UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

DISEÑO E IMPLEMENTACIÓN DE UN VIDEOJUEGO PARA DISPOSITIVOS MÓVILES ANDROID:CASTLE & GOBLINS

Proyecto Final de Carrera

[Ingeniería Técnica en Informática de Gestión]

Autor: Víctor Aragó garcía

Director: Juan Vicente Capella Hernandez

30/09/2014

Resumen

Castle & Goblins es un videojuego multijugador masivo online en tiempo real. El juego va a estar implementado para dispositivos móviles Android. La parte cliente del juego estará desarrollada en Java y utilizando el SDK de Android. Esta parte a su vez puede ser dividida en dos capas, lógica de la aplicación y interfaz gráfica. La interfaz gráfica estará se compone en su mayor parte de layouts xml de Android y una serie de recursos gráficos en formatos comunes como PNG y JPG. La parte de lógica estará dividida en objetos y clases, de acuerdo con lo aprendido en distintas asignaturas de programación de la carrera.

Por otra parte, el servicio web que se encarga de parte de la lógica del servidor (más extensa que la de la aplicación) y toda la parte de persistencia de datos. Esta parte está alojada en un servidor web. Este servidor web contiene versiones recientes de Apache, MySQL y PHP para el funcionamiento del juego. De esta manera toda la parte del servidor estará programada en lenguaje PHP. La persistencia de los datos se llevará a cabo mediante una base de datos MySQL y la a comunicación entre cliente y servidor, se realizara mediante envío y recepción de mensajes codificados en JSON. Adicionalmente y para dar un poco más de valor al juego, se creará una pequeña web donde los jugadores podrá encontrar información del juego, consultar los rankings y interactuar entre ellos en un foro público.

La temática del juego está dentro de un mundo de fantasíca medieval donde el jugador debe tomar el rol de un jefe goblin. Este jefe deberá gestionar sus recursos para conseguir más victorias en sus batallas contra otros clanes (jugadores). Estas victorias o derrotas le posicionarán en un ranking respecto a los otros.

Palabras clave: android, juego, videojuego, cliente-servidor, aplicación

Tabla de contenidos

Contenido

1. Introducción

Este proyecto trata sobre un videojuego para dispositivos Android, nació unos meses antes de la redacción de esta memoria y ha ido evolucionando hasta lo que es ahora. Llegados a este punto de desarrollo, he decidido terminarlo y presentarlo como PFC con el título de “Castle & Goblins”.

Castle & Goblins es un videojuego multijugador online en tiempo real ambientado en un mundo de fantasía medieval. Es multijugador ya que todos los dispositivos estarán conectados entre sí y todos los jugadores rivalizarán para ser los mejores. Castle & Goblins es un videojuego en tiempo real porqué cualquier acción que lleve a cabo un jugador en su dispositivo puede afectar a los otros jugadores de forma inmediata. Es un juego donde todos los jugadores interactúan 24 horas al día estén conectados o no.

El juego se divide en partidas que tienen una determinada duración. Cada jugador asume el rol de un caudillo goblin. El objetivo del juego es intentar conseguir la mayor puntuación para cada partida. Cada jugador comienza en su castillo con un único soldado goblin. A medida que pasa el tiempo el jugador va obteniendo monedas de oro. Este oro puede utilizarse para comprar más soldados o mejorar el castillo para aumentar la producción de oro. A partir de este momento cada jugador podrá elegir entre defenderse en su castillo para resistir mejor los ataques enemigos o bien atacar para conseguir puntos de victoria. Para darle algo más de jugabilidad a las partidas, se han introducido una serie de objetos de un solo uso que los jugadores podrán utilizar para desequilibrar la balanza y poder aventajarse a sus oponentes.

1.1 Objetivos

El proyecto nació con el fin de aprender y trabajar en la mayor cantidad de tecnologías posibles que exige el desarrollo de cualquier aplicación móvil de hoy en día.

En primer lugar se va a desarollar un a aplicación completa para Android, con todo lo que esto conlleva. La aplicación será capaz de identificar al usuario según sus credenciales mediante una petición al

servidor. Y una vez identificado esta le permitirá jugar una partida dónde podrá navegar por su interfaz para realizar las diferentes acciones que puede hacer el jugador. Estas acciones serán transmitidas en todo momento al servidor para que queden registradas y sigan la lógica del programa. Se realizará el modelado de objetos necesarios para la lógica del programa funcione correctamente. En esta parte se estudiará tanto la estructura de archivos de los proyectos Android como la creación de layouts para representar las pantallas del juego. También se estudiarán los hilos de ejecución y la comunicación a través de Internet con el servidor mediante mensajes GET y respuestas en formato JSON.

La segunda gran parte del proyecto, es el desarrollo del servidor que manejará toda la gestión de datos de los usuarios y la mayor parte de la lógica. Se estudiará el funcionamiento del servidor Apache y su módulo de PHP. Toda esta parte del proyecto se realizará con PHP, por lo tanto se investigará mucho este lenguaje de programación, siendo el segundo lenguaje a aprender más importante utilizado también para la parte web que posteriormente explicaremos. En esta parte también estudiaremos la conexión mediante la base de datos en MySQL y el módulo PHP del servidor para poder recuperar en tiempo real los datos de cualquier jugador o partida. Se realizará el diseño de las tablas MySQL para que los datos estén ordenados y se pueda acceder a ellos de una manera lógica con el menor coste posible para el servidor de Bases de Datos.

En último lugar y como apoyo para la aplicación principal se va a habilitar una web con un foro. La web se va a desarollar con la plataforma Wordpress. Se estudiará esta plataforma y su gestión para poder publicar información adicional independientemente de la aplicación. Además se integrará un foro en el mismo espacio web para que los jugadores puedan participar de manera externa a la aplicación entre ellos o con los administradores. El foro se desarollará a partir de la plataforma PHPBB y también se tendrá que estudiar la administración y configuración del sitio. Estas dos plataformas están desarrolladas en PHP por lo que los conocimientos adquiridos en esta parte nos podrán ayudar al desarrollo del servidor y viceversa. También se estudiarán las bases de HTML5, CSS3 y JavaScript para adaptar el Front-End de estas webs.

1.2 Estructura de la memoria:

1.3 Justificación del proyecto

El mundo de los videojuegos ha cambiado con la llegada de los SmartPhones. Las características de estos dispositivos abren muchas posibilidades a nuevos tipos de juegos:

- Van siempre junto a su propietario.
- Disponen de conexión a Internet 24h
- Multitud de sensores (Acelerómetro, pantalla táctil, cámara, micrófono, GPS, brujula, …)
- Portabilidad
- etc...

En mi opinión estas características son las apropiadas para un tipo de juego como el Castle & Goblins, ya que es un juego que requiere conexión a Internet y requiere la intervención del jugador en cualquier momento del día.

Soy un amante de los videojuegos desde mi infancia. Ahora que tengo los conocimientos para poder desarrollar uno y me apasiona poder completarlo y poder compartirlo con otros jugadores.

El proyecto se ha creado sin ánimo de lucro y con el único fin de adquirir conocimientos sobre diferentes plataformas y lenguajes de programación. No se ha invertido ningún tipo de recurso más que el de tiempo de desarrollo propio. Tampoco se ha invertido en herramientas puesto que se han utilizado para todo, herramientas gratuitas como he comentado en el punto 1.3.

La mayoría de recursos gráficos e ilustraciones han sido creadas específicamente para este videojuego por “Pablo Brosseta Micó” quién me las ha cedido gratuitamente para este proyecto. Este ilustrador es un gran Amigo y compañero de trabajo y le agradezco su participación en este proyecto ya que sin su ayuda el aspecto visual de la aplicación no hubiese tenido la calidad necesaria.

El resto de recursos que han sido descargados de bancos de imágenes gratuitos como http://sxc.hu/

2. Tecnologías

2.1 Android

Ilustración 1

Android es el sistema operativo basado en Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como smartphones y tablets, y más tarde portado a otras plataformas como televisores, relojes, etc. Inicialmente desarrollado por Android Inc. desde 2005, que Google respaldó económicamente y más tarde compró. Este sistema operativo se anunció en 2007 y ha evolucionado mucho hasta ahora.

La estructura del sistema operativo se compone de aplicaciones que se ejecutan en un framework Java sobre una máquina virtual Dalvik. Las bibliotecas nativas que utiliza este sistema operativo están escritas en lenguaje C/C++ y Java. La mayor parte del código tiene una licencia libre y se puede consultar Las aplicaciones que vienen por defecto incluidas en el sistema operativo son un cliente de correo electrónico, un programa de SMS, calendario, mapas, navegador, contactos y una tienda en línea donde se pueden descargar nuevas aplicaciones para dotar de más funcionalidad al dispositivo. Estas aplicaciones se pueden empaquetar dentro de archivos apk para su distribución.

Como hemos comentado antes, el sistema operativo ha ido evolucionando con el tiempo y la última versión disponible es la 4.4 también llamada KitKat. En la siguiente tabla podemos ver las últimas

versiones, la fecha de lanzamiento y la cuota que han alcanzado en el último mes de Julio. Estos datos han sido obtenidos de la Wikipedia.

Versión	Nombre en código	Fecha de distribución	API level	Cuota (7 de Julio, 2014)
4.4	*Kit Kat*	31 de octubre de 2013	19	17,9%
4.3	*Jelly Bean*	24 de julio de 2013	18	9,0%
4.2.x	*Jelly Bean*	13 de noviembre de 2012	17	19,7%
4.1.x	*Jelly Bean*	9 de julio de 2012	16	27,8%
4.0.x	*Ice Cream Sandwich*	16 de diciembre de 2011	15	11,4%
3.2	*Honeycomb*	15 de julio de 2011	13	<0,%
2.3.3–2.3.7	*Gingerbread*	9 de febrero de 2011	10	13,5%
2.2	*Froyo*	20 de mayo de 2010	8	0,7%

Tabla 1

En este gráfico podemos ver cómo ha ido evolucionando la cuota de mercado respecto a los principales sistemas operativos para dispositivos móviles. El gráfico ha sido obtenido de la web http://www.muycomputer.com/

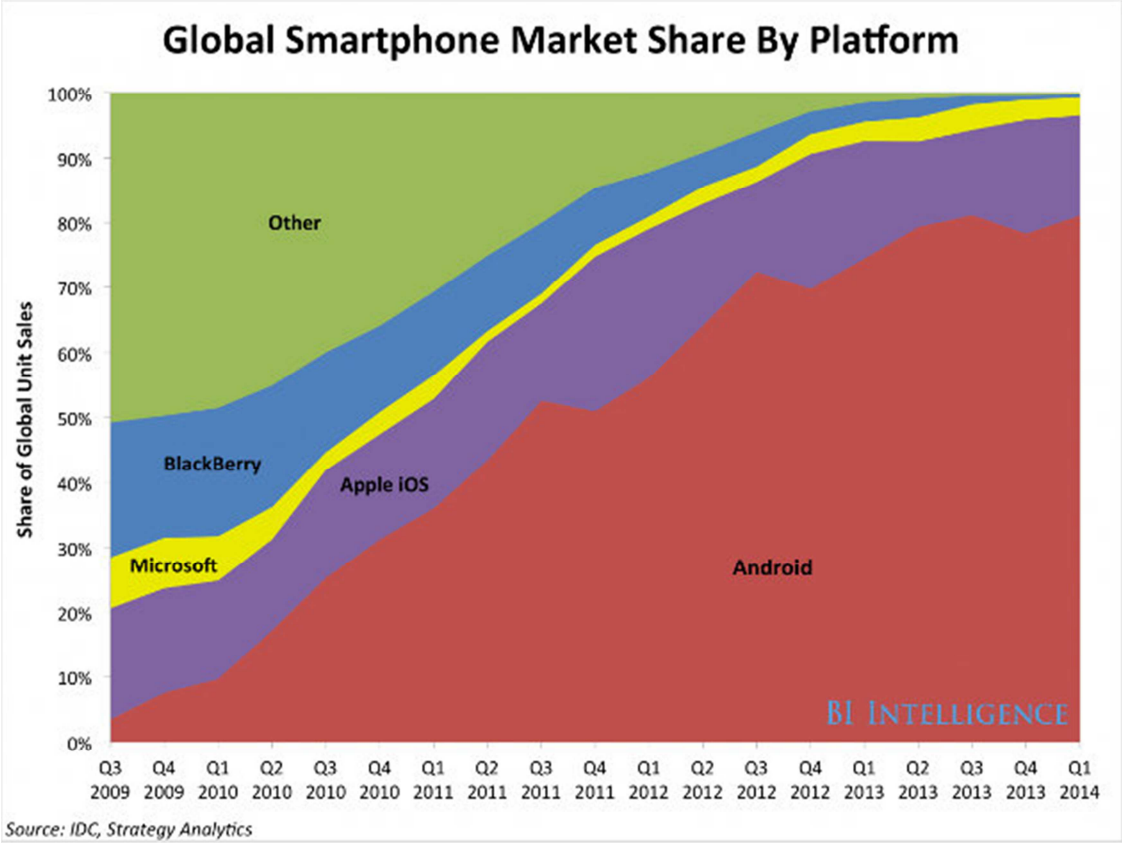


Ilustración 2

En nuestro caso, la parte cliente del videojuego de Castle & Goblins consta de una aplicación para este sistema operativo. Esta aplicación es la que se instala en el dispositivo y permite a los jugadores transmitir sus acciones al servidor.

2.1.1 Proyectos Android

Una parte muy importante de este proyecto y la que mayor tiempo de aprendizaje y desarrollo ha requerido, ha sido la parte del cliente móvil por eso se va a explicar con más detenimiento la estructura del proyecto de la aplicación cliente realizada con la herramienta Eclipse. Esta es la estructura de este proyecto.

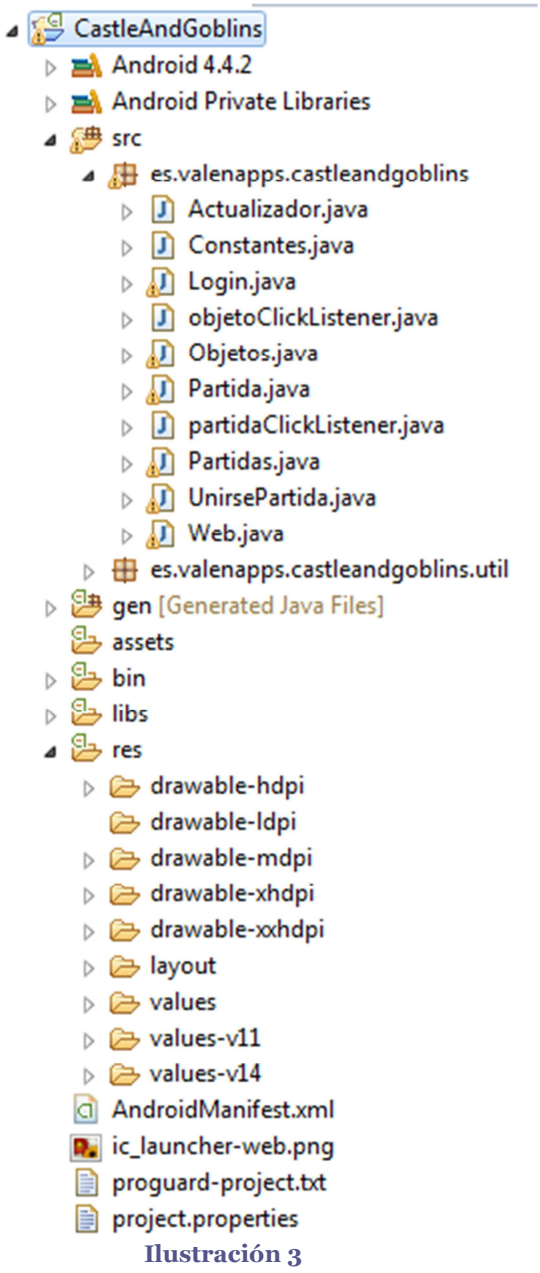


Ilustración 3

- src: Carpeta que contiene el código fuente de la aplicación. Aquí dentro se amlmacenan todos los ficheros de nombre en un espacio de nombres. Este espacio de nombres es lo que se llama también nombre del paquete y es muy importante ya que es el identificador de nuestra aplicación dentro de la tienda de google y dentro de los dispositivos.

- gen: Carpeta que contiene el código generado de forma automática por el SDK. Nunca hay que modificar de forma manual estos ficheros. Dentro encontraremos:

  - BuildConfig.java: Define la constante DEBUG para que desde Java puedas saber si tu aplicación está en fase de desarrollo.

  - R.java: Define una clase que asocia los recursos de la aplicación con identificadores. De esta forma los recursos podrán ser accedidos desde Java.

-

- Android x.x: Código JAR, el API de Android según la versión seleccionada.

- Android Private Libraries: Librerías asociadas al proyecto.

- assets: Carpeta que puede contener una serie arbitraria de ficheros o carpetas que podrán ser utilizados por la aplicación (ficheros de datos, fuentes,…). A diferencia de la carpeta res, nunca se modifica el contenido de los ficheros de esta carpeta ni se les asociará un identificador. En el caso de este proyecto no ha sido necesaria la utilización de esta carpeta.

- bin: En esta carpeta se compila el código y se genera el .apk, fichero comprimido que contine la aplicación final lista para instalar.

- libs: Código JAR con librerías que quieras usar en tu proyecto. Se ha añadido automáticamente la librería android-support-v4. Su objetivo es permitir ciertas funcionalidades importantes no disponbles en el nivel de API seleccionado como mínimo.

- res: Carpeta que contiene los recursos usados por la aplicación. Las subcarpetas pueden tener un sufijo si queremos que el recurso solo se cargue al cumplirse una condición. En este caso hemos utilizado sufijos para cargar recursos con diferente densidad de pantallas (hdpi, ldpi, mdpi, xhdpi y xxhdpi) y también sufijos para diferenciar las versiones

de la api utilizada (v11 y v14). En la carpeta res podemos encontrar:

  - drawable: En esta carpeta se almacenan los ficheros de imágenes (JPG o PNG) y descriptores de imágenes en XML.

  - layout: Contiene ficheros XML con vistas de la aplicación. Las vistas nos permitirán configurar las diferentes pantallas que compondrán la interfaz de usuario de la aplicación.

  - values: También utilizaremos ficheros XML para indicar valores del tipo string, color o estilo. De esta manera podremos cambiar los valores sin necesidad de ir al código fuente. Por ejemplo, nos permitiría traducir una aplicación a otro idioma en un futuro de manera más fácil.

  - xml: Otros ficheros XML requeridos por la aplicación.

- AndroidManifest.xml: Este fichero describe la aplicación Android. En él se indican las actividades, intenciones, servicios y proveedores de contenido de la aplicación. También se declaran los permisos que requerirá la aplicación. Se indica la versión mínima de Android para poder ejecutarla, el paquete Java, la versión de la aplicación, etc.

- ic_launcher-web.png: Icono de la aplicación de gran tamaño para ser usado en páginas Web. El nombre puede variar si se indicó uno diferente en el proceso de creación del proyecto. Ha de tener una resolución de 512x512 (con alfa).

- proguard-project.txt: Fichero de configuración de la herramienta ProGuard, que te permite optimizar y ofuscar el código generado. Es decir, se obtiene un .apk más pequeño y donde resulta más difícil hacer ingeniería inversa.

- project.properties: Fichero generado automáticamente por el SDK. Nunca hay que modificarlo. Se utiliza para comprobar la versión del API y otras características cuando se instala la aplicación en el terminal.

2.2 Servidor Apache

Ilustración 4

2.2.1 Introducción

El servidor HTTP Apache es un servidor web HTTP de código abierto, disponible para las plataformas más comunes como Windows, Unix y Macintosh, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Las ventajas que presenta son que es un servidor modular al que se le pueden añadir muchas extensiones. Es un software de código abierto y tiene mucha documentación al respecto ya que es muy popular.

2.2.2 mod_php

Mod_php es un módulo que permite a apache interpretar archivos escritos en lenguaje PHP. Utilizando CGI Apache lanza un proceso interno de PHP con el que se comunica para ejecutar el código.

El videojuego Castle & Goblins hace uso de este módulo para ejecutar la mayor parte de su funcionalidad de la parte del servidor que está escrita en PHP.

2.3 MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

Para el uso que vamos a dar en Castle & Goblins, en la primera fase que es un producto creado únicamente con fines educativos, podemos utilizarlo bajo la licencia GNU GPL.

Este sistema es muy importante en el proyecto ya que se ocupa de todo el almacenamiento de datos de todos los usuarios. Teniendo como pieza central una base de datos con varias tablas donde se consultan los datos en tiempo real para hacer posible la conectividad entre todos los jugadores.

Por otra parte, este proyecto incluye un foro para los jugadores y una página web de información dónde los usuarios pueden consultar las reglas del juego, actualizaciones etc. Estas dos plataformas PHPBB y WordPress respectivamente, tienen asociadas una base de datos MySQL para su funcionamiento. En estas bases de datos se guarda la información sobre los usuarios, publicaciones y la mayoría de los contenidos de las mismas.

2.4 Wordpress

Ilustración 5

Wordpres, es el sistema elegido para nuestra web de información, donde los usuarios podrán obtener toda la información del juego y estar al corriente de las actualizaciones y estados de los servicios.

WordPress es una avanzada plataforma semántica de publicación personal orientada a la estética, los estándares web y la usabilidad. WordPress es libre y, al mismo tiempo, gratuito.

Dicho de forma más sencilla, WordPress es un gestor de contenidos muy simple e intuitivo de utilizar. Tiene un panel de control donde de manera muy fácil y visual se pueden modificar casi todas las partes de la misma web. Debido a esto, es muy popular en la red y su uso se ha disparado en los últimos años.

Ha sido desarrollado en PHP para entornos que ejecuten MySQL y Apache, bajo licencia GPL y código modificable, y su fundador es Matt Mullenweg. WordPress fue creado a partir del desaparecido b2/cafelog y se ha convertido junto a Movable Type en el CMS más popular de la blogosfera y en el más popular con respecto a cualquier otro CMS de aplicación general.

Otro motivo a considerar sobre su éxito y extensión es la enorme comunidad de desarrolladores y diseñadores, encargados de desarrollarlo en general o crear complementos y temas para la comunidad. En agosto de 2013 era usado por el 18,9% de todos los sitios existentes en internet.

Wordpress en un principio es un sistema de publicación web basado

en entradas ordenadas por fecha, tipo blog, y estas además se clasifican por categorías. Además de esto, se pueden añadir páginas estáticas no cronológicas para completar el resto de la web.

Tiene un sistema de plantillas con el que se puede modificar el aspecto visual de toda la web, y se pueden instalar independientemente del contenido.

Además de todo lo que se ha comentado, WordPress dispone de una gran cantidad de plugins que se pueden añadir al código original y van a dotar la web de nuevas funcionalidades. Al ser esta, una plataforma tan extendida ha habido una gran cantidad de desarrolladores que ya han creado miles de plugins para casi cualquier cosa. Acutalmente (Agosto 2014) hay más de 30.000 plugins oficiales, disponibles para descargar y la mayoría de ellos son gratuitos. Algunos de estos plugins son del tipo Widget, y tienen un componente visual y funcional que podemos colocar en ciertas partes de la web y dan una funcionalidad extra.

2.5 PHPBB

Ilustración 6

PHPBB es un sistema de foros gratuito basado en un conjunto de paquetes de código programados en el popular lenguaje de programación web PHP y lanzado bajo la Licencia pública general de GNU, cuya intención es la de proporcionar fácilmente, y con amplia posibilidad de personalización, una herramienta para crear comunidades. Su nombre es por la abreviación de PHP Bulletin Board, también es conocido como Phpbb3 por su última versión.

Funciona sobre bases de datos basadas en el lenguaje SQL como MySQL, PostgreSQL, Microsoft SQL Server, otras como Microsoft Access y,

con una modificación, también sobre Oracle, donde almacena la información para poder recuperarla en cada petición del lenguaje. Aunque para nuestro foro hemos utilizado como ya hemos comentado una base de datos MySQL.

Como características principales se puede decir que tiene un administrador bastante fácil de utilizar aunque no es tan intuitivo como en el caso de WordPress, pero con un poco de práctica se puede manejar con facilidad. Soporta mensajes públicos y privados entre los usuarios. Tiene creación de encuestas para que los usuarios puedan responder independientemente. Permite definir roles para los usuarios y permitirles o denegarles el acceso a diferentes foros y/o funcionalidades. También permite la carga de archivos adjuntos a cada publicación. Separa la capa de presentación del resto mediante temas visuales al igual que WordPress. Y por último también se le pueden añadir funcionalidades extra mediante los llamados MOD's.

2.6 Lenguajes utilizados

2.6.1 Java

Ilustración 7

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una

plataforma no tiene que ser recompilado para correr en otra. En nuestro caso, corre sobre una máquina virtual instalada en los dispositivos, llamada la máquina Virtual Dalvik. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales, y librerías de clases en 1991 y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento con las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.

Hemos dicho que JAVA es un lenguaje de programación orientado a objetos, esto significa que todas las variables que vamos a almacenar van a ser objetos que imitan algún aspecto de la realidad a excepción de los tipos de datos básicos como enteros, decimales, caracteres y booleanos. Todos los demás objetos se construyen a partir de estos datos básicos. Estos objetos tienen son entidades que tienen un determinado estado, comportamiento e identidad dentro de nuestro programa.

Una de las grandes características de JAVA que lo diferencia de otros es que tiene un recolector de basura. Este recolector es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos pueden tener localizado un objeto mediante una referencia a éste. Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas. En definitiva, el recolector de basura de Java permite una fácil creación y

eliminación de objetos y mayor seguridad.

Este lenguaje de programación se ha utilizado para programar la mayor parte de la aplicación nativa de Android, es decir la aplicación cliente. Esta parte del proyecto se ha desarrollado en Eclipse junto con el ADT (Android Development Tools) de Android. Este programa permite la depuración en tiempo real mediante un dispositivo físico o virtual (viene incluido con el ADT).

Para el juego se han definido varios objetos de la clase Activity, que representan las diferentes pantallas de la aplicación. Esta parte se definirá más adelante en el apartado de Implementación.

2.6.2 PHP

Ilustración 8

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante.

Se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy. Lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico como Facebook, para optar por PHP como tecnología de servidor.

PHP puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. El lenguaje PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores. El enorme número de sitios en PHP ha visto

reducida su cantidad a favor de otros nuevos lenguajes no tan poderosos desde agosto de 2005. El sitio web de Wikipedia está desarrollado en PHP.5 Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

En este proyecto se ha utilizado PHP para crear toda la lógica del servidor. Como ya se ha comentado en la sección de Apache, el módulo mod_php es el encargado de interpretar este lenguaje. Esta parte se ha dividido en dos capas: la capa de datos que contiene todas las funciones encargadas de comunicarse con la base de datos MySQL y la parte de lógica que es la que contiene los diferentes algoritmos para la resolución de los eventos del juego.

Por otra parte el foro en PHPBB y la web de ayuda en WordPress como ya se ha dicho, también están implementados a partir de este lenguaje.

2.6.3 HTML5

Ilustración 9

HTML, siglas de HyperText Markup Language («lenguaje de marcas

de hipertexto»), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, entre otros.

El lenguaje HTML basa su filosofía de desarrollo en la referenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene sólo texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir o hacer referencia a un tipo de programa llamado script, el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTML5 es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. Todavía se encuentra en modo experimental, aunque ya es usado por múltiples desarrolladores web por sus avances, mejoras y ventajas. El desarrollo de este lenguaje de marcado es regulado por el Consorcio W3C.

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas que ya existían, pero tienen un significado semántico propio. En esta nueva versión se han eliminado algunas etiquetas centradas en el estilo de las que ya se encargan las llamadas hojas de estilo escritas en CSS.

2.6.4 CSS3

Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML. El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores al igual que para el HTML5.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser definida en un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo «style».

En la tercera versión de CSS se ha optado por dividir el desarrollo en módulos independientes para poder ser creados por separado. Todavía está en fase de desarrollo pero la mayoría de navegadores ya implementan casi todas sus funcionalidades. Esta versión permite el uso de muchas más características que su versión anterior.

2.6.5 JavaScript

JavaScript (abreviado comúnmente "JS") es un lenguaje de programación interpretado, esto significa que el código no se compila para producir código máquina sino que es interpretado línea a línea en este caso por el navegador.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor que no hemos utilizado para este proyecto.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

El uso más común de JavaScript es escribir funciones embebidas o incluidas en páginas HTML y que interactúan con el Document Object Model (DOM o Modelo de Objetos del Documento) de la página. Algunos ejemplos sencillos de este uso son:

- Cargar nuevo contenido para la página o enviar datos al servidor a través de AJAX sin necesidad de recargar la página (por ejemplo, una red social puede permitir al usuario enviar actualizaciones de estado sin salir de la página).
- Animación de los elementos de página, hacerlos desaparecer, cambiar su tamaño, moverlos, etc.
- Contenido interactivo, por ejemplo, juegos y reproducción de audio y vídeo.
- Validación de los valores de entrada de un formulario web para asegurarse de que son aceptables antes de ser enviado al servidor.
- Transmisión de información sobre los hábitos de lectura de los usuarios y las actividades de navegación a varios sitios web. Las páginas Web con frecuencia lo hacen para hacer análisis web, seguimiento de anuncios, la personalización o para otros fines.33

Dado que el código JavaScript puede ejecutarse localmente en el navegador del usuario (en lugar de en un servidor remoto), el navegador puede responder a las acciones del usuario con rapidez, haciendo una aplicación más sensible. Por otra parte, el código JavaScript puede detectar acciones de los usuarios que HTML por sí sola no puede, como pulsaciones de teclado.

Ilustración 10

Para este proyecto se ha utilizado la librería jQuery, que ha facilitado mucho las cosas sobre todo en temas de compatibilidad con distintos navegadores.

jQuery es una biblioteca de JavaScript, creada inicialmente por John

Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

La ventaja principal de jQuery es que es mucho más fácil que sus competidores. Además al ser tan popular existen muchos plugins creados por diversos desarrolladores que se pueden utilizar fácilmente, traduciéndose esto en un ahorro substancial de tiempo y esfuerzo. De hecho, una de las principales razones por la cual Resig y su equipo crearon jQuery fue para ganar tiempo (en el mundo de desarrollo web, tiempo importa mucho).

En el proyecto objeto de esta memoria, jQuery está presente tanto en la web de información montada a partir de WordPress como en la parte del foro creado con PHPBB. Lo que se ha incluido en estas webs es el fichero minimizado y optimizado que contiene la librería. Aunque se puede ver accediendo directamente al código fuente al ser esta versión optimizada probablemente no se pueda entender a primera vista. En la web oficial podemos descargarnos la versión de desarrollo donde podremos ver la misma librería bien estructurada, tabulada y llena de comentarios.

2.6.6 JSON

Ilustración 11

JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

Los entornos en el servidor normalmente requieren que se incorpore una función u objeto analizador de JSON. Algunos programadores, especialmente los familiarizados con el lenguaje C, encuentran JSON más natural que XML, pero otros desarrolladores encuentran su escueta notación algo confusa, especialmente cuando se trata de datos fuertemente jerarquizados o anidados muy profundamente.

En el proyecto este lenguaje de datos, ha sido fundamental para el intercambio de datos entre la aplicación cliente y el servidor central. El funcionamiento de este proyecto es el siguiente:

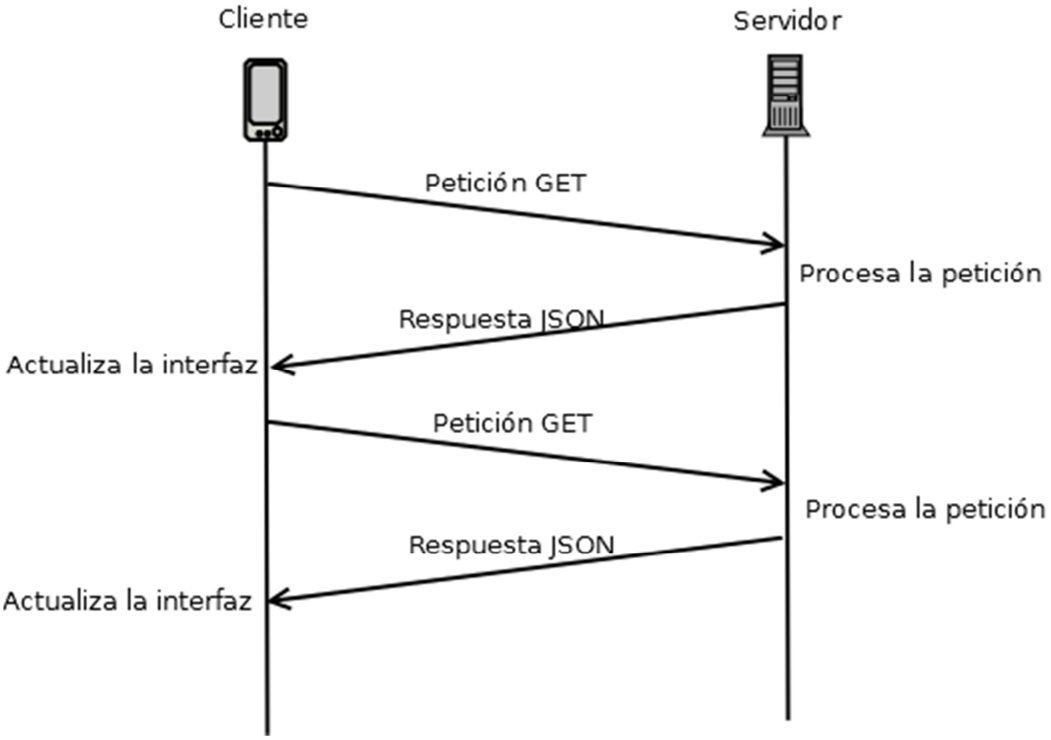


Ilustración 12

El jugador elige hacer una acción, esto se transmite mediante una petición GET al servidor. Este procesa la acción del usuario y devuelve un bloque JSON con la respuesta y todos sus datos actualizados como puede ser el tiempo, el oro acumulado, cantidad de goblins en la reserva, etc.

2.6.7 XML

XML, siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.

Como ya se ha comentado antes, el lenguaje que hemos utilizado para el intercambio de datos entre cliente y servidor es JSON, sin embargo debido a la estructura de proyectos de Android, hemos tenido que utilizar este lenguaje para la definición de los layouts de las vistas de la aplicación cliente. Es con este lenguaje con el que se define la vista de cada una de las pantallas de la aplicación de los jugadores. Este mecanismo se utiliza para separar la capa de negocio y almacenamiento de datos de la parte visual o estética de la aplicación.

Este lenguaje también es utilizado para definir las propiedades de la aplicación mediante un archivo llamado AndroidManifest.xml y es

utilizado por los dispositivos para saber qué tipo de actividades realiza la aplicación y que permisos se requieren al utilizarla.

2.7 Herramientas

Todas las herramientas utilizadas para el desarrollo del videojuego tienen una licencia libre de derechos y pueden ser encontradas y descargadas para ser utilizadas gratuitamente sin pagar ningún tipo de licencia.

Notepad++

Ilustración 13

Este es un editor de texto simple con multitud de ayudas para el programador. Es un programa de código libre que soporta múltiples lenguajes. Se distribuye bajo licencia GPL. Es un programa muy ligero que ayuda mucho al programador.
Las principales características son:

- Resaltado de sintaxis
- Interfaz personalizable
- Muy minimalista
- Auto-completado
- Multi-documento
- Permite el uso de marcadores

GIMP 2

Ilustración 14

Es un programa de manipulación de imágenes de mapa de bits, tanto ilustraciones, dibujos y fotografías. Forma parte del proyecto GNU y está disponible bajo la Licencia pública general de GNU.

Es el software de retoque fotográfico disponible en mas sistemas operativos. (Unix, Linux, Solaris, Microsoft Widnwos, Mac OS...)

GIMP permite trabajar mediante capas para poder modificar dada parte de la imagen de forma totalmente independiente a las demás. También pueden subirse o bajarse de nivel para facilitar el trabajo en la imagen y que no queden partes ocultas. El formato por defecto de los archivos guardados por GIMP es el xcf aunque las imágenes pueden ser guardadas en otros formatos más comunes como jpg, png o gif.

OpenOffice

Ilustración 15

LibreOffice es una suite ofimática libre de código abierto y distribución gratuita que incluye herramientas como procesador de textos, hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y bases de datos. Está disponible para las plataformas más comunes tales como Microsoft Windows, GNU/Linux y Mac OS X. El proyecto LibreOffice se creó como una bifurcación de OpenOffice en 2010.

Dia

Ilustración 16

Dia es un editor de diagramas con las herramientas necesarias para crearlos o modificarlos y muy fácil de utilizar. Incluye herramientas de dibujo para introducir diversos elementos del diagrama y poder editarlos individualmente. Los archivos que maneja son archivos .dia aunque permite exportar los diagramas en formatos de imagen más común como png.

Es un software gratuito y se distribuye bajo la licencia GPLv2.

Eclipse

Ilustración 17

Este software se un entorno de desarrollo integrado de código libre muy completo. Se distribuye bajo la licencia *Eclipse Public License* compatible con la licencia GNU GPL. Se distribuye junto con el SDK de

Android en http://developer.android.com/sdk/. El código está principalmente escrito en java aunque contiene partes escritas con otros lenguajes.

Eclipse dispone de un editor de texto con resaltado de sintaxis. La compilación es en tiempo real y permite la depuración de código en ejecución.

Eclipse emplea módulos (plug-ins) para proporcionar toda su funcionalidad. En mi caso, la versión que he utilizada viene con los siguientes módulos instalados que facilitan la programación para plataformas Android: Android DDMS, Android Developer Tools, Android Development Tools, Android Hierarchy Viewer y Android Traceview.

WAMPServer

Ilustración 18

WampServer es un conjunto de paquetes software que está compuesto por:

- Apache: Apache es el Servidor Web más utilizado, líder con el mayor número de instalaciones a nivel mundial muy por delante de otras soluciones como el IIS (Internet Information Server) de Microsoft. Apache es un proyecto de código abierto y uso gratuito, multiplataforma (hay versiones para todos los sistemas operativos más importantes), muy robusto y que destaca por su seguridad y rendimiento.
- PHP(acrónimo recursivo de PHP: Hypertext Preprocessor): es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. PHP se utiliza para generar páginas web dinámicas. Recordar que llamamos página estática a aquella cuyos contenidos permanecen siempre igual, mientras que llamamos páginas dinámicas a aquellas cuyo contenido no es el mismo siempre. Por ejemplo, los contenidos pueden cambiar en base a los cambios que haya en una base de

datos, de búsquedas o aportaciones de los usuarios, etc. En el caso del WampServer, lo que se inculye es un módulo para apache que es capaz de interpretar este lenguaje.

- MySQL: es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.
- PHPMyAdmin: es una herramienta escrita en PHP con la intención de manejar la administración de MySQL. Para ello ofrece una interfaz web donde mediante formularios se pueden gestionar bases de datos, usuarios, etc.

3. Análisis

Aquí se recogen todos los datos y características previas a la fase de diseño del juego, donde se han estudiado aspectos como la viabilidad, los objetivos y la idea en ámbito general del juego.

3.1 Selección de objetivos

3.1.1 Público objetivo

El juego va dirigido a grupos de jóvenes de entre 15 y 35 años. Aficionados a las nuevas tecnologías. Es un juego que no requiere mucho tiempo de aprendizaje ni tiempo de juego, pero si una cierta constancia a lo largo de la partida.

3.1.2 Objetivos Globales

- Adquirir experiencia en la producción de aplicaciones en plataformas Android
- Entretener a los jóvenes.
- Aumentar la capacidad estratégica del jugador.
- Fomentar competitividad entre amigos

3.1.3 Objetivos Locales

- Conseguir una buena gestión de los recursos
- Conseguir mejor puntuación que los demás jugadores
- Utilizar ciertos objetos para modificar el ranking a tu favor.

3.2 Género

El Castle & Goblins es un videojuego de estrategia. Este tipo de videojuegos requieren que el jugador ponga en práctica sus habilidades de planeamiento y pensamiento para poder conseguir la victoria. Como subgénero, podemos definirlo como videojuego en tiempo real ya que no existe el concepto de turno. Todos los jugadores juegan al mismo tiempo.

La ambientación del juego se basa en un mundo de fantasía medieval similar al descrito en los libros de J.R.R Tolkien, o la ambientación del famoso juego de Dragones y Mazmorras.

Juegos similares:

- Ogame: Juego de estrategia en tiempo real con ambientación futurista. Este juego se juega desde cualquier navegador web.
- Travian: Juego de navegador como el anterior, pero ambientado en la época histórica pre-medieval. El objetivo es construir una maravilla.

3.3 Especificación conceptual

3.3.1 Entrada y salida:

- Entradas: Sensor táctil del dispositivo.
- Salidas: Pantalla del dispositivo
- Linea temporal: Depende de las acciones de todos los jugadores.

3.3.2 Usabilidad

Se ha intentado utilizar iconos estandarizados, para representar las acciones más comunes. Los jugadores más expertos podrán empezar a jugar intuitivamente sin necesitar ninguna ayuda más. Todos los botones están a la vista y no se requieren más de tres toques para acceder a cualquier sitio.

Algunos ejemplos de botones:

		
Ataque	Defensa	Información

Tabla 2

Adicionalmente, se ha creado una web de ayuda (http://enacefio.es/goblins/blog/), para que los usuarios puedan informarse de todos los controles y el funcionamiento del juego.

3.3.3 Aprendizaje

Al igual que todos los juegos, la fase de aprendizaje se puede dividir en tres fases:

- Conocimiento del juego: en esta fase, es donde el jugador aprende cómo funciona el juego, para qué sirve cada botón y cuáles son las reglas del juego.
- Desarrollo de estrategias: una vez ya se conoce el funcionamiento del juego, el jugador comienza a plantearse y probar diversas estrategias para conseguir sus objetivos.
- Perfeccionamiento: Aquí el jugador ya es dónde va perfeccionando sus técnicas y habilidades para poder superar a sus competidores.

3.4 Pre-programación

Al desarrollar Castle & Goblins, he optado por utilizar el modelo incremental dada la magnitud del proyecto que he desarrollado en

solitario.

El proyecto se puede dividir en los siguientes incrementos.

Incremento 1: Juego básico

Versión jugable pero con muy pocas características. El jugador puede unirse a una partida y empezar a recibir monedas de oro para comprar soldados goblin o mejorar el castillo para la obtención de más monedas. El algoritmo de ataque está implementado para ser probado. En esta fase no es importante el aspecto visual y se implementa con formularios y botones básicos de Android.

Incremento 2: Implementación de la defensa.

Se añade la nueva funcionalidad que permite a los jugadores defenderse en el castillo, para poder ocasionar mayor número de bajas enemigas frente a un posible ataque. En esta fase se continúa con la interfaz de formularios y botones y se modifica el algoritmo de ataque.

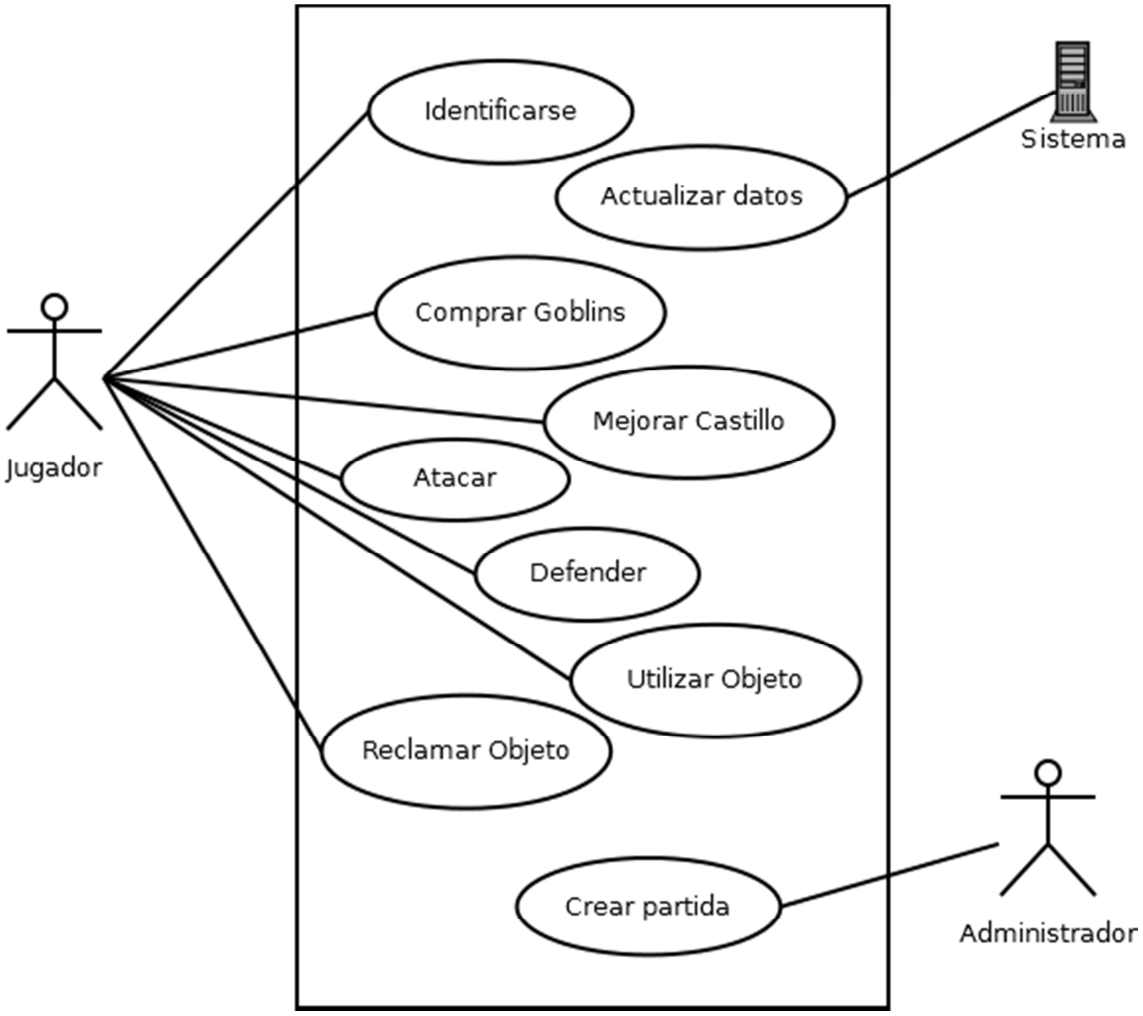
Incremento 3: Objetos

Los jugadores ahora pueden reclamar objetos de un solo uso para desequilibrar el ranking y tomar mejor posición con más facilidad.

Incremento 4: Interfaz gráfica.

En esta fase no hay ningún cambio en la funcionalidad o lógica del juego, solo se mejora la parte visual de la aplicación. Se incluyen dibujos y animaciones para dotar de ambientación al juego, y los formularios y botones son decorados para mejorar la usabilidad y la jugabilidad.

3.5 Casos de uso



- Actualizar Datos
  - Actores: Sistema
  - Descripción: El sistema actualiza los datos según la lógica del programa cuando es necesario.
- Identificarse
  - Actores: Jugador
  - Descripción El jugador se identifica con un email y su contraseña cuando entra por primera vez en la aplicación.
- Atacar:
  - Actores: Jugador
  - Descripción: El jugador ataca a otro jugador con todos sus guerreros goblins. Esta acción está limitada a una vez cada 30 minutos.
- Comprar Goblins:
  - Actores: Jugador
  - Descripción: El jugador compra cualquier cantidad de goblins.
- Mejorar Castillo:

    - Actores: Jugador
    - Descripción: El jugador mejora su castillo para producir más monedas.
- Defender
    - Actores: Jugador
    - Descripción: El jugador se defiende ante posibles ataques de los enemigos. Esta acción está limitada a una vez cada 30 minutos.
- Utilizar Objeto
    - Actores: Jugador
    - Descripción: El jugador utiliza un objeto que posee en su beneficio.
- Reclamar Objeto
    - Actores: Jugador
    - Descripción: El jugador reclama un objeto al sistema para poder utilizarlo posteriormente. Esta acción está limitada a una vez cada hora.
- Crear Partida
    - Actores: Administrador
    - Descripción: El administrador crea una nueva partida especificando su fecha de inicio y la duración.

4. Diseño

4.1 Diseño general

La aplicación está diseñada mediante la arquitectura cliente/servidor, en su bloque central y una web de apoyo que contiene un sistema de foros y un blog.

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, quien le da respuesta.

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.

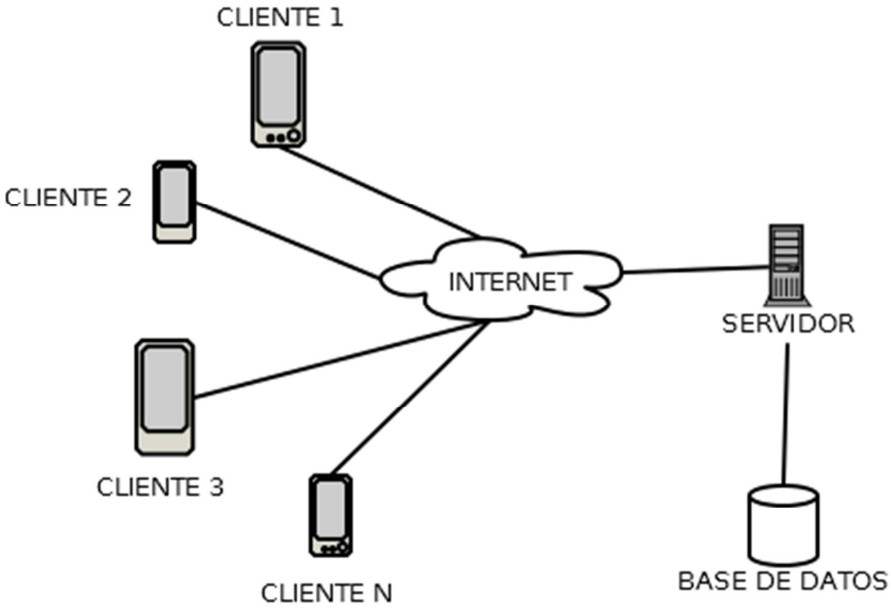


Ilustración 19

Ejemplo de compra de soldados goblins por parte de un jugador mediante un diagrama de flujo:

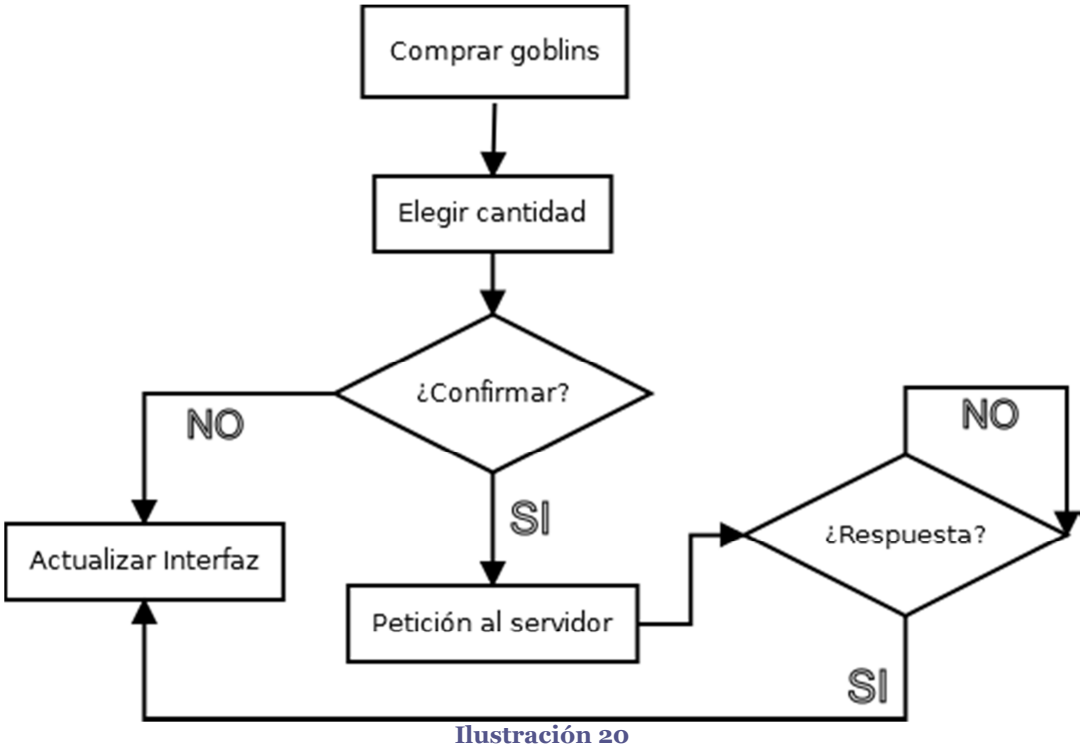


Ilustración 20

Diagrama de flujo para un usuario que se quiere unir a una partida nueva.

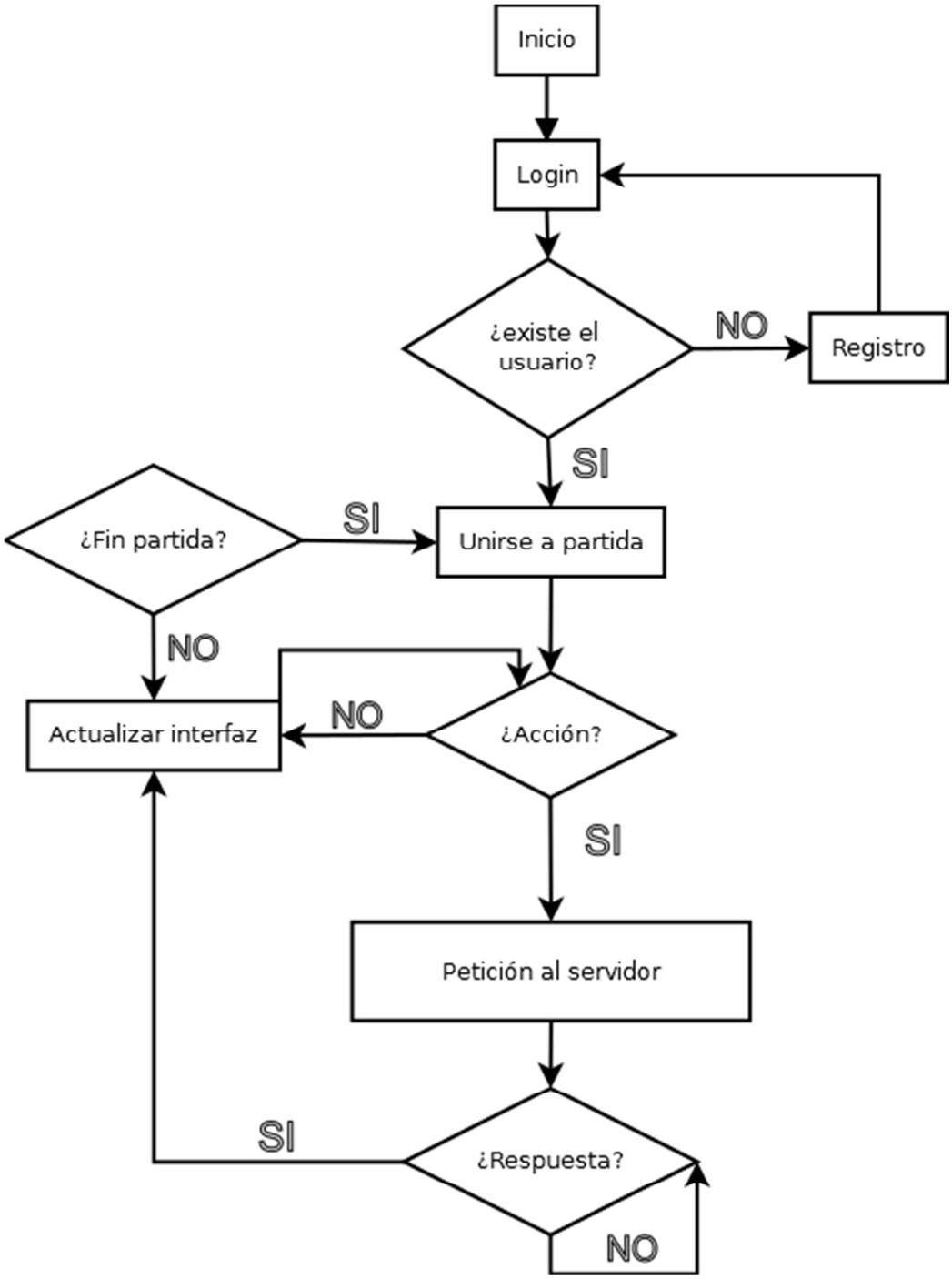


Ilustración 21

4.2 Aplicación servidor

4.2.1 Introducción

La parte del servidor se ha implementado en lenguaje de programación PHP siguiendo una arquitectura de tres capas.

La arquitectura de tres capas es una arquitectura en el que el objetivo primordial es la separación de la lógica de negocios, la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

Los tres niveles:

- Capa de presentación: es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio. En el caso de Castle & Goblins esta capa no existe puesto que todo su peso recae sobre la aplicación cliente. Esta capa en nuestro servidor se limita a formatear los datos para elaborar una respuesta en JSON y se ha fusionado con la capa de lógica.

- Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. En el caso de Castle & Goblin esta capa es la encargada de la resolución de los combates, definir el resultado del uso de un objeto, etc… Cada funcionalidad está en un único archivo php.

- Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. En el caso de nuestra

aplicación esta capa se compone de una serie de archivos que se comunican con el servidor de bases de datos MySQL donde guardamos toda la información relativa a los jugadores. Hay un archivo por cada tabla de la base de datos y se encarga de extraer o almacenar la información en dichas tablas. Para diferenciar estos archivos de los archivos de la capa de negocio se les ha nombrado con el prefijo “m_”.

4.2.2 Archivos PHP y su funcionalidad

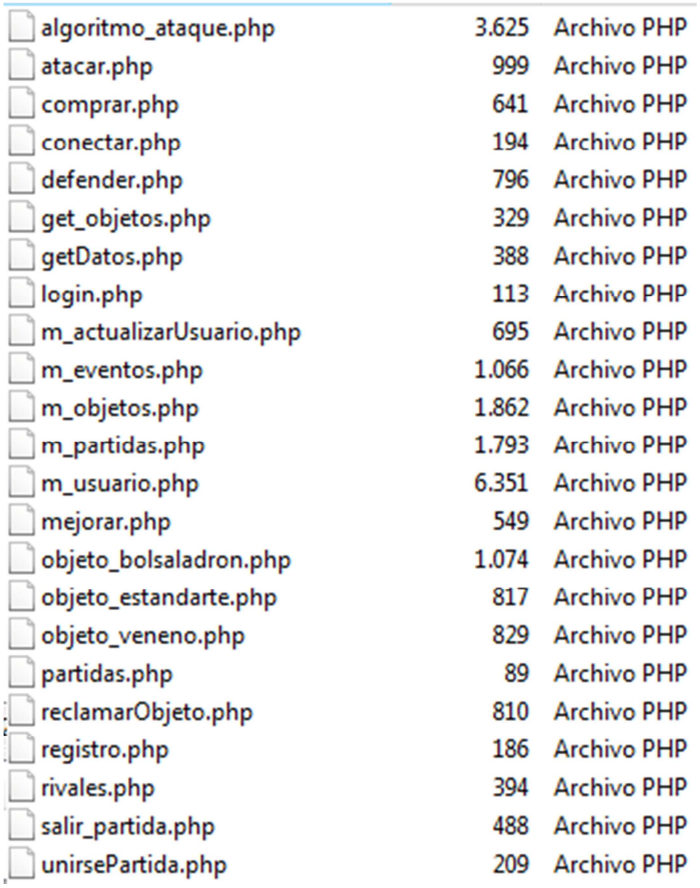


Ilustración 22

- algoritmo_ataque.php ➔ Este archivo contiene el algoritmo de ataque. No es llamado directamente, sino que es incluido por los otros archivos php que requieran que se resuelva un ataque. Se ha decidido separar del resto ya que este algoritmo se va a utilizar en diferentes partes. Requiere que las variables $atacante y $atacado estén inicializadas previamente. Al final de la ejecución de este algoritmo las variables $atacante y $atacado quedan actualizadas,

así como los datos en la base de datos para los dos jugadores afectados.

- atacar.php ➔ Este archivo si tiene una funcionalidad concreta que es la de atacar a otro jugador. Este archivo obtiene un jugador aleatorio y ejecuta el algoritmo de ataque. Una vez ejecutado devuelve un JSON con los datos del usuario actual actualizados para que la interfaz del cliente pueda ser actualizada.

- comprar.php ➔ Recibe la id de usuario , la contraseña cifrada y la cantidad de goblins que se desean comprar. Se encarga de realizar una compra de goblins. Después de hacer todas las comprobaciones actualiza la base de datos consumiendo las monedas correspondientes y actualizando la cantidad de goblins del jugador.

- conectar.php ➔ Este archivo contiene la funcionalidad y los datos para conectarse a la base de datos. Es llamado por todos los php de la capa de modelo datos para poder acceder a la base de datos y realizar sus consultas.

- defender.php ➔ Recibe la id de usuario y la contraseña cifrada. Pone el jugador en estado de defensa por 30 minutos. Nuevamente actualiza los datos del usuario y los devuelve en una cadena JSON para que la aplicación cliente se actualice.

- get_objetos.php ➔ Recibe la id de usuario y la contraseña cifrada. Realiza las comprobaciones necesarias de usuario y se comunica con la capa de datos para devolver un JSON con todos los objetos que posee el usuario en ese momento.

- getDatos.php ➔Recibe la id de usuario y la contraseña cifrada. Actualiza el usuario y devuelve en un JSON el tiempo del servidor, los datos del usuario y los últimos eventos relevantes que se han producido para ese usuario.

- login.php ➔ Recibe la id de usuario y la contraseña. Devuelve true si

la contraseña es correcta o false en caso de error. Dependiendo de esta respuesta la aplicación cliente responderá de manera adecuada.

- m_actualizarUsuario.php ➔ Este archivo corresponde a la capa de datos y no está compuesto por funciones ya que cumple un propósito muy específico. Se encarga de actualizar los datos para un usuario. Para la ejecución de este archivo, las variables $mail y $pass deben estar inicializadas y comprobadas por la capa de negocio. La capa de negocio es la que se encarga de llamar este archivo antes de realizar cualquier acción que implique el cambio de estado de usuario para que los datos sean en todo momento los más actuales.

- m_eventos.php ➔ Este archivo contiene las funciones relacionadas con la tabla eventos de la base de datos. Esta tabla se encarga de almacenar datos relativos a eventos importantes relacionados con cada jugador. Las funciones que contiene son las siguientes:

  - addEvento(): Añade un evento para el usuario indicado.
  - getEventos(): Recupera los últimos eventos para el usuario indicado.

- m_objetos.php ➔ Este archivo contiene las funciones relacionadas con la tabla de objetos de la base de datos. Esta tabla se encarga de almacenar la información de los objetos que posee cada jugador. Las funciones que contiene son las siguientes:
  - generarObjeto(): Genera un nuevo objeto aleatorio para el usuario indicado.
  - getObjetos(): Obtiene todos los objetos que posee un jugador.
  - eliminarObjeto(): Elimina un objeto de la base de datos.

- m_partidas.php ➔ Este archivo contiene las funciones relacionadas con la tabla de partidas de la base de datos. Esta tabla se encarga de almacenar información sobre las diferentes partidas que existen. Almacena tanto partidas terminadas, como actuales y futuras. Las funciones que contiene son:
  - partidas(): Obtiene las partidas activas y futuras.
  - getPartida(): Obtiene toda la información de una partida dada.
  - unirse(): Une un jugador indicado a la partida indicada. Ese jugador

queda unido a esta partida hasta que la misma finalice.
- crear_partida(): crea una nueva partida en el servidor.

- m_usuario.php➔ Este archivo contiene las funciones directamente relacionadas con el usuario. Es el archivo más extenso puesto que maneja la tabla con más datos del proyecto. Esta tabla contiene todos los datos del jugador y es la que se actualiza con mayor frecuencia. Las funciones que contiene son las siguientes:
  - login(): comprueba a partir de los datos proporcionados si el usuario se ha identificado correctamente.
  - registrar(): crea una nueva entrada de usuario en la tabla de usuarios
  - getDatos(): obtiene los datos de un usuario según los datos proporcionados
  - getRanking(): obtiene los puntos actuales del usuario en la partida a la que está unido.
  - getDatosId(): obtiene los datos de un segundo jugador a partir de su id.
  - comprar(): realiza una compra de goblins, actualiza tanto las monedas como la cantidad de goblins que posee el jugador.
  - mejorar(): aumenta en 10 la producción de monedas en la tabla del usuario.
  - salir_partida(): fuerza la salida de la partida actual a la que está unido el usuario
  - getUsuarioAleatorio(): Obtiene un usuario aleatorio que esté en la misma partida.
  - getIdUsuario(): Obtiene la id del usuario a partir de los datos de login
  - getPartidaUsuario(): Obtiene la partida a la que el usuario está unido en el momento actual.
  - defender(): Pone en modo defensa al usuario actual
  - getOponentes(): Obtiene una pequeña cantidad de oponentes, esto se utiliza para la utilización de algunos objetos.

- mejorar.php ➔ Recibe la id de usuario y la contraseña cifrada. Realiza una mejora de castillo para incrementar la producción de oro. Devuelve los datos actualizados para que la interfaz se actualice de y muestre los nuevos datos del usuario.

- objeto_bolsaladron.php ➔ Contiene la funcionalidad que se

produce al utilizar el objeto Bolsa de Ladrón. Se consume el objeto y actualiza los propietarios del objeto robado.

- objeto_estandarte.php➔ Contiene la funcionalidad que se produce al utilizar el objeto Estandarte. Se consume el objeto y se suman puntos al jugador.

- objeto_veneno.php ➔ Contiene la funcionalidad de la utilización del objeto Veneno. Se consume el objeto y se actualiza la cantidad de goblins del jugador afectado.

- partidas.php ➔ Obtiene de la capa de datos las partidas activas y las devuelvo utilizando la notación JSON

- reclamarObjeto.php ➔ Genera un nuevo objeto aleatorio para el usuario actual y lo añade a su lista de objetos. Finalmente devuelve en formato JSON los datos de usuario actualizados para la actualización de la interfaz.

- registro.php ➔ Registra un nuevo usuario en la base de datos, se comunica con la capa de datos para mandarle los datos correspondientes. Una vez realizado el registro envía un email de cortesía al usuario registrado.

- rivales.php ➔ obtiene una serie de rivales para que el usuario pueda elegir uno de ellos para llevar a cabo alguna acción. Esto se utiliza en el caso de utilizar un objeto que afecte a un enemigo.

- salir_partida.php ➔ Fuerza al usuario a abandonar la partida en la cual está jugando. Este archivo se llama una vez ha finalizado la partida para permitir unirse a una nueva.

- unirsePartida.php ➔ Une a un usuario a una partida. Este usuario quedará unido a esta hasta su finalización.

4.3 Aplicación Cliente

4.3.1 Introducción

La aplicación cliente es la parte aplicación desarrollada en java para dispositivos móviles. Esta aplicación debe ser descargada por los jugadores para poder jugar. La aplicación final se distribuye en formato apk.

Esta parte de la aplicación se encarga de mostrar una interfaz agradable con los datos de la partida del jugador y recopilar las acciones del usuario para transmitirlas al servidor. El servidor es el que se encarga de llevar toda la lógica del juego y las comprobaciones, para evitar posibles trampas de los jugadores.

4.3.2 Clases del proyecto

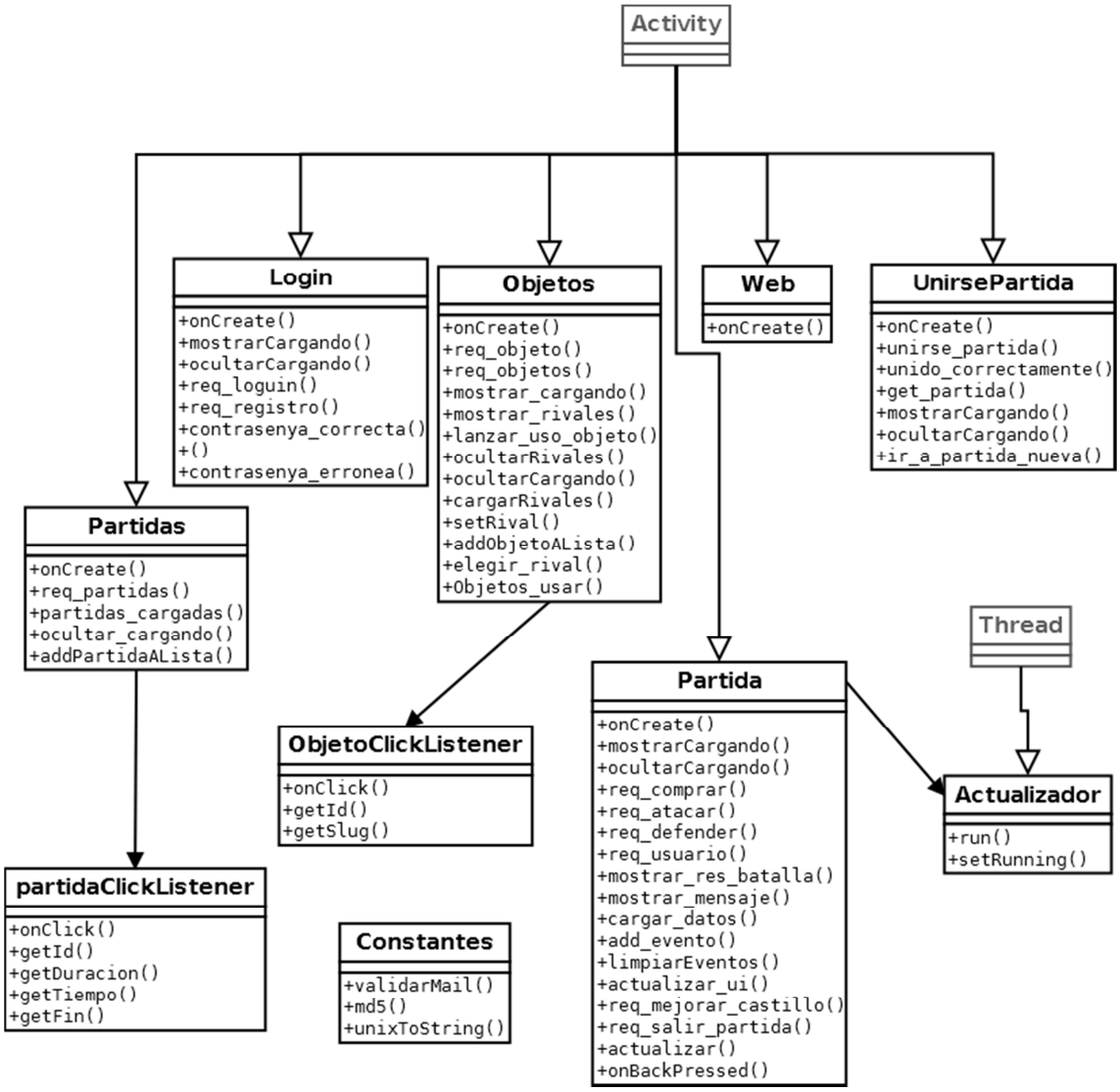


Ilustración 23

- Activity: Esta es la superclase que se utiliza para extender las clases Partidas, Login, Objetos, Partida, Web y UnirsePartida. Estas clases representan la parte lógica de las pantallas de la aplicación.

- Login: Esta clase extiende de Activity, y representa la pantalla de inicio donde los jugadores deben introducir el nombre de usuario para loguearse. Esta es una captura de la pantalla:

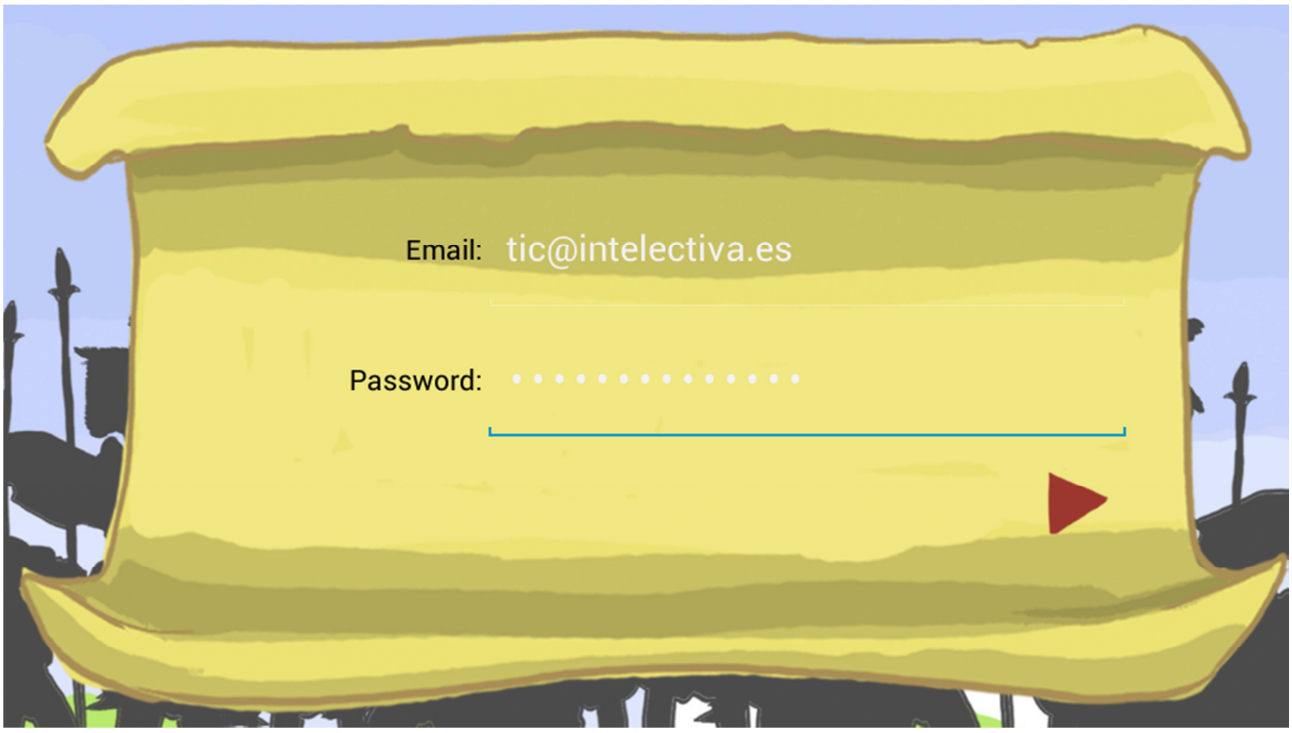


Ilustración 24

- Partidas: Esta clase también extiende de Activity y representa la pantalla donde se puede elegir la partida para unirse. Muestra una lista de partidas. Captura de la pantalla:

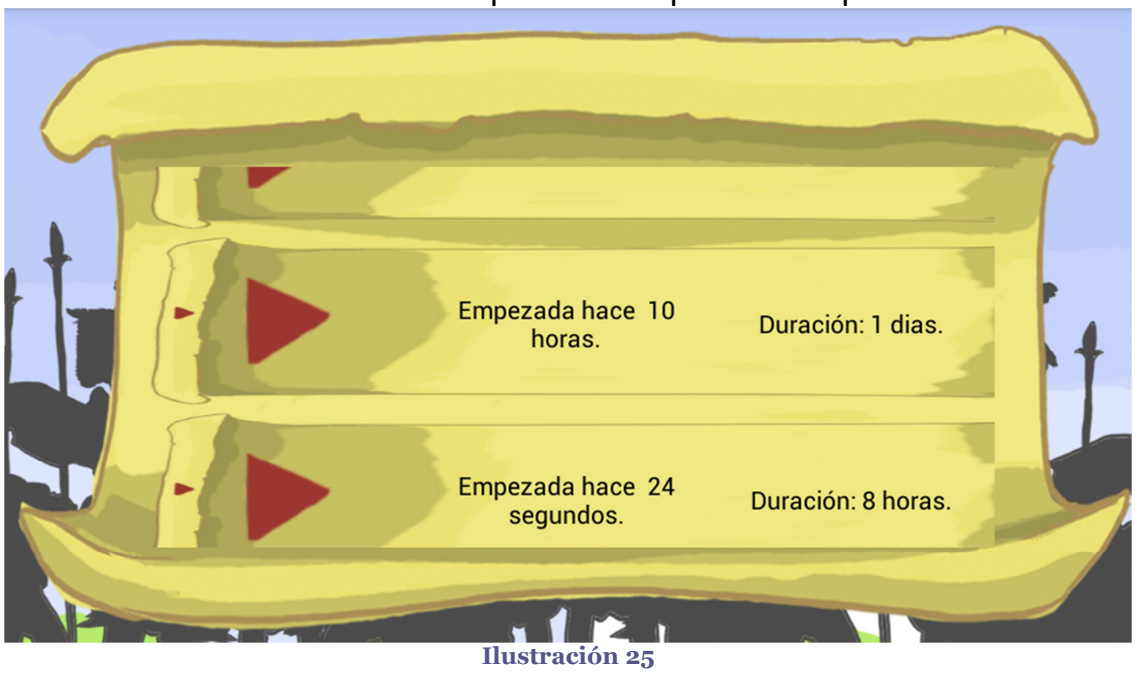


Ilustración 25

- UnirsePartida: Esta clase es nuevamente una Activity que representa una pantalla de confirmación. Esta pantalla salta cuando un usuario intenta unirse a una partida y se utiliza como confirmación, ya que una vez unido, no es posible abandonar una partida hasta que esta termine.

Ilustración 26

- Web: Esta clase también sub-clase de Activity es el acceso a la web donde consultar el ranking de la partida. Esta web se deja abierta para más ampliaciones en caso necesario, ya que la modificación de esta no requiere la modificación del paquete ni reinstalación de la aplicación. Esta activity solo contiene un WebView y un botón para volver atrás al juego de nuevo. Captura de pantalla:

Ilustración 27

- Objetos: Una nueva Activity para listar los objetos que posee el jugador. Desde esta pantalla se pueden reclamar nuevos objetos cada cierto tiempo y además utilizarlos cuando se quiera pulsando sobre ellos.

Ilustración 28

- Partida: Esta es la última Activity de la aplicación y es la pantalla principal del juego. En esta pantalla encontramos todos los controles para atacar, defender, realizar compras de goblins y

mejoras del castillo. Desde esta pantalla también podemos consultar los registros o acceder a la web para ver el ranking de la partida. Desde esta Activity es desde donde se realizaran la mayoría de las peticiones al servidor. La interfaz se actualiza cada segundo para dar en todo momento la información actualizada. Esta es una captura de la pantalla.

Ilustración 29

- Actualizador: Es una clase que extiende de thread, y es utilizada como componente por la Activity partida. Esta clase se encarga de actualizar la interfaz de la clase Partida. Esta clase contiene el método run() que se ejecuta en un hilo distinto al principal de la aplicación y se activa una vez por segundo, después actualiza la interfaz y se duerme hasta su próxima activación.

- partidaClickListener: Es una clase que extiende onClickListener pero se le han añadido tres atributos para poder asignar a cada botón en la lista de partidas. De esta manera podemos enviar datos a esta clase y cargar la partida correcta en el momento de que el jugador quiera unirse a una partida en concreto. De esta manera se puede cargar las partidas de forma dinámica.

- objetoClickListener: Es similar a la anterior, pero en este caso la utilizamos para la lista de los objetos del usuario. De esta manera

diferenciamos en que objeto ha tocado el jugador y podemos cargarlos de manera dinámica.

5. Implementación

5.1 Introducción

En este apartado se van a comentar las partes de implementación más relevantes así como la explicación de las decisiones de programación por las que se ha optado. Puesto que debido a la extensión no es posible insertar todo el código de la aplicación, se ha optado por insertar fragmentos de código para poder comentarlos seguidamente.

5.2 Servidor

5.2.1 Recepción de datos por parte del cliente

Como ya se ha explicado, el servidor recibe peticiones GET. Aquí tenemos un fragmento de código del archivo comprar.php que es el que se encarga de realizar una compra de soldados goblins.

```
require_once('conectar.php');
$mail = mysql_real_escape_string($_GET['mail']);
$pass = mysql_real_escape_string($_GET['pass']);
$cant = mysql_real_escape_string($_GET['cant']); //Cantidad de Goblins a comprar
```

Código 1

En php es muy fácil obtener las variables enviadas por GET, simplemente se recuperan de la variable global $_GET.

Una vez recuperados estos datos, se filtran con la función mysql_real_escape_string(), para evitar la inyección de código. La inyección de código SQL es un ataque en el cual se inserta código malicioso en las cadenas que posteriormente se pasan a una instancia de SQL Server para su análisis y ejecución. Por ejemplo si la consulta final fuese:

```
SELECT *
FROM tabla
WHERE email = '$email';
```

Código 2

Supongamos que se le manda el parámetro email = “falso’ DROP Database ‘bd” , una vez interpretado el código por PHP la consulta quedaría así:

```
SELECT *
FROM tabla
WHERE email = 'falso' DROP Database 'bd';
```

Código 3

De esta manera un usuario malintencionado podría borrar nuestra base de datos. La función mencionada, se encarga de sustituir los caracteres peligrosos para que no sean interpretados como si no fuesen simples caracteres.

5.2.2 Consultas a la base de datos:

Las consultas a la base de datos se realizan realizando una conexión a esta con la función mysql_connect(), y después seleccionando la base de datos correspondiente con la función mysql_select_db():

```
$link = mysql_connect('localhost', 'enacefio_goblins','1228I#]bi16G3l');
mysql_select_db('enacefio_goblins',$link);
```

Código 4

Una vez conectados, podemos realizar las consultas que queramos mediante la función mysql_query() que nos devuelve el resultado. Aquí un ejemlo de la función login() que comprueba si existe el usuario:

```php
function login($email, $pass){
    $email = mysql_real_escape_string($email);
    $pass = mysql_real_escape_string($pass);
    $query = "SELECT * FROM usuario WHERE email = '$email' AND pass = '$pass'";

    $res = mysql_query($query);
    if(mysql_num_rows($res) > 0){
        return "true";
    }
    return "false";
}
```

Código 5

La función mysql_num_rows() devuelve el número de filas que contiene el resultado de la consulta.

5.2.3 Varias consultas en un mismo procedimiento:

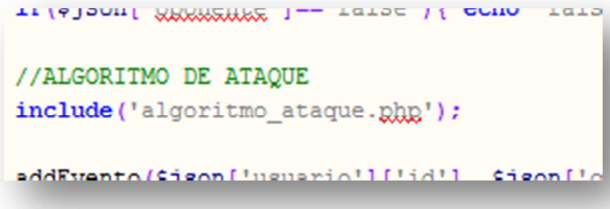
Hay veces que se requiere modificar varios valores mediante varias consultas, pero es obligatorio su cumplimiento para asegurar la consistencia de la base de datos. Para ello utilizamos las transacciones, que comienzan con un BEGIN y acaban con un ROLLBACK si ha habido algún error o con un COMMIT si todo ha funcionado bien. Aquí tenemos un ejemplo de la función salir_partida() del archivo m_usuario.php:

```php
mysql_query("BEGIN");
$query = "UPDATE usuario SET partida = -1, monedas=0, puntos=0, soldados=1,
mysql_query($query);
//echo $query;
if(mysql_affected_rows()!=1) $error = true;
if($error){
    //rollback
    mysql_query("ROLLBACK");
    return false;
}else{
    //commit
    mysql_query("COMMIT");
    return true;
}
```

Código 6

5.2.4 Reutilización de código:

Para la reutilización de código, en el proyecto se ha decidido separar el código que se utiliza reiteradamente en diferentes archivos php, para posteriormente incluirse donde sea necesario. No se ha incluido dentro de funciones dada que la extensión es grande y debe devolver varios valores. Este es el caso del algoritmo de ataque que se llama en varias ocasiones a lo largo de todo el código.



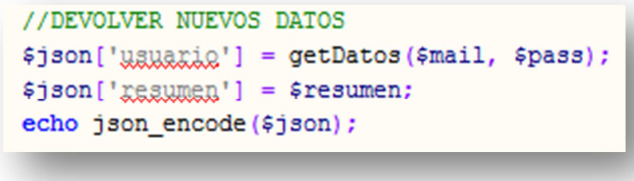
```
//ALGORITMO DE ATAQUE
include('algoritmo_ataque.php');
```

Código 7

Para este algoritmo se han definido unos parámetros que antes de su llamada deben estar inicializados y durante su ejecución se preparan otros parámetros para su posterior recuperación.

5.2.5 Encriptación JSON:

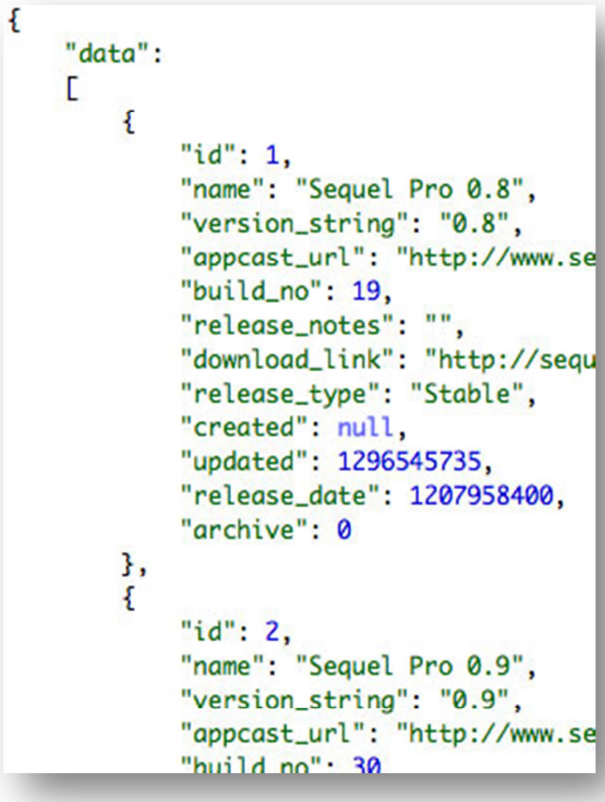
La comunicación entre el servidor y el cliente se realiza con peticiones GET y respuestas JSON. Durante toda la lógica de la aplicación se trabaja con arrays que no son más que mapas ordenados por claves. Para transformar finalmente un array en una cadena JSON se utiliza la función json_encode():



```
//DEVOLVER NUEVOS DATOS
$json['usuario'] = getDatos($mail, $pass);
$json['resumen'] = $resumen;
echo json_encode($json);
```

Código 8

Y el resultado que recibirá la aplicación del cliente será algo parecido e**sto:**



```
{
    "data":
    [
        {
            "id": 1,
            "name": "Sequel Pro 0.8",
            "version_string": "0.8",
            "appcast_url": "http://www.se
            "build_no": 19,
            "release_notes": "",
            "download_link": "http://sequ
            "release_type": "Stable",
            "created": null,
            "updated": 1296545735,
            "release_date": 1207958400,
            "archive": 0
        },
        {
            "id": 2,
            "name": "Sequel Pro 0.9",
            "version_string": "0.9",
            "appcast_url": "http://www.se
            "build_no": 30
```

Código 9

5.3 Cliente

5.3.1 Activity de Android

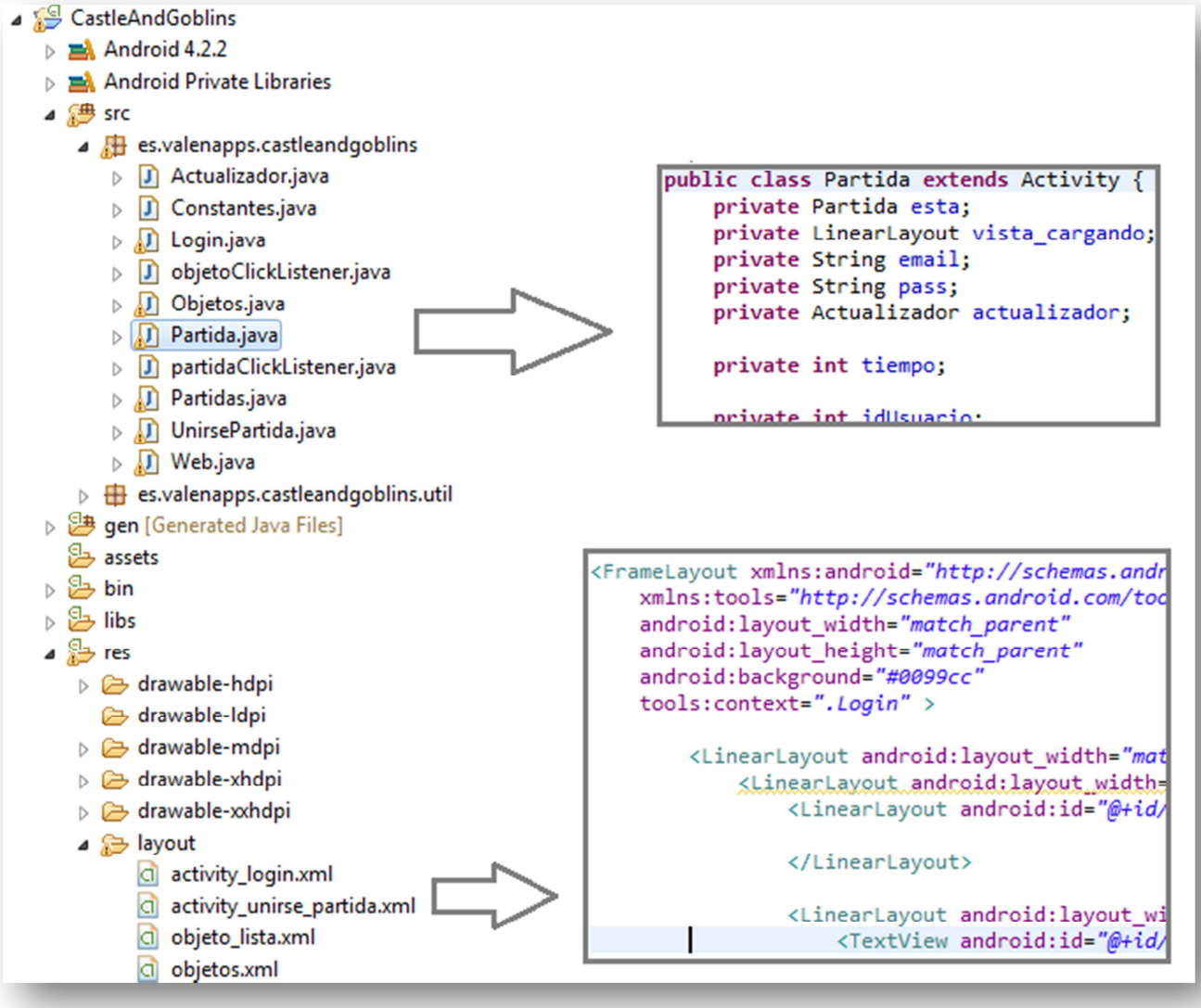
Podemos decir que todas las pantallas de una aplicación son una "activity". Es decir, que si una aplicación tiene cinco pantallas, tiene 5 "Actividades" o activities. Las activities están conformadas por dos partes: la parte lógica y la parte gráfica. La parte lógica es una archivo .java que es la clase que se crea para poder manipular, interactuar y colocar el código de esa actividad.

La parte gráfica es un XML que tiene todos los elementos que estamos viendo de una pantalla declarados con etiquetas parecidas a las del HTML, es decir, que el diseño de una aplicación en Android se hace similar a una página web.

```
public class Login extends Activity {
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       this.requestWindowFeature(Window.FEATURE_NO_TITLE);
       this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, Window
       setContentView(R.layout.activity_login);
```

Código 10

Todas las activities deben llevar por lo menos un método, el método "oncreate", que es en donde se crea la actividad o podemos decir que es donde se le da vida. Del método "onCreate" lo más importante es la línea de código: SetContentView(R.Layout.acivity_main) que es la que hace el trabajo de enlazar la parte lógica con la parte gráfica. El archivo XML que va a mostrarse cuando se mande a llamar la clase "MainActivity" es el archivo XML llamado "activity_main".



Código 11

5.3.2 Almacenamiento de datos en la aplicación:

La aplicación cliente requería almacenar ciertos datos de forma permanente tales como la contraseña o el correo que identifica al jugador. Este es un fragmento de código de la clase Login.

```
SharedPreferences prefs = getSharedPreferences("Goblins",Context.MODE_PRIVATE);
SharedPreferences.Editor editor = prefs.edit();
editor.putString("email", mail);
editor.putString("pass", pass);
editor.commit();
```

Código 12

La plataforma de Android nos da varias facilidades para el almacenamiento permanente de datos, como en este caso los datos que se guardan son muy limitados utilizaremos la clase Sharedpreferences.

Obtenemos una referencia de un objeto de la clase SharedPreferences a través del método getSharedPreferences de Activity. El primer parámetro es el nombre del archivo de preferencias y el segundo la forma de creación del archivo (MODE_PRIVATE indica que solo esta aplicación puede consultar el archivo XML que se crea).

```
prefs.getString("email","-")
```

Código 13

Para extraer los datos del archivo de preferencias debemos indicar el nombre a extraer y un valor de retorno si dicho nombre no existe en el archivo de preferencias (en nuestro ejemplo la primera vez que se ejecute nuestro programa como es lógico no existe el archivo de preferencias lo que hace que Android lo cree, si tratamos de extraer el valor de mail retornará el segundo parámetro es decir el String con la cadena "-").

```
SharedPreferences.Editor editor = prefs.edit();
editor.putInt("partida", esta.get_partida());
editor.commit();
```

Código 14

Para guardar datos utilizamos una instancia del objeto Editor, obtenido a partir de las SharedPreferences. Con los métodos putInt(), putString(),… podemos insertar un nuevo dato junto a su etiqueta dependiendo del tipo de dato que queremos insertar. Finalmente y hacer efectivo el guardado de las variables se invoca al método commit().

5.3.3 Manejando eventos en las Activities.

Tal y como se manejan los clicks en una aplicación web o de escritorio, en las aplicaciones para dispositivos móviles se deben manejar las pulsaciones de pantalla, que no siempre son tan fáciles de manejar como un simple click de escritorio. No es este nuestro caso, pero es posible que el usuario en un momento dado presione varios puntos de la pantalla, o arrastre el dedo cambiando su posición inicial. Todo esto se puede manejar con los Listeners de Android.

En este caso solamente hemos manejado eventos simples, puesto que la aplicación no requería ninguna funcionalidad avanzada de este tipo.

```
ImageView cancelar = (ImageView)this.findViewById(R.id.cancelar);
cancelar.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        Intent intent = new Intent(esta, Partidas.class);
        esta.startActivity(intent);
        finish();
    }
});
```

Código 15

En este fragmento de código manejamos la pulsación de un botón de la interfaz. Este caso concreto se encuentra en la actividad UnirsePartida y la funcionalidad que tiene es saltar a la Activity Partidas.

Lo primero es obtener el objeto que va a responder a esa pulsación, en este caso es una imagen que hace la función de botón cancelar.

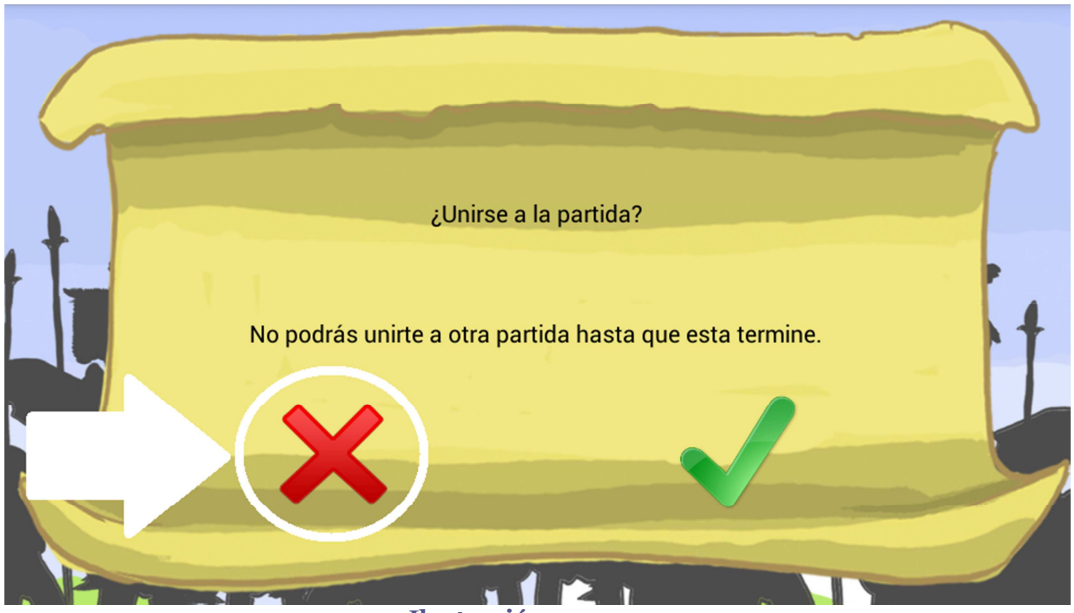


Ilustración 30

Una vez recuperado este botón, le asignamos un Listener, más concretamente un onClickListener, que responderá cuando el jugador pulse y seguidamente levante el dedo de la pantalla. Este Listener lanzará

su método onClick().

En este método onClick() hemos definido que la Activity actual se cierre y se inicie una nueva de la sub-clase Pantallas, que permite al jugador volver a elegir una partida.

5.3.4 Navegación entre Activities.

Cada vez que queremos cambiar de pantalla dentro del juego tenemos que crear una instancia de una clase Activity. Esto funciona como una pila de Objetos Activity que se visualizan en la pantalla. De manera que si desde la Activity A instanciamos una Activity B y cerramos luego la B, volveremos a visualizar la Activity A. Para evitar este comportamiento y que el flujo de pantallas sea el que se desea en todo momento, la mayoría de Activities que utilizamos se cierran con la función finish() en el momento de instanciar una nueva.

```
Intent intent = new Intent(this, Partidas.class);
this.startActivity(intent);
finish();
```

Código 16

Para crear una nueva pantalla, lo hacemos mediante un objeto tipo Intent al que se le pasa como parámetros, la actividad actual y la clase nueva que se va a instanciar. Una vez creado este objeto se lanza llamando al método startActivity de la clase Activity y se le pasa este Intent como parámetro. Seguidamente se finaliza la actividad actual llamando a su método finish().

```
Intent intent = new Intent(esta, UnirsePartida.class);
intent.putExtra("id", this.getId());
intent.putExtra("duracion", this.getDuracion());
intent.putExtra("tiempo", this.getTiempo());
esta.startActivity(intent);
finish();
```

Código 17

En ciertas ocasiones es necesario el paso de parámetros entre las diferentes pantallas. Esto se puede hacer añadiendo información extra al Intent cuando se llama. Para ello se utiliza el método putExtra(), que acepta varios tipos de parámetros.

```
Intent i = esta.getIntent();
partida_nueva = i.getIntExtra("id", -1);
```

Código 18

El Activity invocado, recibe los parámetros recuperando el Intent que la actividad invocadora ha lanzado. Para recuperar cada dato añadido a este Intent, se utilizan los métodos getIntExtra(), getStringExtra(),... según el tipo de dato introducido.

5.3.5 Peticiones al servidor.

Otra de las partes características de la aplicación cliente es la necesidad de realizar peticiones al servidor a través de Internet. En estas peticiones la aplicación cliente comunica al servidor que se va a realizar una acción, o simplemente una solicitud de información. En estos casos el tiempo de respuesta varía según la conexión, distancia, carga del servidor, etc. A causa de estos factores la respuesta del servidor puede tardar un tiempo que para el usuario puede ser demasiado largo, o incluso puede haber una pérdida de conexión y que la petición no reciba respuesta. Por estos motivos el cliente no puede quedarse bloqueado a la espera de la respuesta y necesita lanzar otro hilo de ejecución que se encargue de esta tarea para seguir respondiendo a las interacciones del jugador.

En este caso este problema se ha resuelto creando clases que implementan la Interfaz Runnable. Estas clases permiten lanzar su método run() en un nuevo hilo independiente del de la aplicación central. Este método run() es el encargado de realizar la petición, esperar la respuesta y avisar al hilo principal cuando se reciba dicha respuesta por parte del servidor. Veamos un fragmento de código de la clase UnirsePartida:

```
class Partidas_run implements Runnable {   //La petición debe ejecutarse en otro hilo

       String response;    //Respuesta del servidor -> json con los datos de la partida
       Partidas_run() { super(); }

       public void run() {
           StringBuilder builder = new StringBuilder(); //Donde almacenaremos la respuesta del servidor
           HttpClient client = new DefaultHttpClient(); //Cliente de conexión del servidor
           String email = prefs.getString("email", "-");
           String pass = prefs.getString("pass", "-");
           String url_login = Constantes.URL+"unirsePartida.php?email="+email+"&pass="+pass+"&partida="+esta.get_partida(); //Url
           HttpGet httpGet = new HttpGet(url_login);
           try {
             HttpResponse response = client.execute(httpGet);
             StatusLine statusLine = response.getStatusLine();
             int statusCode = statusLine.getStatusCode();
             if (statusCode == 200) {
               HttpEntity entity = response.getEntity();
               InputStream content = entity.getContent();
               BufferedReader reader = new BufferedReader(new InputStreamReader(content));
               String line;
               while ((line = reader.readLine()) != null) {
                 builder.append(line);
               }
             } else {
                runOnUiThread(new Runnable() { public void run() { Toast.makeText(esta, "Error Ps-3", Toast.LENGTH_SHORT).show();
                                                                   ocultarCargando(); } }); //Mostrar cargando
             }
           } catch (ClientProtocolException e) {
             e.printStackTrace();
           } catch (IOException e) {
             e.printStackTrace();
           }
          if(builder.toString().equals("true")){ //Si el servidor no devuelve false
               SharedPreferences.Editor editor = prefs.edit();
               editor.putInt("partida", esta.get_partida());
               editor.commit();
               runOnUiThread(new Runnable() { public void run() { unido_correctamente(); } }); //Cargamos la lista de partidas
          }
          else{
             runOnUiThread(new Runnable() { public void run() { Toast.makeText(esta, "Error UP-1", Toast.LENGTH_SHORT).show();
                                                                ocultarCargando(); } }); //Mostrar cargando
          }

       }
   }
```

Código 19

En este fragmento creamos una clase local que implementa la Interfaz Runnable. En algunos casos necesitamos recuperar la respuesta del servidor y por ello se ha decidido añadir un atributo response para guardar la respuesta. En este ejemplo el atributo no se utiliza, pero queda ahí para posibles modificaciones.

Las peticiones siempre se realizan verificando el jugador para evitar posibles trampas. Por este motivo necesitamos recuperar el email y contraseña almacenados localmente en la aplicación y enviarlas junto con la petición.

La petición GET se envía utilizando la clase httpGet y pasándole la url con el formato típico de las peticiones GET. Este formato contiene la url del

archivo php del servidor, seguido del símbolo ? y la lista de parámetros separados por el carácter &. Un ejemplo de petición GET es el siguiente: http://enacefio.es/goblins/unirsePartida.php?email=tic@intelectiva.es&partida=21&pass=aHIGHguh7989bjbHYTY6&%vhJVy

En esta cadena podemos ver que el archivo al que se llama es http://enacefio.es/goblins/unirsePartida.php y se le envían tres parámetros:

- Email = tic@intelectiva.es
- Partida = 21
- Pass = aHIGHguh7989bjbHYTY6&%vhJVy

El caso de la contraseña es especial ya que no se envía el dato literal sino que se encripta antes utilizando el formato de encriptación md5.

Una vez que se envía la petición con el método execute de HttpGet, el hilo se broquea para esperar la respuesta. Esta respuesta se recibe en un objeto del tipo HttpResponse que contiene toda la información. Esta información se extrae mediante los objetos y métodos que se pueden ver en el código y finalmente obtenemos un BufferedReader el cual podemos leer línea a línea.

En caso de recibir una respuesta negativa por parte del servidor, o lo que es lo mismo, una respuesta con código diferente a 200, mostramos un mensaje de error. En caso de recibir una respuesta afirmativa se continúa uniendo al jugador a la partida siguiendo la lógica de la aplicación cliente.

Finalmente y en el punto que se desea de la aplicación se crea un Thread pasándole como argumento una instancia del objeto que acabamos de definir y se arranca su ejecución con su método start() el cual invocará el método interno run() del objeto en un hilo secundario.

Hay múltiples funciones implementadas de esta manera dentro del código de la aplicación cliente, pero en su base todas son iguales. La diferencia reside en los parámetros que se pasan en la petición y en los datos que se reciben, siempre en formato JSON.

5.3.6 Lectura de respuestas JSON:

Como hemos dicho, las respuestas por parte del servidor son cadenas

de texto en formato JSON, para convertirlas en variables que podamos manejar en Java hacemos lo siguiente:

```
JSONObject datos = new JSONObject(json); //Convertimos el resultado en un Objeto JSON para acceder a sus partes
tiempo = Integer.parseInt(datos.getString("tiempo"));
String resumen = datos.getString("resumen");
```

Código 20

Siendo la variable json un String que contiene toda la cadena completa de la respuesta del servidor. Se pasa esta cadena a un nuevo objeto JSONObject y a partir de este y parseando los datos podemos convertirlos en cualquier tipo de variable de tipo básico.

5.3.7 Actualizar una interfaz desde un hilo:

Uno de los problemas que surgieron durante el desarrollo al trabajar con hilos es que según las características de Android, solo el hilo principal de la Activity puede modificar su estado. Por lo tanto no se puede cambiar un botón, o cambiar el valor de un TextView.

```
partida.runOnUiThread(new Runnable(){ public void run(){    partida.actualizar(); }});
```

Código 21

Para solucionar este problema, se ha optado por invocar el método runOnUiThread de la misma Activity. Esto permite lanzar un método en la misma Activity y permite modificar la Interfaz.

5.3.8 El objeto WebView:

El objeto WebView es un componente de la interfaz gráfica que contiene un navegador web en su interior. Dentro de este componente podemos visualizar cualquier página web mediante el motor web que tenga instalado el dispositivo.

```
WebView myWebView = (WebView) this.findViewById(R.id.webview);
myWebView.loadUrl(Constantes.URL+"web/?mail="+email+"&pass="+pass);
myWebView.setWebViewClient(new WebViewClient() {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return false;
    }
});
```

Código 22

Este código corresponde a la clase Web de la aplicación. Este fragmento se encarga de cargar una web mediante el método loadUrl de la clase. Adicionalmente y para poder mostrar los datos referentes al usuario, se envían los datos de login del usuario.

Para evitar que cada vez que pulsemos un enlace web se abra el navegador por defecto debemos crear un WebViewClient donde se sobreescriba el método shouldOverrideUrlLoading() que cargará las nuevas url en la misma WebView.

5.3.9 Layouts y tamaños:

Una de las tareas más complicadas en este proyecto ha sido la implementación de la interfaz, debido a la enorme cantidad de resoluciones existentes en dispositivos Android. Para que la apariencia de la aplicación fuese igual en todos los dispositivos, se ha optado por utilizar LinearLayouts, que son los únicos layouts que permiten especificar sus partes en porcentajes. De esta manera se consigue que todos los objetos queden bien alineados y no se desplacen en diferentes resoluciones.

```
<LinearLayout android:layout_width="match_parent" android:layout_
    <LinearLayout android:layout_width="match_parent" android:lay
        <LinearLayout android:id="@+id/boton_log" android:visibil

        </LinearLayout>

        <LinearLayout android:layout_width="0dp" android:layout_h
            <TextView android:id="@+id/nick" android:layout_weigh
            <LinearLayout android:layout_width="match_parent" and
        </LinearLayout>
        <LinearLayout android:visibility="visible" android:layout_
            <ImageView android:id="@+id/boton_info" android:layou
        </LinearLayout>
    </LinearLayout>
    <LinearLayout android:layout_width="match_parent" android:lay
        <View android:layout_width="0dp" android:layout_height="m
        <TextView android:id="@+id/monedas" android:layout_weight
            <!-- <View android:layout_width="0dp" android:layout_
        <TextView android:id="@+id/puntos" android:layout_weight=
        <View android:layout_width="0dp" android:layout_height="m
    </LinearLayout>
    <LinearLayout android:layout_width="match_parent" android:lay
```

Código 23

LinearLayout coloca las vistas una a continuación de otra, sin superponerlas nunca. Es decir, las alinea.

Esta alineación puede ser vertical u horizontal. O sea, que LinearLayout sirve para colocar vistas en una misma fila o columna, pero no ambas cosas a la vez. Para ello las podemos anidar.

Las vistas se colocan en el mismo orden en se agregan al diseño o en el que aparecen en el archivo XML.

De forma predeterminada, LinearLayout tiende a establecer el mismo tamaño para cada vista que contiene, repartiendo el espacio disponible de forma equitativa entre todas ellas. Sin embargo, algunas propiedades de las vistas y ciertos parámetros que el propio LinearLayout añade permiten variar este comportamiento. Para ello podemos asignar un valor de 0 al ancho a los layouts con orientación horizontal o un 0 a los veticales y después asignarles un peso dentro del layout. Para hacer esta tarea más fácil, se ha optado por darles un valor de 1 a 100 siendo la suma de estos siempre de 100. De esta manera podemos utilizar un sistema porcentual.

5.3.10 Popups en el Layout.

En ciertas partes de la aplicación también se pueden ver popups, como es el caso de la compra de goblins. Los LinearLayout que hemos comentado en el apartado anterior no permiten la inserción de estos PopUps. Lo que se ha hecho en este caso es la superposición de varios LinearLayout, el layout base y un layout para cada PopUp. Estos LinearLayout se insertan dentro de un FrameLayout. Estos PopUp no son visibles hasta que se los invoca, y se vuelven a esconder cuando no son necesarios.

FrameLayout no realiza ninguna distribución de las vistas, simplemente las coloca unas encima de otras. Esto le evita tener que relacionar los tamaños de unas vistas con los de las demás, por lo que se ahorra recorridos del árbol de vistas, tardando menos en mostrar su contenido.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#0099cc"
    tools:context=".Login" >

        <LinearLayout android:layout_width="match_parent" android:layout_
            <LinearLayout android:layout_width="match_parent" android:lay
                <LinearLayout android:id="@+id/boton_log" android:visibi

                </LinearLayout>

                <LinearLayout android:layout_width="0dp" android:layout_h
                    <TextView android:id="@+id/nick" android:layout_weigh
                    <LinearLayout android:layout_width="match_parent" and
                </LinearLayout>
```

Código 24

Ilustración 31

6. Conclusiones y trabajos futuros

Creo que los objetivos del proyecto se han logrado, ya que al final se ha desarrollado una aplicación completa y funcional utilizando una gran variedad de tecnologías existentes. Estas tecnologías a día de hoy son muy novedosas y están en el mercado muy poco tiempo. Es por esto que los conocimientos adquiridos en el desarrollo de esta aplicación creo que me van a ser muy útiles en el futuro.

La parte del servidor se creó la primera y me ha servido para aprender el lenguaje de programación PHP, la utilización de un servidor Apache y su conexión con el servidor de bases de datos.

Por otra parte la aplicación cliente me ha servido para aprender toda la estructura de un proyecto Android. Una de las partes que más difícil me resultó, fue la implementación de la interfaz mediante layouts ya que es bastante diferente a todo lo que había visto hasta el momento.

También he tenido la oportunidad de trastear con las plataformas WordPress y PHPBB que me parecen muy completas y funcionales a pesar de ser software libre sin ninguna empresa que los apoye.

Durante todo el proyecto se realizaron distintas pruebas de funcionamiento. En estas pruebas han participado entre 6 y 10 jugadores como máximo y fueron todo un éxito. Por supuesto se detectaron algunos fallos y se corrigieron fallos de diseño.

Este proyecto queda terminado y funcional, pero aun así hay varias ideas para el futuro. Las más inmediatas serían la promoción tanto de la aplicación como de la web y el foro. Más adelante también se podrían desarrollar nuevos objetos y funcionalidades que aumentasen la jugabilidad y la diversión de los jugadores.

7. Bibliografía

La mayor parte del tiempo y como primera fase de aprendizaje, la fuente bibliográfica más grande que he utilizado han sido los vídeos de Youtube. Hay gran cantidad de tutoriales para casi cualquier cosa. Para temas más específicos he utilizado las webs de documentación oficiales.

- Youtube: http://youtube.com
- Buscador de Google: http://google.es
- Wikipedia: http://es.wikipedia.org
- Stack Overflow: http://stackoverflow.com/
- Documentación de Java: http://docs.oracle.com/javase/7/docs/api/
- Documentación de Android: https://developer.android.com/guide/index.html
- Documentación de PHP: http://php.net/docs.php
- Documentación de Apache: http://httpd.apache.org/docs/
- Página oficial de Wordpress: https://es.wordpress.org/
- Página oficial de PHPBB: http://www.phpbb-es.com/
- Foros de PHPBB: http://www.phpbb-es.com/foro/