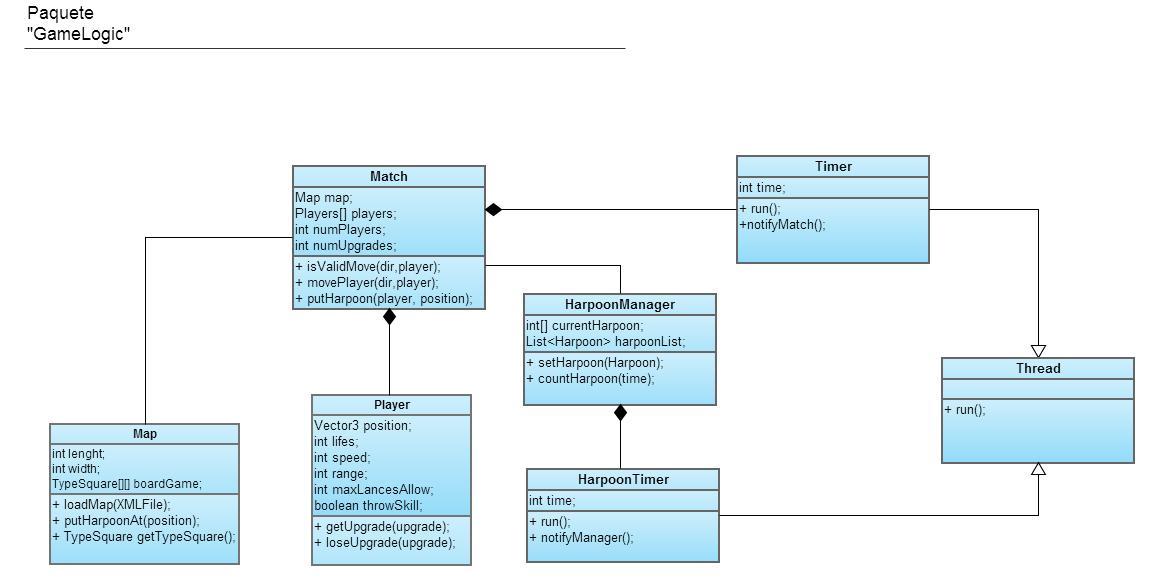
Para el desarrollo del juego se utiliza el MVC(Modelo-vista-controlador) integrándose con la librería libgdx[2] y SmartFoxServer[3]. En este MVC **“Application”** representa el controlador, **“Screen”** la vista y **“GameLogic”** el modelo. Además se ha modificado ligeramente para que además de controlar la lógica del juego, haga como intermediario entre el **“Server”** (servidor) y la lógica de juego para un desarrollo más modular.



**Paquetes significativos de la arquitectura**

El paquete Application implementa el controlador del modelo MVC. Principalmente sus funciones son el de ser intermediario entre eventos del servidor y el modelo(GameLogic), y enviar peticiones a la vista para mostrar cambios gráficamente.

* **Assets:** La función de esta clase es contener todas las imágenes y animaciones que son utilizadas en la aplicación.
* **Desktop:** La función de esta clase es lanzar la aplicación.
* **GameSettings:** La función de esta clase es cargar y guardar los ajustes del juego.
* **MatchManager:** La función de esta clase es organizar el juego, haciendo de enlace entre varias clases.
* **OpenGame:** La función de esta clase la de cargar las imágenes y establecer la página de inicio.



El paquete Screen implementa la vista del modelo MVC. Su función es manejar la parte gráfica del juego. Este paquete integra la librería libgdx facilitando la parte gráfica de forma de que sea mucho más cómoda de desarrollar la interfaz del juego e integración de animaciones y imágenes. Cada clase en este paquete es una “pantalla” y el controlador va alternando dichas pantallas.

* **ConfirmScreen:** La función de esta clase mostrar una pantalla de confirmación de finalización del juego.
* **GameScreen:** La función de esta clase es mostrar una pantalla donde aparece el desarrollo de la partida en curso.
* **InitialScreen:** La función de esta clase es mostrar el menú principal del juego.
* **LoadScreen:** La función de esta clase es mostrar una pantalla inicial cuando se carga el juego.
* **PenguinAnimation:** La función de esta clase es guardar toda la información acerca de las animaciones de cada jugador.
* **SettingsScreen:** La función de esta clase es mostrar una pantalla con los ajustes, y guardar la información relacionada con ellos.

El paquete GameLogic implementa el modelo en el modelo MVC. Su función es controlar toda la lógica del juego, como posiciones de jugadores, vidas, arpones, mapas etc

* **Harpoon:** La función de esta clase es crear los arpones guardando su posición y su rango.
* **HarpoonTimer:** La función de esta clase es implementar un thread para que funcione como un contador para calcular la explosión del arpón.
* **HarpoonManager:** La función de esta clase es controlar el número de arpones que hay en la partida actual, cual es el siguiente que se tiene que estallar y todo lo relacionado con eventos de los arpones.
* **MatchTimer:** La función de esta clase es implementar un thread el cual se encarga de contar el tiempo de juego y en función de ese tiempo, desplegar eventos.
* **Map:** La función de esta clase es guardar toda la información sobre el mapa de la partida en curso.
* **Player:** La función de esta clase es guardar la información del jugador.
* **Match:** La función de esta clase es guardar la información de la partida.

**Vista de procesos**

Esta sección describe la descomposición del sistema en procesos ligeros (hilos simples de control) y procesos pesados ​​(agrupaciones de procesos ligeros). Organización la sección de grupos de procesos que se comunican o interactúan entre sí. Describir las principales vías de comunicación entre procesos, como por ejemplo el paso de mensajes, alarmas, y cita.

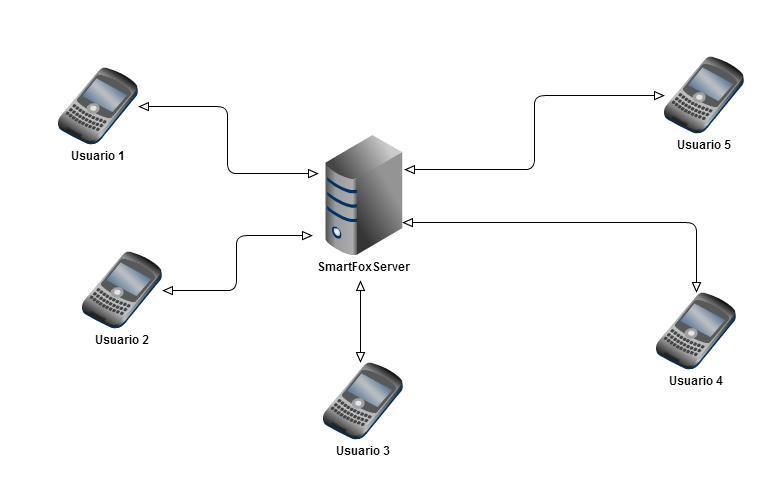
Además del hilo principal de juego, existen hilos encargados de tareas auxiliares para garantizar la consistencia y sincronización, así como el procesado de eventos.Los diferentes hilos son:

* **Hilo de ejecución**: Se ejecuta en la instancia del jugador que crea la partida. Envía a intervalos de tiempo su estado actual al servidor para que se sincronicen con él los demás usuarios.
* **Hilo de timer de partida**:Se trata de un contador del tiempo de la partida, es utilizado en la modalidad Battle Royal y su función es que cumplido el tiempo de partida se finaliza.
* **Hilo de timer para arpones**: Su función es contabilizar el tiempo que llevan puestos los arpones.

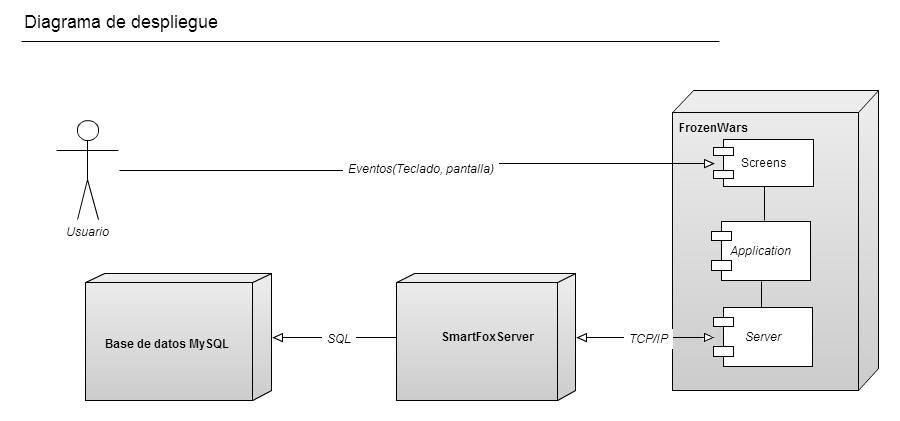
Existe además un caso especial, que es el SmartFoxServer, que no es un hilo en sí pero es tratado como tal. El SmartFoxServer, es un servidor independiente de la aplicación por lo que es otro hilo más a tener en cuenta por la aplicación. Para controlar los eventos del servidor, existe una clase (Server) que implementa un listener para eventos de SmartFoxServer. Esta se comunica,como se describió anteriormente, con el controlador para que este gestione dichos eventos y dependiendo del evento puede haber modificaciones en la lógica del juego. Por lo que el controlador tiene la responsabilidad de ser el intermediario entre la lógica de juego y el servidor.

**Vista de desarrollo:**

Esta sección describe una o más redes físicas y su configuración en la que se despliega el software a ejecutar. Para el correcto funcionamiento de la aplicación FrozenWars se necesita conexión a internet para poder establecer conexión con el servidor y así poder jugar con otros usuarios. Lo recomendable es utilizar una red que ofrezca poco retardos con el servidor como por ejemplo(Wifi). Si se utilizan redes como 3G se puede establecer conexión en el chat del juego, sin embargo al ser mayor el retardo con este tipo de red, la experiencia de juego puede ser alterada ya que es probable que haya numerosos paros en el juego debido al poco ancho de banda de este tipo de redes.

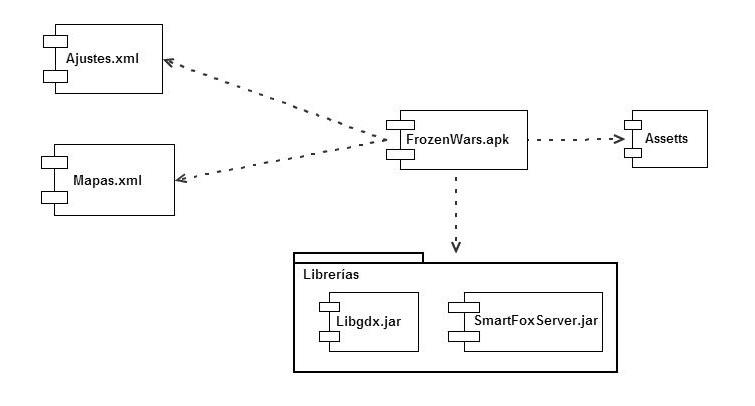


El diagrama de funcionará de la siguiente manera: El usuario se comunica con el programa mediante la pantalla(*Touch Events*) o mediante la entrada de teclado. Después la pantalla manda petición al controlador para que gestione dichos eventos. A su vez éste, mediante la interfaz del servidor(clase Server) envía al servidor los eventos correspondientes. El servidor gestiona dichos eventos consultando la base de datos o haciendo un ‘*broadcasting’* del evento y con ello, hacérselo llegar a varios usuarios. En este caso, la clase Server se encargaría de hacer una petición al Application para que gestione dichos eventos y hacer cambios en las Screens para representar de manera gráfica al usuario la petición que hizo mediante la entrada de pantalla o teclado.



**Vista de implementación**

Los componentes del sistema se ofrecen empaquetados en un solo archivo ejecutable en sistemas android con extensión apk, que recogerá toda la funcionalidad(código) recursos de la aplicación (imágenes,sonidos, animaciones, todo ello incluido en Assets) y archivos xml que contienen los mapas y los ajustes.



**Calidad**

Se proponen ciertas características de sistema a las cuales el diseño de la arquitectura debe acercarse lo máximo posible:

Extensibilidad

Se refiere a la capacidad del sistema para ofrecer una facilidad al

ampliar la aplicación con nuevos elementos enfocados a la jugabilidad o comportamiento

de los diferente elementos del juego.

Como ejemplos propuestos para la extensibilidad de la aplicación:

* Introducir la opción de guardar los rankings de puntuaciones.
* Introducir una aplicación de construcción de mapas personalizados, la cual será perfectamente compatible con la aplicación.
* Introducir un modo un jugador en el que haya rivales controlados por IA.
* Incorporar la posibilidad de jugar contra rivales en modo multijugador manejados por IA de la máquina, en cualquiera de los modos.

Portabilidad

Por estar implementado en Java, la mayoría de componentes del sistema son susceptibles de ser desplegados en cualquier plataforma, lo cual es interesante para una hipotética versión para otras plataformas como Windows, Mac o Linux.

Eficiencia de la red

La aplicación se encargará de gestionar adecuadamente el tráfico de la red, de forma que minimice el uso de esta y haciendo así un uso eficiente de la red.