Universidad Rey Juan Carlos

Escuela Técnica Superior de Ingeniería Informática

Ingeniería Informática

Curso Académico 2011/2012

Proyecto Fin de Carrera

Implementación y Despliegue de un Juego Geolocalizado para Dispositivos Android: Los Conquistadores

Autor: Francisco Javier Quero Castrillo

Tutor: Gregorio Robles Martínez

Agradecimientos

Quiero aprovechar este momento para dar las gracias a todas las personas que han participado, a su manera, en la consecución de esta obra.

En primer lugar a mis padres y hermano, por el apoyo y confianza que me han dado durante toda mi vida, y especialmente en estos años de estudio universitario.

A Gregorio Robles, por su inestimable ayuda y otorgarme la oportunidad de trabajar en este proyecto. Al departamento del GSyC por facilitar los materiales necesarios para realizar las pruebas.

Finalmente, a la universidad Rey Juan Carlos y a todos los compañeros que he conocido a lo largo de la carrera, gracias a ellos he vivido algunos de mis mejores momentos.

Resumen

En este proyecto fin de carrera se acomete el diseño e implementación de un juego geolocalizado para la plataforma Android, denominado "Los Conquistadores". A través de esta aplicación los usuarios disfrutarán de una forma de juego diferente. Con sus terminales móviles podrán desplazarse por el mundo real, interactuando con elementos virtuales y demás participantes. De esta manera se ofrece una experiencia innovadora y gratificante.

El sistema especificado se basa en el paradigma cliente-servidor. El formato JSON ha sido seleccionado para el intercambio de datos y el protocolo HTTP es el encargado de establecer la conexión entre ambas partes. Por un lado se construye el servidor web haciendo uso del *framework* Django y el lenguaje de programación Python. Con el gestor de base de datos PostgreSQL, se almacenan todos los elementos relacionados con las partidas. El servicio web está integrado en la red social LibreGeoSocial, ya que proporciona la estructura y funciones necesarias para manejar elementos geolocalizados en el sistema. Mediante cualquier navegador web los usuarios pueden acceder al servidor, con el fin de crear y controlar sus propias partidas.

Los participantes formarán parte del juego gracias al manejo del cliente Android. Haciendo uso de las últimas tecnologías de los dispositivos móviles (conectividad 3G, GPS, cámara, etc.), se brindará un amplio abanico de posibilidades: conquistar, espiar o mover tropas entre territorios, son algunas de las funcionalidades. Todas estas acciones se llevarán a cabo a través de un Google Maps habilitado en la aplicación. También se integra un servicio de mensajería cuyo objetivo es facilitar la comunicación entre los participantes.

En último lugar, se han desarrollado una serie de pruebas en un entorno real con voluntarios, cuyo objetivo son las de verificar la robustez del sistema, detectar y corregir errores, comprobar la aceptación y recoger sugerencias para mejorar la aplicación.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este capítulo se explicarán los principales conceptos que se han tratado a lo largo de la memoria. En primer lugar se revelarán los motivos que me han guiado para la realización de este proyecto. Seguidamente, se expondrán las características más relevantes de los juegos ubicuos así como algunos ejemplos que han servido de base, además se detallará la mecánica y cualidades del juego desarrollado. Por último, se describe el estudio realizado de cada herramienta y tecnología usadas a lo largo de este PFC.

1.1. Motivación

La evolución de los dispositivos tecnológicos en las últimas décadas ha sido sensacional, cada vez son más pequeños pero sus prestaciones y funcionalidades no dejan de crecer. Como no podía ser menos, los teléfonos inteligentes o *smartphones* se han aprovechado de este progreso. En rasgos generales, estos dispositivos abren un nuevo abanico de posibilidades (instalación de aplicaciones, mayor conectividad, procesamiento de datos, etc.) para el usuario, donde los teléfonos convencionales no son capaces de llegar.

A mediados de la década de los 90's, la compañía Nokia lanzaba los primeros terminales que cumplían estas características. A principios del 2007, Apple anunciaba el iPhone y un año después, Google lanzaba Android (su primer sistema operativo móvil) junto al HTC Dream, el primer terminal con este sistema operativo. Estos acontecimientos supusieron una gran revolución en todos los sentidos, ya que se ponía a disposición de cualquier usuario *smartphones* que cabían en la palma de la mano con un gran potencial, comparable a la de un ordenador personal.

Poco antes de elegir el proyecto a desarrollar para terminar mis estudios, adquirí el HTC Desire, mi primer teléfono Android. En un principio poco sabía de esta plataforma pero enseguida despertó mi curiosidad. Durante la presentación de los PFC ofertados por el departamento GSyC, se me presentó la gran oportunidad de poder conocer en mayor profundidad Android y desarrollar una aplicación que pudiese aprovechar las características punteras que ofrecen estos teléfonos inteligentes. Esto, unido a la temática de la aplicación a desarrollar (implementación de un juego geolocalizado) fueron un gran acicate para decantarme por su elección.

Además, este proyecto ha sido ideal para poner en práctica los valiosos conocimientos adquiridos a lo largo de la carrera, asignaturas relacionadas con bases de datos, redes o programación en general, han sido fundamentales para el diseño y desarrollo de algunos componentes que forman parte de este. Por otro lado, ha sido una gran motivación el reto de conseguir nuevas capacidades, aprender otros lenguajes de programación y conocer herramientas hasta entonces desconocidas por mi parte.

Por estas razones, puedo asegurar que este proyecto fin de carrera es lo suficientemente completo ya que se tratan multitud tecnologías que han sido debidamente estudiadas para poder llevar a cabo con éxito esta obra.

1.2. Juegos geolocalizados

Los juegos geolocalizados o ubicuos se pueden considerar como una nueva forma de entretenimiento, donde los participantes no tienen que estar sentados frente a un televisor, si no que son parte del propio juego y mediante su dispositivo móvil podrán realizar diversas acciones, de una manera natural, con el escenario que les rodea.

La localización de los propios usuarios deja de ser un problema y pasa a ser una pieza fundamental en el desarrollo de la partida, ya que dependiendo de su ubicación, la aplicación le permitirá realizar determinadas acciones y otras podrán estar bloqueadas porque no se cumplen ciertos requisitos. Este hecho implica que se tengan que realizar al aire libre, de tal manera que el usuario pueda involucrarse con el mundo real, es decir, las calles, edificios, parques, etc.

Otra peculiaridad de estos tipos de juegos, que hacen que sean tan flexibles para los usuarios, es el tema de la temporalidad. En algunos casos, se le solicitará realizar alguna acción en un periodo de tiempo. En otras ocasiones, los

jugadores gozarán de una mayor libertad pudiendo conectarse a lo largo del día para ver el estado de su partida.

También hay que tener en cuenta el componente social. Son los propios usuarios quienes ponen límites al juego, a través de la interacción con el escenario, modificándolo según sus necesidades o las propias reglas de la aplicación. Podrán tomar diferentes roles y estrategias según las relaciones que se establezcan entre ellos.

Como conclusión, este concepto de entretenimiento es una vuelta de tuerca de los juegos clásicos y sus posibilidades son inmensas. Fomentan valores como la actividad física, establecer nuevas relaciones sociales o la diversión, mediante la competitividad sana entre los participantes. A continuación se ponen algunos ejemplos, destinados a la plataforma Android, que se engloban dentro de esta categoría de juegos. Además los dos últimos me sirvieron para decidir qué tipo de género iba a seguir la aplicación a desarrollar.

1.2.1. Libresoft Gymkhana

Un buen ejemplo de cómo las nuevas tecnologías son utilizadas para mejorar actividades tradicionales se puede ver en el proyecto LibreSoft Gymkhana [1] desarrollado por el departamento del GSyC/LibreSoft de la universidad Rey Juan Carlos con la colaboración de la red eMadrid.

Haciendo uso de la red social LibreGeoSocial, se trata de un juego libre, geolocalizado y se enmarca dentro del ámbito de las aplicaciones destinadas a la educación (*M-Learnig*) y turismo. Desde una página web el organizador podrá crear y monitorizar, de una manera fácil, los elementos que componen una gymkhana. Los usuarios, a través de su terminal móvil Android, tendrán que ir desplazándose a diferentes ubicaciones y resolver el reto propuesto por la aplicación. El juego es variado y completo gracias a los diferentes tipos de pruebas a completar: textuales, fotográficas, geolocalizadas y de realidad aumentada.

En definitiva este proyecto busca optimizar la organización de las gymkhanas clásicas gracias a una reducción general de costes, una continua comunicación entre jugadores y organizador, y la posibilidad de ofrecer retos únicos que de otra manera serían imposible de llevar a cabo.

Figura 1.1: Logotipo de Libresoft Gymkhana.

1.2.2. Fable 3 Kingmaker

Fable III Kingmaker [2] se trata de una aplicación gratuita para iPhone y Android lanzada para promocionar el juego Fable III, que saldría pocas semanas después para Xbox 360 y PC.

Este juego geolocalizado permite tomar parte en la batalla eligiendo uno de los dos bandos disponibles: Rebeldes o Reales. Mediante la interfaz de realidad aumentada y el GPS, el usuario podrá ir por la calle recogiendo objetos escondidos y reclamar territorios reales con la colocación de banderas virtuales. El color de la zona cambiará según el bando propietario y otros jugadores rivales podrán desplazarse a ese lugar para recuperarlo. Finalmente el equipo que haya colocado más banderas en el territorio se quedará con este.

Con estas acciones, los usuarios se veían recompensados con oro que posteriormente podían transferirlo al juego promocionado. Aunque actualmente ya no se encuentra disponible, cabe mencionar que tuvo una gran aceptación por parte del público. Durante los 92 días que estuvo activo se plantaron más de 3.5 millones de banderas a lo largo del continente europeo, dando como ganador al bando de los Rebeldes.

Figura 1.2: Logotipo de Fable III Kingmaker.

1.2.3. Parallel Kingdom

Parallel Kingdom [3] es un RPG (*Role Playing Game*) multijugador que utiliza la localización obtenida del GPS para ubicar al usuario en un espacio virtual

dentro del mundo real. Fue el primer juego de rol basado en la localización para las plataformas iOS y Android.

Entendiendo el género en el que se enmarca, las posibilidades que ofrece son casi infinitas. Permite crear tu propio personaje con unas aptitudes iniciales y mejorarlas según avance de nivel. También se le puede equipar armas y armaduras obtenidas como recompensas de los combates librados con otros usuarios o monstruos.

El principal objetivo es crear desde cero un nuevo reino y hacerlo prosperar. Utilizando la ubicación del usuario en el mundo real, tendrá que reclamar nuevos territorios, obtener recursos, construir máquinas de guerra, entre otras posibilidades. Todas estas acciones se llevan a cabo a través del mapa de la aplicación. Otra característica a destacar es el gran componente social con el que cuenta, el usuario podrá formar parte de una gran comunidad mediante la interacción con otros jugadores de todo el mundo. Gracias al comercio y colaboración entre ellos, la construcción y defensa de las ciudades es más fácil y divertida.

Parallel Kingdom es un juego *freemium*, es decir, descargar y jugar a la aplicación es totalmente gratuito, pero los usuarios tienen la posibilidad de comprar contenido exclusivo. Además tiene una gran comunidad y tutoriales que facilitan su iniciación y el descubrimiento de los secretos que aguarda.

Figura 1.3: Logotipo de Parallel Kingdom.

1.3. Introducción a Los Conquistadores

Los Conquistadores es un juego geolocalizado pensado para disfrutarlo con tus amigos al aire libre. A través del mapa habilitado, el jugador deberá conquistar y defender el mayor número posible de **territorios**, objetivo que no será fácil de cumplir debido a tus rivales.

Para lograr este fin, cada equipo contará con la ayuda de sus tropas. Las **tropas** son soldados ubicados en los territorios y en el propio **equipo**, pudiendo

ser reclutadas para realizar ataques e intentar conquistar las diferentes zonas del mapa. También son esenciales para defender nuestros intereses, cuanto más unidades tenga un territorio mayor será la probabilidad de repeler un ataque rival. Para favorecer este tipo de acciones y aumentar el entretenimiento, la cantidad de tropas se incrementarán periódicamente.

Los **espías** serán de gran ayuda para desarrollar una buena estrategia a la hora de realizar una conquista. Son unidades limitadas que proporcionan una valiosísima información del rival, el número exacto de tropas que defienden un territorio. De esta forma el equipo atacante puede realizar un asalto más preciso y minimizar así sus pérdidas.

El enfrentamiento directo entre los equipos es posible. Aunque la localización entre ellos es anónima, esto dejará de serlo cuando se encuentren a una determinada distancia y puedan retarse en un **duelo**. Al igual que una conquista exitosa, el equipo vencedor sumará unos preciados puntos para alzarse con la victoria final.

Cabe destacar que los equipos podrán **comunicarse** en tiempo real mediante el envío de **mensajes**. A través de este sistema de comunicación, integrado en la propia aplicación, los jugadores conseguirán establecer nuevas estrategias y asociarse para acortar distancia a los rivales más poderosos. Estas alianzas pueden romperse en cualquier momento y volver a la situación original. Finalmente ganará el equipo con la mayor puntuación lograda a lo largo de la partida, convirtiéndose en el gran Conquistador.

1.4. Paradigma Cliente-Servidor

El modelo arquitectónico cliente-servidor [4] se puede entender, desde un punto de vista funcional, como una arquitectura distribuida donde los dos tipos de componentes se comunican mediante un modelo de interacción petición-respuesta. Como se observa en la figura 1.4, el cliente solicita un servicio al servidor (envía la petición) a través de la red, este la procesa y envía la respuesta con la información solicitada. A continuación se describen los elementos principales que componen este modelo.

El **cliente** es la parte activa del sistema porque es el que inicia el dialogo con el servidor mediante sus peticiones. Este proceso está diseñado para que interaccione con el usuario final y entre sus funciones destacan las siguientes: administrar la

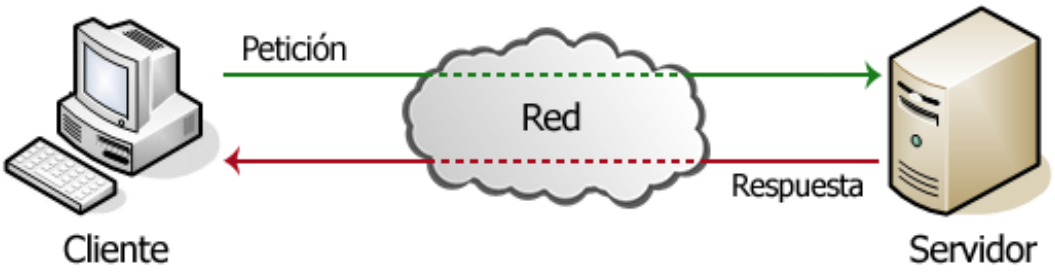


Figura 1.4: Modelo Cliente-Servidor.

interfaz de usuario, acceder a los servicios distribuidos, procesar la lógica de la aplicación y dar formato a las respuestas recibidas.

El **servidor** es la parte pasiva del sistema y proporciona los servicios a otros procesos. Es el encargado de atender las múltiples peticiones, aplicar las reglas de negocio y generar una respuesta para transmitírsela al cliente.

El **middleware** es el conjunto de protocolos que permite a los clientes acceder a los servicios a través de una red. La existencia de este componente no es estrictamente necesaria ya que los procesos pueden residir en la misma máquina. Sin embargo esta práctica no es muy habitual, porque la mayoría de los sistemas clientes-servidor se implementan como sistemas distribuidos.

Según donde se encuentre la lógica de negocio, se pueden identificar diversos tipos de clientes:

- **Cliente pesado:** También denominado grueso. Es aquel que implementa gran parte de la lógica de la aplicación, es decir, procesa la información antes de ser enviada. La complejidad de este tipo de cliente es mayor y requiere una mayor capacidad de procesamiento y almacenamiento.
- **Cliente fino:** También conocido como liviano o flaco. Este tipo de cliente apenas implementa la lógica de la aplicación y únicamente actúa de intermediario entre el usuario y el servidor. No precisa de gran cantidad de recurso hardware para su funcionamiento.
- **Cliente híbrido:** Como su nombre indica, es una mezcla de los dos anteriores. Implementa, en mayor o menor media, parte de la lógica de la aplicación.

Dependiendo del tipo de cliente en el que se base el modelo, el desarrollo del otro

componente se verá afectado. Cuanto más pesado sea, el diseño e implementación del servidor será más sencillo, y viceversa.

El uso de este modelo arquitectónico aporta las siguientes ventajas:

- Escalabilidad del sistema. Gran facilidad para añadir o quitar clientes sin afectar al funcionamiento del sistema.
- Hardware y software más heterogéneo según la funcionalidad de los componentes. Esto implica una mayor flexibilidad y reducción de costes.
- Control y protección. El servidor se encarga de gestionar el acceso y mantener la integridad de los datos de posibles ataques o clientes defectuosos.

Las principales desventajas de este paradigma son:

- Una gran cantidad de tráfico puede saturar al servidor, afectando negativamente al rendimiento de los componentes.
- Escasa robustez. Si el servidor se cae se ve comprometido todo el sistema, porque no será posible proporcionar los servicios a los clientes.

1.5. Python

Python [5] es un lenguaje de programación creado a finales de la década de los 80's por Guido van Rossum bajo una licencia de de código abierto, que posteriormente se denominó *Python Software Foundation License*. En esa época Guido van Rossum se encontraba trabajando en la plataforma Amoeba y la necesidad de crear una nueva herramienta que interactuase con este sistema, fue el artífice para la creación del lenguaje Python.

Es un lenguaje interpretado o de *script*, esto significa que cuando se ejecuta el código por primera vez, se generan unos bytecodes que serán utilizados por el intérprete. Esta característica favorece el rápido desarrollo de programas y la independencia de la plataforma. Por el contrario, la ejecución de estos tipos de lenguaje es menos eficiente que los lenguajes compilados. Algunos de los lenguajes interpretados más conocidos y utilizados en la actualidad son: ASP, Java, Ruby, PHP y Perl.

El tipado dinámico es otra de las características que ofrece Python, esto implica que el tipo de variable se detecta en tiempo de ejecución, y puede cambiar

según el tipo de valor que se le asigne. También es fuertemente tipado, por lo que una variable no puede ser tratada como otro tipo diferente al que tiene asignado, siendo necesario convertirla explícitamente.

Python está considerado como un lenguaje multiparadigma. Permite al desarrollador crear aplicaciones siguiendo un estilo de programación orientada a objetos, imperativa e incluso funcional. También se encuentra disponible para múltiples plataformas (UNIX, Windows, Linux, Mac OS, etc.), por lo que en condiciones normales, cualquier programa desarrollado en Python funcionará en estos sistemas sin la necesidad de realizar importantes modificaciones.

Un aspecto importante es la gran cantidad de librerías y módulos incorporados en el lenguaje que simplifica la tarea al programador. Estas librerías son utilizadas para el tratamiento de diferentes elementos como pueden ser string, números, archivos, etc. Además, se pueden importar librerías para abordar temas específicos como la gestión de BBDD, desarrollo de sitios web o tratamiento de algunos archivos multimedia.

Todas estas características unidas a la sencillez de la sintaxis que favorece el desarrollo de código fácilmente legible, hace que Python sea un lenguaje simple, potente e ideal para aprender a programar. Por estos motivos, es manejado en grandes proyectos e importantes instituciones como Google, Blender 3D, NASA, IBM, etc. [6]

1.6. Django

Django [7] es un *framework* de desarrollo web de código abierto, escrito en el lenguaje Python y que sigue el patrón MVC (Modelo-Vista-Controlador). Fue creado en el año 2003 para gestionar, de una forma más eficaz, varias páginas web del diario estadounidense *Lawrence Journal-World*. En 2005 se publicó bajo la licencia de software libre BSD y en 2008 Django pasó a ser cargo de la *Django Software Foundation*.

Como cualquier otro *framework*, Django es un conjunto de componentes que pretende agilizar la creación de sitios web relativamente complejos, reutilizar código ya existente (siguiendo el principio *Don't Repeat Yourself*) y promover buenas prácticas mediante el uso de patrones. Python es el lenguaje utilizado en este *framework*, desde la configuración hasta el desarrollo de las librerías. A continuación, se detallan algunas de las características más importantes:

- **Mapeo objeto-relacional:** Facilita la creación del modelo de datos en Python, junto con una API para la manipulación de la base de datos. Esto simplifica enormemente la labor al desarrollador cuando tiene que interactuar con la base de datos.
- **Interfaz de administración:** Se pone a disposición un sitio donde el administrador puede crear, modificar o eliminar contenidos del sitio web. Aunque su activación no es obligatorio, permite ahorrar tiempo al desarrollador evitando la creación de formularios para realizar las tareas indicadas anteriormente.
- **Elegante diseño de las URLs:** Django permite crear un esquema claro y conciso del conjunto de URLs, basado en expresiones regulares.
- **Sistema de caché:** Mejora del rendimiento del sistema mediante el uso de un sistema de caché, que le permite guardar páginas ya visitadas y evitar nuevas consultas a la base de datos.
- **Sistema de plantillas:** Pone a disposición del desarrollador un potente lenguaje para la generación de plantillas, permitiendo separar el diseño y contenido de estas del código.

Incorpora un ligero servidor web y la base de datos SQLite3 lo que facilita, todavía más, el desarrollo y pruebas del software. El problema de estos productos es que no se recomienda su uso fuera del entorno de pruebas, por esta razón se recomienda la utilización de los principales servidores web de producción (Apache, Cherokee, Nginx, etc.) y sistemas de gestión de bases de datos (PostgreSQL, MySQL o Oracle). También es capaz de soportar aplicaciones SIG (Sistemas de Información Geográfica) mediante el uso del *add-on* GeoDjango [8]. En el caso de utilizar alguno de estos componentes será necesario utilizar módulos auxiliares, como se explica en la sección 3.6, para que el servidor y la base de datos integren el lenguaje Python o gestionen correctamente los objetos geográficos.

Como ya se ha indicado, Django hace uso del patrón MVC pero con una ligera interpretación por parte de sus desarrolladores [9], de tal manera que lo llegan a definir como un *framework* MTV (Modelo-Template-Vista). La asociación entre ambos conceptos es directa: El Modelo no se ve afectado, la Vista son los datos "qué" se van a representar y el "cómo" queda relegado a las plantillas (templates). El Controlador está considerado como el propio *framework*, es decir, toda la maquinaria encargada de enlazar y hacer funcionar todos los componentes del sistema.

1.7. Apache

Apache [10] es un servidor web HTTP de código abierto para los principales entornos, incluyendo UNIX (BSD, Linux, etc.), Windows y Mac OS. Actualmente se desarrolla dentro del proyecto HTTP Server de la *Apache Software Foundation*. Su origen se remonta hacía 1994, cuando su diseñador, Robert McCool, abandonó el desarrollo del servidor web NCSA quedando estancado el proyecto. Un grupo de programadores decidió retomar dicho proyecto y en 1995 fundaron el grupo Apache. Utilizando como base de trabajo el servidor NCSA, corrigieron sus errores y añadieron mejoras, publicaron la primera versión de Apache.

Entre las principales características de este servidor destaca su gran modularidad. Tiene presente un *core* que lo provee de las funcionalidades básicas, permitiendo ampliar sus capacidades mediante la inclusión de módulos externos. Actualmente están disponibles multitud de módulos para su instalación y utilización dentro del servidor. Además estos se pueden crear, mediante la programación en C y Perl, para que resuelvan determinados problemas.

También puede dar soporte, mediante la utilización de módulos externos, a la mayoría de lenguaje de programación como Python, Perl, PHP, Java, etc. Apache es capaz alojar aplicaciones creadas en estos lenguajes y servir contenido dinámico de una forma nativa y con una ejecución más rápida que otros *scripts* CGI (*Common Gateway Interface*).

Otra característica fundamental es la gran información que puede consultar el administrador mediante la creación y gestión de logs. De esta manera, el administrador tiene conocimiento de los accesos y errores producidos, teniendo un mayor control de los sucesos que ocurren en el servidor web.

Los principales problemas que se le achaca son su complejidad en la configuración para hacerlo funcionar y su considerable curva de aprendizaje. Estos inconvenientes pueden hacer que los usuarios menos expertos desistan en utilizarlo. A pesar de esto, Apache lleva bastante tiempo siendo el servidor web más popular en Internet y se estima que casi un 58 % de los sitios web del mundo lo utilizan. [11]

1.8. PostgreSQL

PostgreSQL [12] es un sistema de gestión de datos objeto-relacional, libre y publicado bajo licencia BSD. Se trata de un sistema multiplataforma porque se

encuentra disponible para la mayoría de sistemas operativos, incluyendo UNIX, Windows y Mac OS. Fue creado en 1985 por Michael Stonebraker basándose en las ideas del proyecto Ingres, el cual también lideró. Desde 1996 hasta la actualidad, la comunidad PGDG (*PostgreSQL Global Development Group*) es la encargada de evolucionar y mantener este sistema. En rasgos generales, este gestor es ideal para el tratamiento de gran cantidad de datos y su comportamiento es excepcional cuando una gran números de usuarios acceden al sistema. Por estos motivos es una de las bases de datos más potentes y robustas del mercado. A continuación se citan algunas características y funciones que soporta.

Es una base de datos compatible con ACID (*Atomicity, Consistency, Isolation and Durability*). Esta norma especifica que características tienen que cumplir las transacciones para que sean fiables y seguras. Las propiedades de ACID [13] están definidas en el ISO/IEC 10026-1:1992 sección 4.

Mediante el sistema MVCC (*Multi-version Concurrency Control*), mantiene copias de los datos y realiza un control de concurrencia entre las distintas versiones que se van escribiendo. Esta solución busca acelerar los procesos de escritura y evitar bloqueos innecesarios de tablas o filas. De esta forma no es necesario que los procesos lectores tengan que esperar a que se terminen las escrituras en la base de datos.

También admite la creación y ejecución de funciones escritas en varios lenguajes de programación, en los que se incluyen: C, C++, Java, Python y Ruby. Esto es una gran ventaja ya que permite aprovechar la potencia de otros lenguajes para realizar tareas más o menos complejas. Además incluye de forma nativa el lenguaje procedimental PL/pgSQL, similar al lenguaje PL/SQL de Oracle.

PostgreSQL tolera el uso de aplicaciones o módulos externos para extender su funcionalidad, como pueden ser OpenFTS, PL/Proxy y PostGIS. Este último se ha utilizado en este proyecto dar soporte a objetos geográficos en la base de datos. Por último, las últimas versiones de este sistema de gestión datos cumple con el estándar ANSI-SQL:2008, por lo que soporta gran parte de los tipos de datos, características y operaciones que recoge este estándar y sus predecesores.

Debido a su conjunto de características y apreciables ventajas, PostgreSQL es mundialmente utilizado en sectores tan importantes como el gubernamental, financiero, tecnológico y educativo, entre otros. [14]

1.9. LibreGeoSocial

LibreGeoSocial [15] es una red social móvil con una interfaz de realidad aumentada. Ha sido desarrollado por el departamento GSyC (Grupo de Sistemas y Comunicaciones), adscrito a la Escuela Técnica Superior de Ingeniería de Telecomunicaciones de la universidad Rey Juan Carlos. Se trata de un proyecto de software libre y código abierto, por lo que su código fuente es totalmente accesible y se puede utilizar para crear nuevas aplicaciones, como turismo [16], educación, juegos geolocalizados, etc.

LibreGeoSocial tiene características muy importantes e interesantes. Haciendo uso de las coordenadas latitud y longitud, todos los nodos que componen esta red social están geolocalizados. Además, según la altitud de estos nodos, los usuarios podrán obtener diferente información del mismo punto.

Otra característica a destacar es la posibilidad de interactuar con una interfaz de realidad aumentada, pudiendo añadir y visualizar tags en objetos a través del uso de la cámara del terminal móvil. De esta manera cuando un usuario se aproxime sobre un punto ya etiquetado, el sistema le avisará, mediante un sonido, que puede consultar la información dejada por otro miembro. Y por supuesto, también tiene las funciones típicas de una red social, permitiendo el envío de mensajes o compartir contenido multimedia (audio, imágenes o vídeo) con nuestros contactos.

A continuación, se detallan los principales bloques de esta aplicación:

- **LibreGeoSocial:** Se trata de un *framework*, desarrollado en Django y Python, que proporciona los instrumentos necesarios para la creación y gestión de redes sociales con nodos geolocalizados.

- **LibreGeoSocialApp:** Es la parte que permite utilizar LibreGeoSocial en terminales móviles Android. Utiliza los recursos de estos dispositivos, como el GPS y la cámara, para poder llevar a cabo todas las funcionalidades que brinda.

Por último, mencionar que ha sido necesario estudiar el modelo de datos, plasmado en la figura 1.5, para aprovechar mejor sus características y desarrollar de una manera más eficiente este PFC.

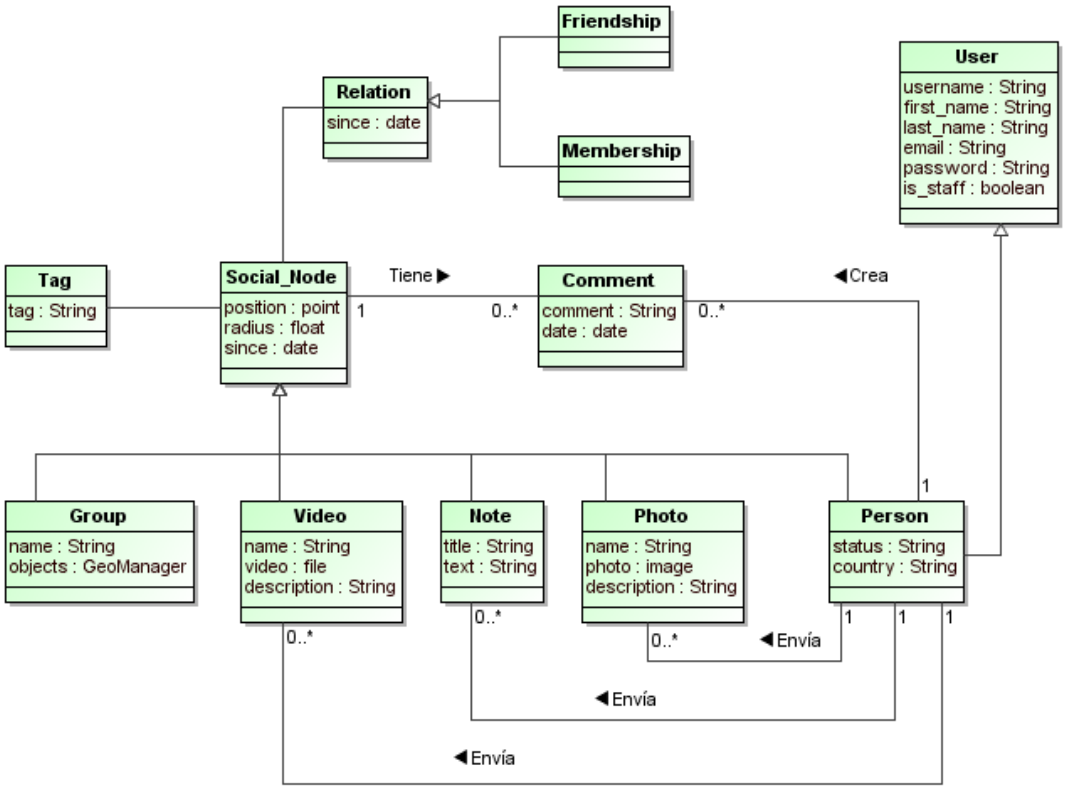


Figura 1.5: Diagrama UML con el modelo de datos de LibreGeoSocial.

1.10. Android

Android [17], [18], [19] es una plataforma software, destinada a dispositivos móviles (*smartphone* o *tablets*), que engloba un sistema operativo y las principales aplicaciones para el usuario. Es una plataforma de software libre y código abierto. Los desarrolladores cuentan de forma gratuita con un SDK que proporciona las herramientas y APIs necesarias para desarrollar, de una manera fácil y cómoda, aplicaciones usando el lenguaje de programación Java. Estas aplicaciones se ejecutan en la máquina virtual, denominada Dalvik, que se encuentra integrada en el sistema.

Su inicio se remonta al año 2003, cuando Andy Rubin, Rich Miner, Nick Sears y Chris White fundaban Android Inc., una compañía con sede en California y cuya principal actividad se centraba en el desarrollo de software para teléfonos móviles. Dos años después, Google empezó a adquirir empresas con un gran futuro prometedor, entre las que se encontraba Android Inc., contratando a sus fundadores con Andy Rubin como líder del proyecto. Desde entonces, se acrecentaron los rumores de una posible incursión de Google en el mundo de la telefonía móvil.

El 5 de noviembre de 2007 fue la fecha clave para la creación de Android. Se fundaba la OHA (*Open Handset Alliance*), un conglomerado de fabricantes de hardware, software y operadores de servicio entre la que destacan HTC, Samsung y Qualcomm, entre otras más. El principal objetivo de la OHA, liderada por Google, es el desarrollo de estándares abiertos para dispositivos móviles. Ese mismo día se disiparon todo tipo de rumor y se anunció lo que hoy conocemos como Android, una plataforma bajo licencia Apache (libre y de código abierto) y basada en el sistema operativo Linux.

Pocos días después, concretamente el 12 de noviembre, se liberaba la primera versión del SDK para que los programadores empezasen a crear aplicaciones para este nuevo sistema.

1.10.1. Características

A continuación se listan las principales características de Android. Su conocimiento es esencial para aprovechar el gran potencial que ofrece en el desarrollo de programas.

- Se facilita la reutilización de componentes gracias al *framework* de aplicaciones de la arquitectura. Este hecho junto a la gran API proporcionada por el SDK, reduce el esfuerzo en el desarrollo de nuevos bloques.
- Mediante el uso de la máquina virtual Dalvik, optimizada específicamente para dispositivos móviles, se ejecutan las aplicaciones en el sistema.
- Navegador web integrado basado en el motor WebKit. Este motor también es usado en el entorno Mac OS X, concretamente en el navegador Safari.
- Uso de librerías específicas para la representación eficiente de gráficos 2D y 3D basado en la especificación OpenGL ES.
- Almacenamiento estructurado de la información mediante la base de datos liviana SQLite.
- Gran soporte de formatos multimedia (H.264, MPEG-4, WebM, ACC, MP3, BMP, JPG, PNG, etc.).
- Dependiendo del hardware de cada dispositivo, se soporta las principales tecnologías de conectividad: GSM, UMTS, LTE, Bluetooth, Wi-Fi, etc.

- Soporte para hardware opcional como la cámara, GPS, brújula o sensores de proximidad.
- Un completo entorno de desarrollo, en el que se incluye un emulador de dispositivo, herramientas para la depuración y un *plugin* para facilitar la creación de proyectos en el IDE Eclipse [20].

1.10.2. Arquitectura

La arquitectura del sistema operativo Android está compuesta por un conjunto de capas, cada una de estas utiliza los servicios de las anteriores, y ofrece los suyos a las capas de niveles superiores. En la figura 1.6 se muestra los niveles de la arquitectura, los cuales se describen a continuación:

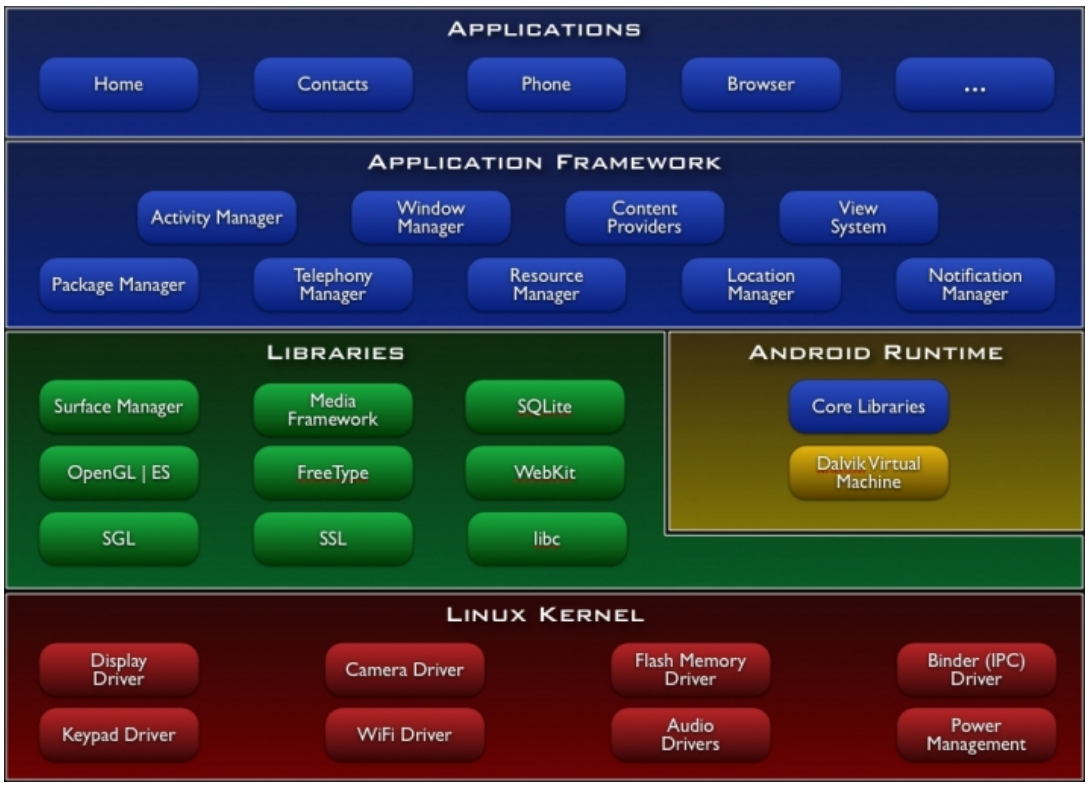


Figura 1.6: Diagrama de la arquitectura del sistema Android.

Aplicaciones

Son el conjunto de APIs que permite a los desarrolladores la creación de aplicaciones en Android con las funcionalidades básicas que pone a su disposición

(gestores de ventana, de ubicación, de notificación, de telefonía, etc.). Como ya se indicó en el conjunto de características, este *framework* facilita la reutilización de componentes, de tal manera que cualquier aplicación, siempre y cuando respete las restricciones de seguridad, puede utilizar las funcionalidades de otra.

Framework de Aplicaciones

Esta capa se sitúa justamente por encima del *kernel*. Android dispone de un conjunto de bibliotecas programadas en C/C++ y se compilan según las características hardware de cada terminal. Son utilizadas por los componentes del sistema y sus capacidades son ofrecidas al desarrollador a través del *framework* de aplicaciones.

Runtime de Android

Formando parte de la anterior capa, también contiene un núcleo de bibliotecas que proporcionan la mayor parte de las funcionalidades disponibles en las librerías del lenguaje Java así como otras específicas de Android. El principal componente del Runtime es la máquina virtual Dalvik, encargada de ejecutar las aplicaciones del sistema. Cada aplicación ejecuta su propio proceso, lo que también implica una instancia de Dalvik. Por este motivo, Google tuvo que reescribir el código de la máquina virtual para que un dispositivo pudiese ejecutar, de una manera eficiente, varias instancias de la misma. No es compatible con el bytcode de Java (.class) generado, por lo estos archivos tienen que ser transformados por el SDK al formato Dalvik Executable (.dex).

Núcleo de Linux

Se utiliza la versión 2.6 del núcleo de Linux y sirve como abstracción entre los elementos físicos del dispositivo y el resto de capas de la arquitectura, de tal manera que no es necesario conocer las características del hardware instalado para realizar una aplicación. Esto se consigue mediante los drivers o controladores incluidos en esta capa, posibilitando que cualquier componente del terminal pueda ser utilizado por las llamadas de la aplicación. Además de proporcionar estos controladores, el núcleo se encarga de controlar partes tan importantes como la gestión de la batería, memoria, seguridad o procesos del sistema, entre otros.

1.10.3. Componentes de una aplicación

Existen cuatro componentes esenciales para la construcción de una aplicación en esta plataforma. Cada uno tiene un propósito diferente y un ciclo de vida que define como se crean y se destruyen. Estos componentes junto a los permisos u otros elementos fundamentales para el buen funcionamiento de la aplicación, tienen que aparecer en el fichero *AndroidManifest.xml*. Debido a la importancia de este fichero se explicará con detalle, en el subapartado 1.10.4, su finalidad y los diferentes elementos que puede contener.

Cabe destacar que en el desarrollo de una aplicación Android no tienen que aparecer todos los componentes, pero sí es seguro que algunos de ellos estarán presentes.

Activities

Se podría considerar como uno de los más importante y utilizado en una aplicación. Se implementa mediante la clase *Activity* y es el encargado de representar los elementos gráficos en la pantalla para que el usuario pueda interactuar y realizar las acciones.

La mayoría de aplicaciones hacen uso de multitud de actividades pero el desarrollador tiene que indicar cuál de estas será la principal, y por tanto, la primera pantalla que se visualizará. La invocación o el paso de una actividad a otra se realiza mediante la llamada al método *startActivity()*, pasándole como parámetro un *Intent* que especifica la actividad que se quiere comenzar.

Este componente sigue un ciclo de vida desde que se crea hasta que se destruye. Durante este ciclo, la actividad transita entre diferentes estados (activo, pausado o parado) mediante la ejecución de determinados métodos. Android permite redefinir estos métodos para que el desarrollador decida qué acciones se van a realizar entre estas transiciones. En la figura 1.7, se observa los caminos que una actividad puede tomar entre los diferentes estados. Los rectángulos representan las llamadas a los métodos.

Analizando el diagrama de la figura anterior, se obtienen las siguientes conclusiones:

- El ciclo completo de la actividad se produce entre las llamadas a los métodos *onCreate()* y *onDestroy()*, definiendo su inicio y final respectivamente.

- El tiempo que es visible la actividad transcurre entre los métodos *onStart()* y *onStop()*. En este periodo, el usuario puede ver la actividad e interactuar con esta. Puede darse el caso de que se pierda su foco, siendo relegada a un segundo plano. Estos métodos pueden ser llamados múltiples veces durante su vida.

- La actividad se encuentra en primer plano entre las llamadas a los métodos *onResume()* y *onPause()*. Durante este momento, la actividad es visible y el foco de la acción se encuentra sobre esta, posibilitando una total interacción por parte del usuario.

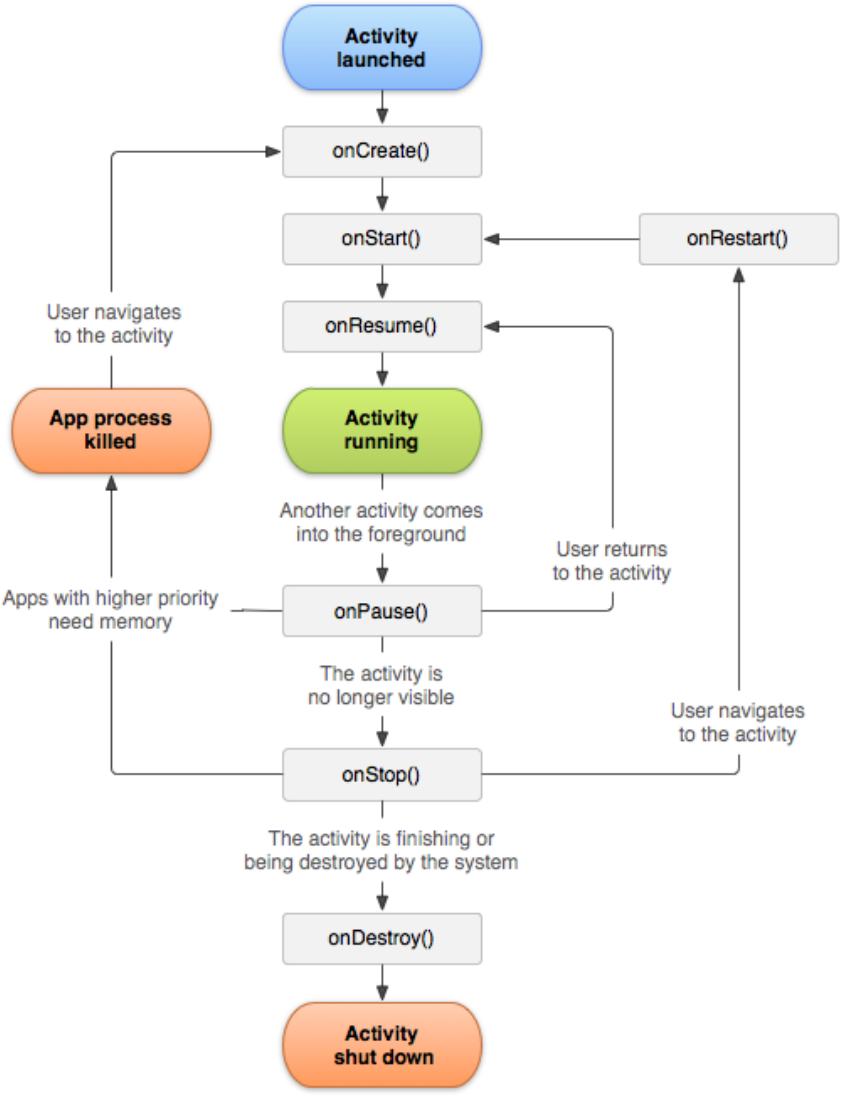


Figura 1.7: Diagrama de flujo con el ciclo de vida de una *Activity*.

Services

Un servicio es un componente que corre en segundo plano (*background*) realizando operaciones o tareas para otros procesos, durante un periodo de tiempo más o menos prolongado, mientras otras aplicaciones se encuentran activas en la pantalla del dispositivo. A diferencia del componente anterior, los servicios no proporcionan una interfaz con la que el usuario pueda interaccionar. Este componente se implementa a través de la clase *Service* y pueden ser lanzados por otros componentes, como por ejemplo una *Activity*. Por defecto, un servicio corre en el hilo principal de la aplicación, por lo que si hace un uso intensivo de la CPU o realiza operaciones bloqueantes (tareas de comunicación o reproducir música), puede ocasionar que la aplicación no responda. Para reducir este riego se recomienda crear un hilo propio para el servicio.

Content providers

Mediante los proveedores de contenido, cualquier aplicación Android puede almacenar información en un fichero, en una base de datos SQLite o en cualquier sistema de almacenamiento que la aplicación tenga acceso. Este componente también proporciona los métodos necesarios para permitir a otros programas leer o modificar esos datos previamente almacenados. Un claro ejemplo donde Android utiliza este tipo de componentes es en la gestión de la agenda telefónica. Si una aplicación tiene los permisos adecuados, puede utilizar un *ContetProvider* específico para leer o modificar un contacto en particular.

Broadcast receivers

Está destinado a detectar y reaccionar ante anuncios producidos en el sistema (batería baja, mensaje recibido, captura de una imagen, etc.) o por otras aplicaciones. Para que un programa reaccione ante un determinado evento, antes tiene que registrarlo en su *BroadcastReceiver*. Al igual que un servicio, este no muestra una interfaz al usuario pero puede alertarle del evento producido creando una notificación en la barra de estado del terminal. Se implementa a través de la clase *BroadcastReceiver*.

1.10.4. Android Manifest

Todo proyecto Android necesita un elemento fundamental y se trata del archivo *AndroidManifest.xml*. Este fichero es generado automáticamente por el *plugin* de Eclipse en la raíz del proyecto y representa la información que necesita conocer el sistema para poder ejecutar el código de la aplicación.

El fichero, desarrollado en el lenguaje XML, describe el comportamiento de cada componente que forma la aplicación, ya mencionados anteriormente. Además se especifican que permisos necesita nuestra aplicación para funcionar adecuadamente, así como los permisos requeridos a otras partes del sistema para que puedan interactuar con sus componentes. A través del nombre del paquete Java declarado en este manifiesto, la aplicación queda unívocamente identificada en nuestro terminal móvil. Otros aspectos que abarca este archivo son la definición del nivel mínimo y máximo de la API de Android requeridos y el listado de las librerías que son enlazadas al programa.

Un error muy común es intentar ejecutar un componente que no ha sido declarado en el manifiesto, en cuyo caso se forzará el cierre de la aplicación alertando al usuario de lo sucedido.

1.10.5. Google Play: el sustituto del Android Market

Google Play es la tienda de software en línea creada por Google que comercializa aplicaciones, música, películas y libros electrónicos; accesibles desde cualquier dispositivo Android o la Web. Los desarrolladores poseen libertad para publicar sus aplicaciones, únicamente tienen que registrarse, subir y comentar el contenido para finalmente publicarlo.

Este servicio, que inicialmente solo distribuía aplicaciones, se puso a disposición de los usuarios en octubre de 2008, bajo el nombre de Android Market. En marzo de 2012, Google anunció en su blog oficial [21] el cambio de nombre de esta plataforma por el de Google Play, manteniendo los mismos servicios que se ofrecían. El principal motivo del cambio es aumentar su presencia en la parte de ventas electrónicas y mejorar la competencia frente a las soluciones propuestas por sus rivales Apple y Amazon.

El crecimiento de esta plataforma ha sido increíble, según el análisis realizado por la compañía Distimo [22], tiene publicada más de 400.000 aplicaciones a finales del año 2011. El mismo estudio afirma, que en un breve periodo de tiempo, Google Play alcanzará la cifra de 100.000 desarrolladores que han difundido

algún contenido en la tienda. En la figura 1.8 se observa el crecimiento de las aplicaciones, ya sean gratuitas o de pago durante todo el año 2011, así como la evolución de los desarrolladores activos.

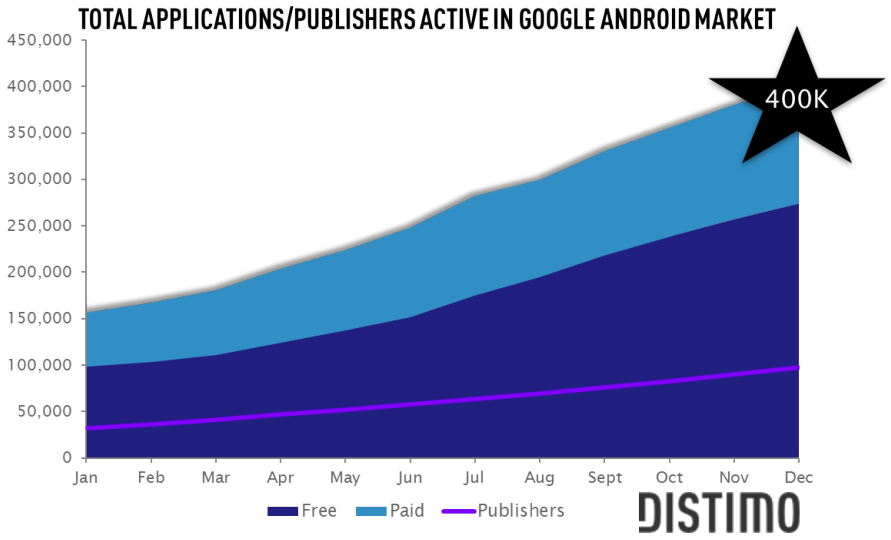


Figura 1.8: Gráfica de la evolución del Android Market en 2011.

Apple es uno de los rivales más duros que tiene Android, ya sea a nivel de ventas de teléfonos, gracias a su iPhone, o a su servicio App Store. Sobre la misma fecha que Google Play alcanzó su gran hito, la App Store contaba con más de 500.000 aplicaciones [23]. Con estas cifras está claro que Android tiene una gran competencia en multitud de sectores, pero es evidente que vive un gran momento y se prevé un futuro prometedor, debido entre otros factores, a su gran evolución y que es una de las plataformas favoritas de los desarrolladores.

1.10.6. La fragmentación

Google se preocupa de mantener su sistema operativo, lanzando periódicamente versiones que mejoran e introducen nuevas características. Visto así es una buena noticia para desarrolladores y clientes, ya que les permite aprovechar los últimos avances tecnológicos de sus terminales. El problema surge cuando las nuevas actualizaciones no sustituyen a las anteriores, dando lugar a una gran variedad de versiones funcionado al mismo tiempo. Este fenómeno se le denomina fragmentación y Android lo está empezando a sufrir.

En la figura 1.9 se puede observar la gráfica con la distribución histórica de los sistemas que se encuentran actualmente en uso. El eje-y representa el porcentaje

de dispositivos compatibles con una determinada versión y el eje-x refleja la línea de tiempo en periodos de 14 días.

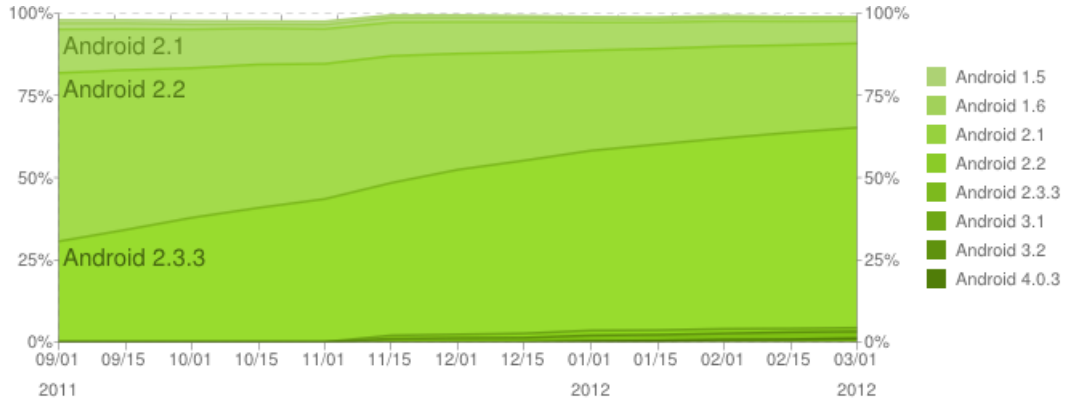


Figura 1.9: Gráfica de la distribución de las diferentes versiones de Android.

Como se aprecia, las versiones más antiguas (2.x) son las más utilizadas y acaparan casi la totalidad del mercado. En el otro lado se encuentran las últimas actualizaciones (3.x y 4.x) que llevando ya varios meses lanzadas, todavía cuentan con una cuota ínfima. Como ya se ha comentado, Android cuenta con un gran número de desarrolladores pero la fragmentación puede repercutir en la buena salud de la que goza. El principal motivo de esta afirmación es que los mismos desarrolladores tienen que preocuparse de crear y adaptar sus aplicaciones para una multitud de versiones, pudiendo ocasionar desinterés por la plataforma.

Cada vez más personas se hacen eco de este problema, por lo que Google debería remediarlo y cuanto antes mejor. Una posible solución pasaría por obligar a compañías telefónicas y fabricantes a no interferir en las actualizaciones y acogerlas en el menor tiempo posible.

Capítulo 2

Objetivos

En este capítulo se explica de una forma general el problema presentado junto a los principales objetivos que se pretenden lograr a la finalización de este proyecto. Por último, se expone la metodología utilizada detallando el trabajo realizado en cada fase.

2.1. Descripción del problema

El problema que se nos presenta es diseñar e implementar una arquitectura cliente-servidor. Por un lado, el sistema posibilitará la creación y gestión de los elementos necesarios para el transcurso de una partida. También permitirá realizar el seguimiento y la consulta del estado de cada recurso. Todas estas acciones se llevarán a cabo a través de un sitio web.

Otro punto importante será la participación de los usuarios mediante un aplicación, concretamente un juego geolocalizado instalado en los dispositivos móviles Android. Estos tipos de clientes harán uso de las principales características que ofrecen estos terminales (conectividad 3G, GPS, pantalla táctil, etc.) para llevar a cabo las funciones disponibles, introducidas en el apartado 1.3, de la aplicación.

En definitiva, se pretende desarrollar un sistema robusto, fiable y que sea capaz de responder al continuo tráfico de datos generado entre los clientes y el servidor.

2.2. Objetivos del proyecto

Una vez presentado, de una forma bastante general, el problema a resolver en este proyecto, a continuación se listan los objetivos marcados para solucionarlo de una manera eficiente.

- **Análisis de las tecnologías utilizadas:** El primer paso antes de diseñar e implementar el sistema, es el estudio de las tecnologías y herramientas utilizadas en este PFC. Para lograr este fin, ha sido de gran ayuda las diversas páginas web, libros, manuales y en general cualquier tipo de documentación que proporciona información sobre el comportamiento, arquitectura y principales ventajas de estas tecnologías.

- **Desarrollo del servidor:** El servidor será uno de los componentes esenciales del sistema. Mediante la utilización de un navegador web, un usuario con permisos de administrador, podrá interactuar con una interfaz gráfica para crear las partidas y sus recursos asociados (equipos, territorios y tropas). Así como llevar el control de la partida en curso y establecer comunicación con los participantes a través del intercambio de mensajes.

  Otro objetivo es el uso de un gestor de BBDD para almacenar todos los datos generados y ponerlos a disposición de los clientes cuando los soliciten.

- **Desarrollo de la aplicación Android:** Una vez especificadas las necesidades del sistema, se debe desarrollar una aplicación que esté en permanente comunicación con el servidor y obtenga toda la información necesaria para su funcionamiento. De tal manera que el usuario pueda interactuar con la interfaz gráfica de la aplicación y participar en la partida junto a los demás jugadores.

- **Probar el funcionamiento del sistema:** Finalmente será necesario diseñar una serie de pruebas en un escenario real, con voluntarios y con dispositivos físicos. El objetivo es comprobar el funcionamiento tanto del servidor como de la aplicación y en caso de detectar algún problema, buscar sus causas y corregirlo. También interesa recoger la aceptación y las opiniones realizadas por los participantes, siendo clave esto último para mejorar el producto realizado.

2.3. Metodología empleada

Las metodologías de desarrollo software proporcionan un conjunto de procedimientos y técnicas que posibilitan crear un nuevo software siguiendo un proceso formal. El principal objetivo es conseguir un producto de calidad que cumpla con las necesidades del usuario. El Proceso Unificado [24] es un marco de trabajo que reúne estas características y que permite transformar los requisitos iniciales del usuario en el artículo final. A lo largo de la carrera, este tipo de proceso ha sido estudiado y puesto en práctica por lo que ya se tiene un conocimiento base, esto ha sido fundamental para su elección entre las diferentes metodologías existentes en la actualidad.

El Lenguaje Unificado de Modelado (UML) es utilizado en esta metodología para diseñar los diferentes diagramas que guiaran el desarrollo del sistema. Con UML se consigue un mayor rigor en la especificación y plena independencia de los diseños con el lenguaje de implementación. A continuación se detallan los tres aspectos claves del Proceso Unificado.

- **Dirigido por casos de uso:** Los casos de uso representan los requisitos funcionales del sistema y son obtenidos cuando el usuario interacciona con una funcionalidad de este. También guían el proceso de desarrollo, es decir, son el hilo conductor en el diseño, implementación y pruebas del sistema construido. Además, el modelo de casos de uso está íntimamente relacionado con la arquitectura del sistema, evolucionan de forma paralela mientras avanza el ciclo de vida.

- **Centrado en la arquitectura:** La arquitectura de un software incluye los aspectos estáticos y dinámicos del sistema. Queda definida por las necesidades de los usuarios y otros factores externos como: limitaciones hardware, sistema operativo donde se va a implantar, aspectos legales, requisitos no funcionales, etc. Tiene que haber un equilibrio entre la forma y funcionalidad del sistema, de tal manera que la arquitectura permita el desarrollo de los casos de uso sin ningún tipo de problema.

- **Iterativo e incremental:** Pretende reducir el esfuerzo que implica el desarrollo de un sistema relativamente grande, dividiéndolo en pequeñas etapas o fases. Cada fase está compuesta por iteraciones que representan un flujo de trabajo (requisitos, análisis, diseño, implementación y prueba). Además, cada iteración se basa en el artefacto producido por la anterior con lo que se va consiguiendo pequeños incrementos que forman el producto final.

En la figura 2.1 se visualiza la estructura del Proceso Unificado. Como se puede comprobar, la intensidad de cada flujo de trabajo depende de la fase en la que se encuentre el desarrollo. Seguidamente se explica de forma general las labores que se realizan en cada etapa.

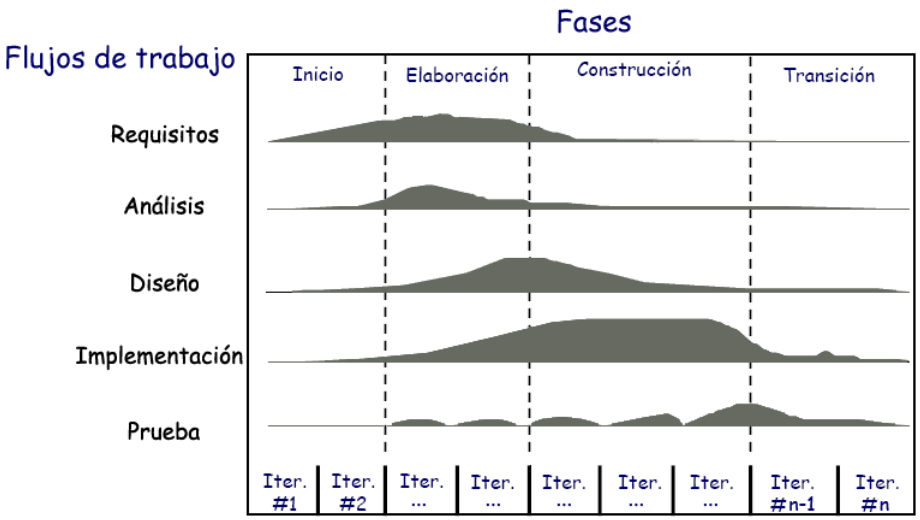


Figura 2.1: Diagrama del ciclo de vida del Proceso Unificado.

Durante la **fase de inicio** se describe la finalidad del sistema. Para esto se definen los primeros casos de uso, se realiza una aproximación de la arquitectura y se identifican los principales riesgos que pueden surgir y hacer peligrar el proyecto.

En la **fase de elaboración** se especifica con mayor detalle los diagramas de casos de uso y se afianza la arquitectura del sistema. Es el momento de planificar las tareas y recursos necesarios para una finalización exitosa del proyecto.

En la siguiente **fase de construcción** se obtiene el producto final, añadiendo nuevas funcionalidades sobre la arquitectura. Aunque el producto ya se encuentra acabado y pueda ser entregado al usuario final, es posible que contenga errores.

Por último, durante la **fase de transición**, usuarios con experiencia prueban el producto para detectar y corregir los errores más graves. Una vez solventadas las deficiencias el producto se pone a disposición de la comunidad de usuarios.

Cabe mencionar que se ha reducido el número de diagramas utilizados para modelar el comportamiento del sistema. Para suplir esta falta y con el objetivo de maximizar la comprensión, en el siguiente capítulo se ha realizado una detallada explicación de los componentes y funcionalidades presentes en el proyecto.

Capítulo 3

Descripción Informática

Una vez expresadas las nociones más significativas y explicar los objetivos que se pretenden conseguir a lo largo de este proyecto, es el momento de detallar el proceso de desarrollo de las distintas partes que constituyen el sistema. En primer lugar, se exponen los requisitos para posteriormente realizar el análisis y diseño del sistema. Por último se describirá detalladamente la implementación de cada parte, obteniendo una visión más clara y definida.

3.1. Especificación de requisitos

Antes de poder diseñar e implementar cualquier software, es necesario comprender de una manera clara y meridiana el comportamiento y restricciones que va a tener. Para este fin se van a especificar los requisitos del servicio web y de la aplicación Android. Se puede distinguir dos tipos de requerimientos a la hora de realizar su especificación: los requisitos funcionales (RF) son aquellos que determina el comportamiento interno de nuestro sistema y los requisitos no funcionales (RNF) definen propiedades y restricciones de la aplicación, complementando a los anteriores.

Gracias a las diferentes reuniones que se llevaron a cabo con el tutor al principio del proyecto, se pudieron identificar y detallar algunos de los requerimientos que se van a precisar a continuación.

3.1.1. Requisitos funcionales

Servicio web

- **RF - 1. Control de Acceso:**
  Todo usuario podrá autentificarse en el servicio web con su nombre y contraseña. En caso de hacerlo, se le tratará como un cliente u organizador según los privilegios de su cuenta.

- **RF - 2. Gestionar recursos:**
  Un organizador, mediante el cumplimiento de formularios, podrá crear y modificar partidas, equipos, territorios y tropas. También se le da la posibilidad de eliminar cualquier elemento mencionado anteriormente.

- **RF - 3. Visualizar recursos:**
  Los usuarios podrán listar y obtener información detallada de cualquier partida y elementos asociados.

- **RF - 4. Servicio de mensajería:**
  Los organizadores serán capaces de comunicarse con los equipos a través del intercambio de mensajes.

- **RF - 5. Seguimiento de la partida:**
  Los usuarios conseguirán monitorizar cualquier partida en tiempo real. Para esta función es necesario la utilización de un Google Maps.

- **RF - 6. Validación de formularios:**
  Los campos de los formularios se comprobarán para detectar posibles errores y evitar fallos en el sistema. En caso de error, el organizador debe ser informado.

- **RF - 7. Notificación de errores:**
  Cualquier error que se produjese en el sistema, será informado al usuario para su posible corrección.

Aplicación Android

- **RF - 8. Selección de recursos:**
  El jugador podrá seleccionar la partida y equipo con el que participará.

- **RF - 9. Estadísticas de la partida:**
  En todo momento, el jugador tendrá la posibilidad de visualizar el tiempo restante, información de su equipo y clasificación de la partida en curso.

- **RF - 10. Gestión del mapa:**
  El usuario podrá acceder al mapa donde se representarán los diferentes elementos geolocalizados (territorios y equipos) de la partida.

- **RF - 11. Acciones de la aplicación:**
  El jugador realizará las siguientes acciones en la interfaz del mapa: conquista y espionaje de territorios rivales, información de sus territorios y equipo, movimientos de tropas y duelos entre los participantes.

- **RF - 12. Zona de acción del equipo:**
  Algunas acciones, como las conquistas y duelos, solo se podrán realizar cuando el equipo se encuentre lo suficientemente cerca del objetivo.

- **RF - 13. Ubicación de los equipos participantes:**
  El jugador desconocerá la localización de los equipos rivales, salvo que estos últimos se encuentren dentro de su radio de acción.

- **RF - 14. Servicio de mensajería:**
  Los jugadores se comunicarán con el organizador y demás equipos participantes, mediante la utilización del servicio de mensajería interno.

- **RF - 15. Menús de ayuda:**
  En las principales pantallas de la aplicación, el usuario podrá consultar el menú de ayuda donde se recogen las acciones disponibles de esa interfaz.

- **RF - 16. Notificaciones de errores:**
  El jugador será informado de los errores, y si es posible, actuará para su corrección.

3.1.2. Requisitos no funcionales

Servicio web

- **RNF - 1. Compatibilidad con diferentes navegadores web:**
  El sitio web tendrá que funcionar correctamente en los principales navegadores web: Mozilla Firefox, Internet Explorer, Google Chrome, etc.

- **RNF - 2. Usabilidad:**
  Los elementos mostrados en la interfaz web tendrán que tener un aspecto y tamaño adecuado para su correcta visualización.

- **RNF - 3. Prestaciones:**
  Se minimizará el tiempo de respuesta del servidor para que la navegación de los usuarios sea cómoda y fluida.

- **RNF - 4. Tolerancia a fallos:**
  Ante cualquier error, se intentará que el servidor web responda adecuadamente.

Aplicación Android

- **RNF - 5. Compatibilidad con dispositivos móviles:**
  La aplicación funcionará en dispositivos móviles Android 1.6 o posteriores.

- **RNF - 6. Elementos visuales de la aplicación:**
  Los diferentes menús, iconos e información que se presentarán al usuario tendrán que tener un aspecto adecuado y amigable.

- **RNF - 7. Precisión en los elementos geolocalizados:**
  Es esencial una buena precisión para el posicionamiento de los territorios y equipos en el mapa.

- **RNF - 8. Rendimiento:**
  Se minimizará el tiempo de respuesta cuando el usuario interaccione con la aplicación.

- **RNF - 9. Tolerancia a fallos:**
  Ante cualquier imprevisto surgido a lo largo de una partida, se intentará que el programa responda adecuadamente y no fuerce su cierre.

3.2. Diagramas de Casos de Uso

Mediante la utilización de los diferentes diagramas que ofrece UML se puede visualizar, especificar y construir el sistema detallado en este documento. Una vez obtenidos los requisitos del servicio web y la aplicación Android, se usarán los diagramas de casos de usos para puntualizar el comportamiento del sistema cuando los principales actores interaccionan sobre este.

En la figura 3.1 se visualiza el diagrama de casos de uso general del cliente web. Los usuarios una vez autentificados y dependiendo de sus privilegios, podrán relacionarse con el sistema efectuando diferentes acciones, como gestionar partidas, visualizar los equipos o realizar un seguimiento de la partida, entre otras.

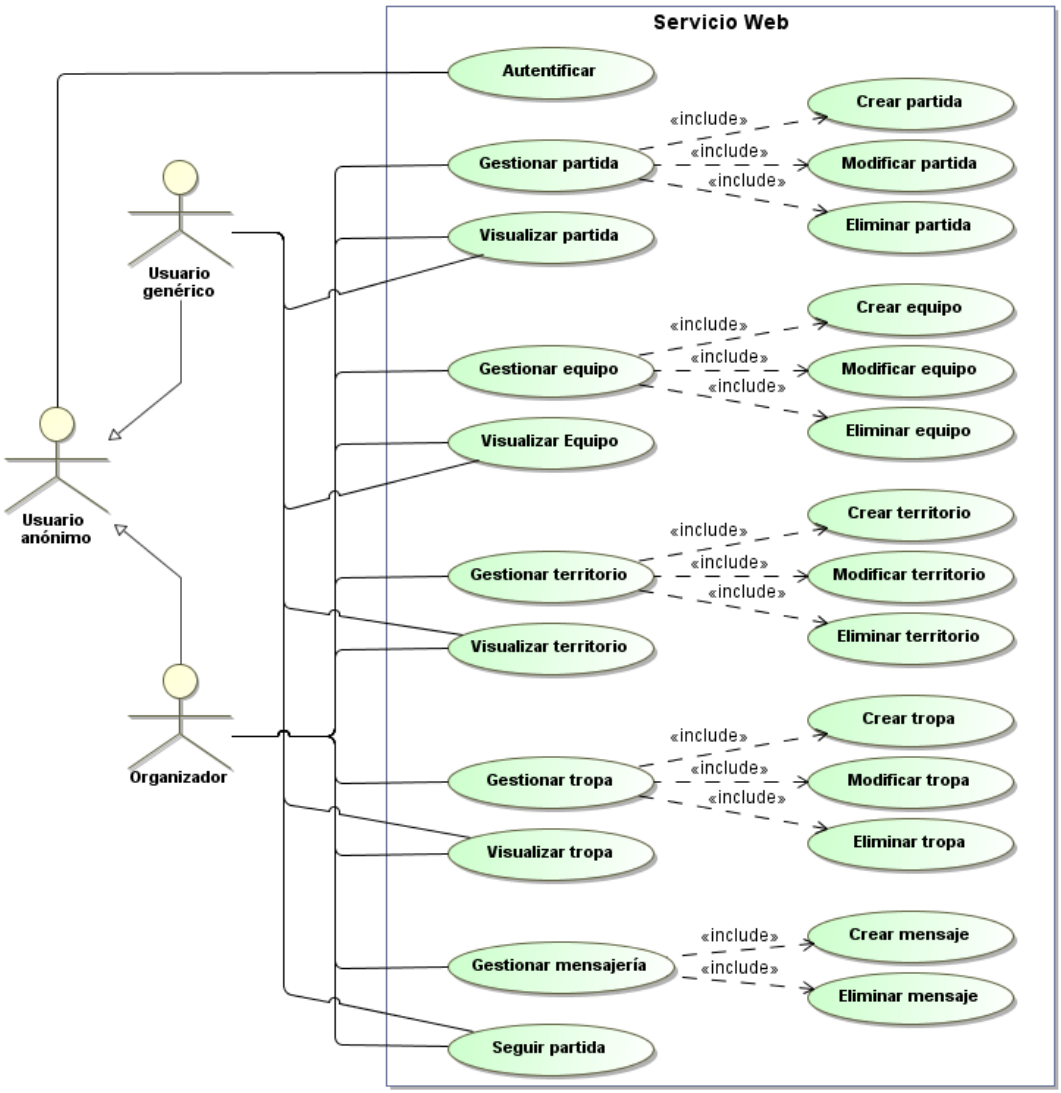


Figura 3.1: Diagrama de casos de uso del servicio web.

La figura 3.2 muestra el diagrama de casos de uso general de la aplicación. En este caso hay un único actor que, mediante el uso del terminal móvil, interactuará con las diversas funciones proporcionadas por el cliente.

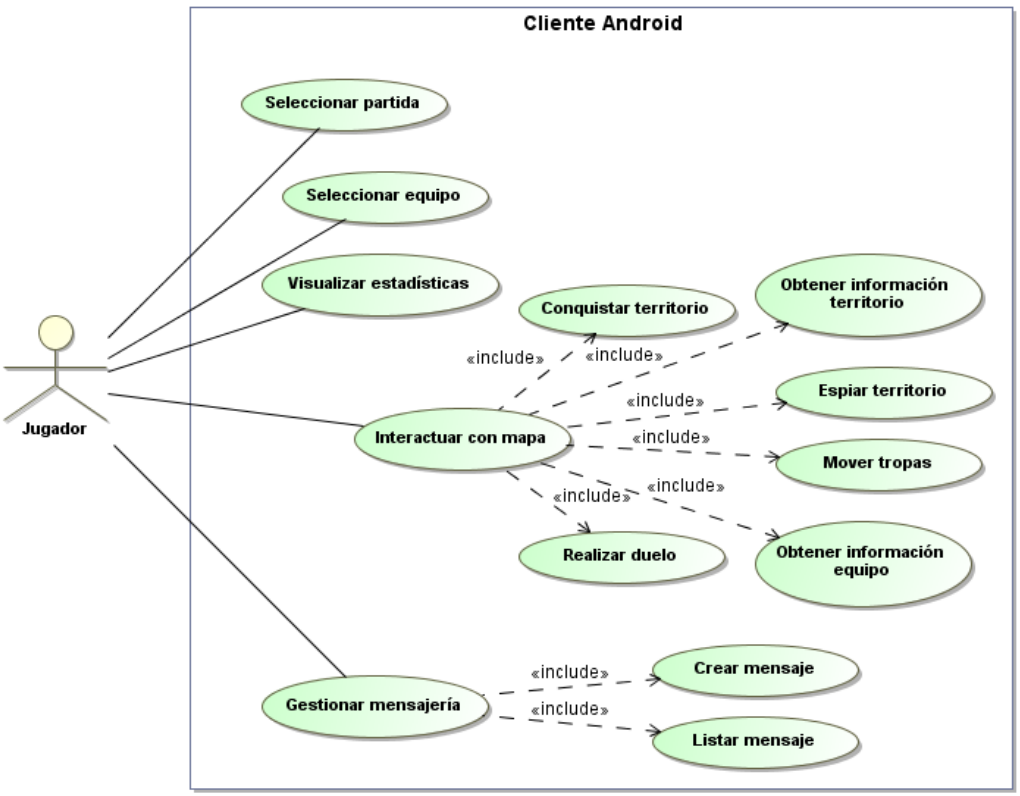


Figura 3.2: Diagrama de casos de uso de la aplicación Android.

3.3. Arquitectura del sistema

La arquitectura de un sistema se encarga de definir los componentes y las interacciones de estos dentro del sistema. La correcta elección de una arquitectura es crítica, porque estará presente durante el desarrollo de la aplicación y posteriormente influirá en su mantenimiento y evolución. Teniendo en cuenta los objetivos y restricciones del proyecto, se ha considerado la arquitectura de tres niveles (presentación, negocio y datos) como la mejor opción.

La arquitectura de tres capas o niveles es una arquitectura cliente-servidor cuyo objetivo principal es la separación entre los conceptos de diseño y la lógica de negocio. Una de las características más relevantes de este modelo es la desconexión entre las capas de presentación y la de datos, permitiendo únicamente las conexiones entre las capas contiguas.

Como el desarrollo se realiza en varias capas independientes, si una de ellas sufre algún cambio, solo es necesario centrarse en ese nivel. Otra ventaja es la posibilidad de realizar el desarrollo en paralelo, abstrayéndose de la implantación

de las demás niveles. En la figura 3.3 se observa la arquitectura del sistema y la comunicación que se produce entre las capas.

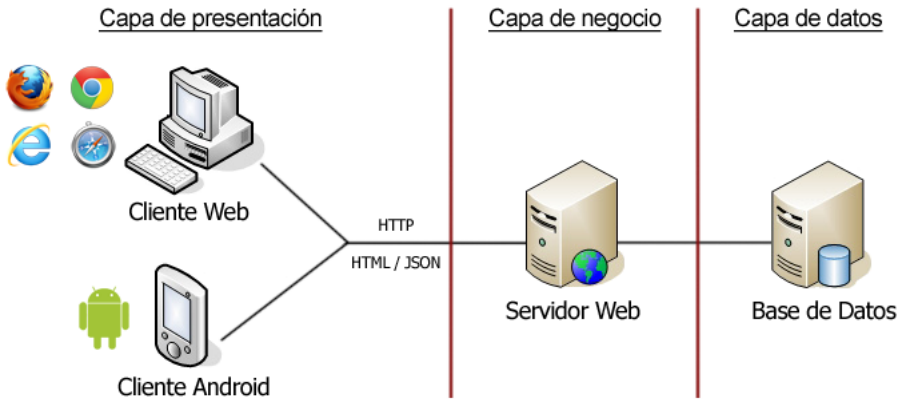


Figura 3.3: Diagrama de la arquitectura del sistema.

A continuación se detallan los diferentes niveles que componen la arquitectura, así como su funcionalidad dentro del sistema:

- **Capa de presentación:** Este nivel se encarga de presentar la GUI al usuario, es decir, muestra la información y recoge los datos introducidos para posteriormente entregarlos a la capa de negocio. Se distinguen dos grupos de usuario que interactuará en esta capa: el primero son los clientes que utilizan un navegador web y el segundo grupo son los clientes que emplean la aplicación para terminales móviles.

- **Capa de negocio:** En este nivel se establecen las reglas que el sistema debe cumplir. Es la encargada de procesar las peticiones del usuario y comunicarse con el servicio de datos, aislándole de la interacción directa con la base de datos. Esta capa reside en el lado del servidor.

- **Capa de datos:** Es la encarga de recuperar, modificar y mantener los datos del sistema según las instrucciones proporcionadas por el servicio de negocio. Al igual que el nivel anterior, también se encuentra alojado en la parte del servidor.

Al residir las capas de negocio y datos en la misma máquina, se consigue que la comunicación con la base de datos sea más rápida y eficaz. De esta manera se provee de un mejor servicio y prestaciones a los clientes.

3.4. Diseño del servidor

En esta sección se tratará el diseño del servidor y las distintas partes que lo conforman. En primer lugar se detallará el modelo de datos que será la base del funcionamiento del sistema, también se describirá la integración del servicio web dentro de la red social LibreGeoSocial, la utilización del patrón MVC (Modelo-Vista-Controlador) en la arquitectura y el diseño de unas URLs amigables siguiendo los principios del estilo arquitectónico REST.

3.4.1. Modelo de datos

Para el diseño de la base de datos relacional [25] del proyecto, se especificará el modelo de datos con el fin de detallar su estructura, es decir, que tipos de datos están incluidos y la relación que tienen los elementos entre sí. Debido a que el proyecto se encuentra integrado en LibreGeoSocial, como se explicará posteriormente, en el modelo de datos se crearán diversos vínculos de herencia con algunas clases de esta red social. Mediante el diagrama UML de la figura 3.4, se presenta dicho diseño incluyendo las siguientes tablas:

- **Orgsanizador:** Guarda información de los usuarios que gestionan las partidas. Mediante la referencia a la tabla *Person* de LibreGeoSocial, cada usuario se le asigna un nombre, contraseña y los permisos adecuados para que se convierta en organizador. También se almacena una referencia a todas las partidas que ha creado.

- **Partida:** Contiene información de todas las partidas. De cada una se obtiene su nombre, lugar de celebración, fecha, duración, cantidad de espías y escolta disponibles inicialmente, una breve descripción y un campo booleano que indicará si la partida se encuentra disponible para ser jugada.

- **Color:** Almacena los colores que estarán disponibles para los equipos en cada evento. Por cada equipo creado se suprimirá la fila que coincida con el color y partida.

- **Equipo:** Almacena los datos de los equipos definidos. Tiene una referencia a un grupo de LibreGeoSocial con el cual se le asociará un nombre y propiedades de geolocalización, cuenta con una referencia a la partida en la que participa. Contiene el atributo color, con el que será identificado durante el

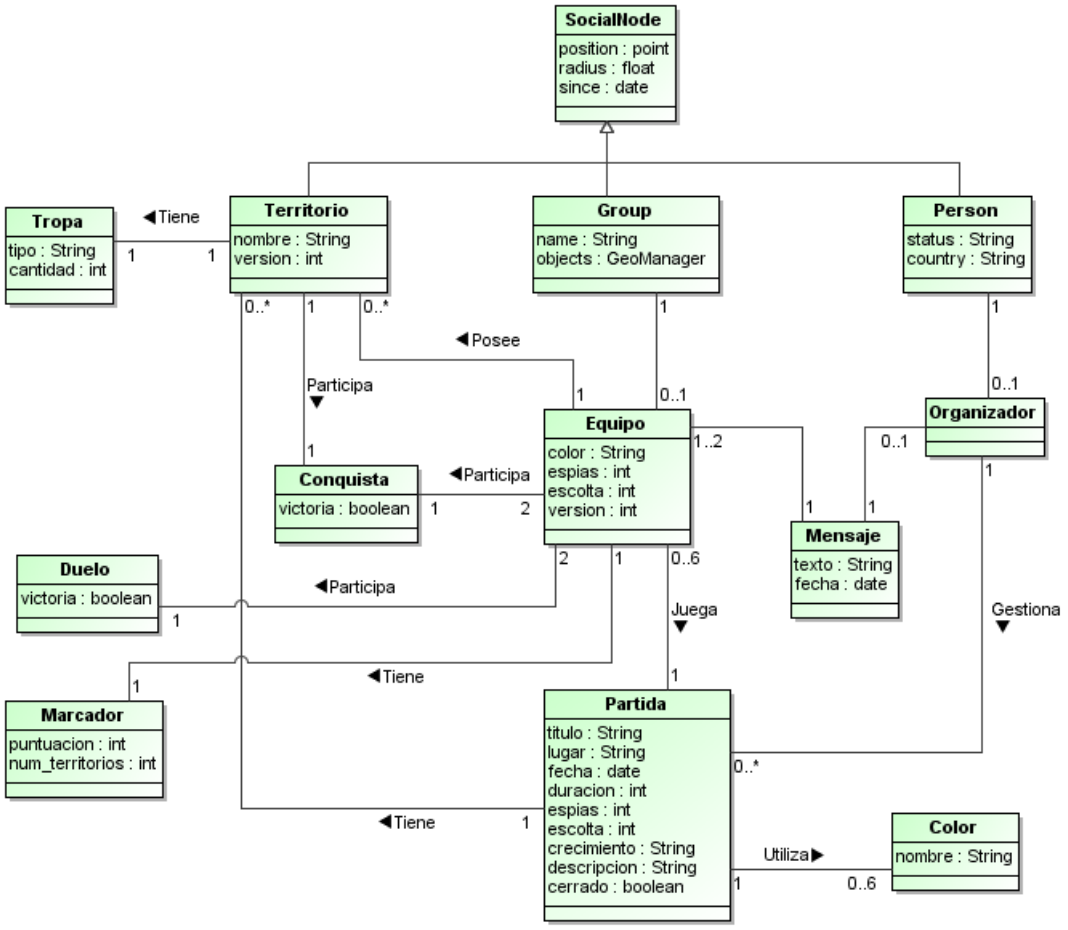


Figura 3.4: Diagrama UML del modelo de datos del sistema.

juego, la cantidad de espías y escolta que dispone. Por último, se encuentra el atributo versión que será utilizado para el control de concurrencia implementado en el servidor.

- **Marcador:** Guarda información relativa a los marcadores de las partidas. Contiene una referencia al equipo que pertenece, también están los campos que recogen la puntuación que ha ido sumando y números de territorios que poseen en ese momento.

- **Territorio:** Esta tabla almacena la información de los territorios creados. Contiene una referencia al equipo que lo posee y en caso de que no tenga ningún propietario, la referencia será nula. También tiene otros campos como el nombre, con el que se identificará en la partida, y la versión para el control de concurrencia. Destacar que esta tabla hereda de *Social_node* por lo que dispondrá de los atributos necesarios para su geolocalización.

- **Tropa:** Guarda información de las tropas establecidas en el servidor. Contiene una referencia al territorio al que pertenece, tipo y cantidad restante. Matizar que solo se permitirá la creación de una tropa por territorio.
- **Conquista:** Almacena las batallas que se producen en los territorios. Se tienen referencias a los equipos (atacante y defensor) y la zona implicada durante la conquista. El campo booleano victoria indica si el equipo atacante tuvo éxito en el asalto.
- **Duelo:** Similar a la tabla anterior, esta guarda los duelos que se realizan a lo largo de un partida. Contiene las referencias a los equipos atacante y defensor, y el atributo victoria que indicará el vencedor del duelo.
- **Mensaje:** Recopila todos los mensajes que se intercambian entre los usuarios (organizador y equipos). Guarda las referencias del emisor y destinatario al que va dirigido, también contiene el cuerpo y la fecha de creación del mensaje.

3.4.2. Integración del Servicio Web en LibreGeoSocial

Desde un principio, el desarrollo de este proyecto se encuentra centrado en la utilización de LibreGeoSocial, permitiendo de una manera natural, la reutilización de funcionalidades en nuestro sistema. La integración de nuestro servicio web en esta red social es sencilla, tan solo hay que integrar nuestro proyecto Django en la carpeta *apps* y configurar el fichero *settings.py*. De esta manera se reconocerá satisfactoriamente las aplicaciones y librerías alojadas en su estructura.

3.4.3. Patrón Modelo-Vista-Controlador

Como ya se adelantó en el apartado 1.6, Django es un *framework web* que hace uso del patrón arquitectónico MVC [4] pero de una forma especial. Una vez recordada esta matización, se pasa a explicar las capas que lo componen y que se observan en la figura 3.5.

El **Modelo** representa la estructura del modelo de datos y las reglas de negocio que gobiernan el acceso y modificación de estos datos. Esta capa se encuentra definida en el archivo *models.py* de cada proyecto Django.

La **Vista** constituye la lógica de presentación, es decir, actúa como interfaz para que el usuario pueda interactuar con el sistema y es la encargada de generar

la respuesta (HTML, JSON [26], etc.) para ser enviada al cliente. En Django esta parte queda recogida en las funciones del fichero *views.py* y la forma de presentar estos datos se delega a las plantillas, situadas en la carpeta *templates*.

El **Controlador** es el encargado de gestionar las peticiones por parte del cliente y dar las órdenes para realizar los cambios oportunos. Una vez determinada las tareas a realizar, invocará a los componentes de la aplicación (Modelo y Vista) para que realicen la acción demandada. En este proyecto el Controlador es toda la maquinaria que hace funcionar Django, desde el fichero *urls.py* hasta las librerías ubicadas en la carpeta *core*.

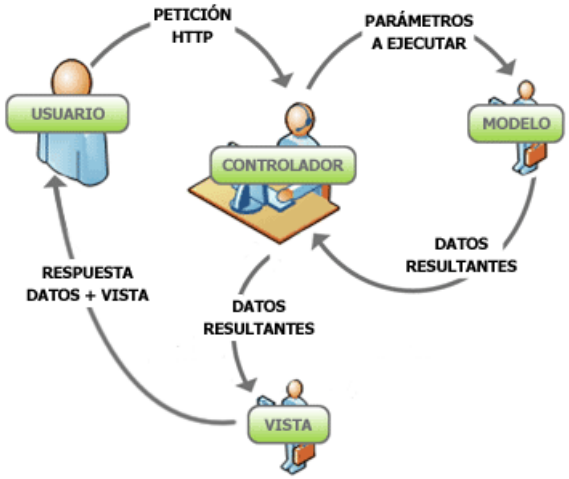


Figura 3.5: Diagrama del patrón MVC con las interacciones cliente-servidor.

En definitiva se ha conseguido integrar, de manera satisfactoria, el patrón MVC en el sistema, separando claramente la lógica de negocio y de presentación. Esto nos aporta una serie de ventajas:

- Desarrollo más rápido de los componentes del patrón debido a que se pueden implementar en paralelo.
- Las labores de mantenimiento se simplifican, ya que las correcciones se realizan en único lugar.
- Agregación o modificación de nuevas vistas sin paralizar el sistema.
- Aplicaciones más eficientes y por consiguiente, más escalables.

Pero también tiene una serie de desventajas seguir este planteamiento, como la mayor dificultad en el diseño e implementación del sistema, o el aumento en el número de ficheros dificultando su gestión.

3.4.4. Diseño de las URLs siguiendo la arquitectura REST

REST [27], acrónimo de *REpresentational State Transfer*, es un estilo arquitectónico para sistemas distribuidos hipermedia como la *World Wide Web*. Este estilo pretende proporcionar una colección de principios para el diseño de sistemas en red y seguir las principales características por las que la Web ha tenido tanto éxito.

El principal elemento de esta arquitectura son los recursos y son cualquier elemento que contenga información como por ejemplo: imágenes, documentos, vídeos, etc. Mediante la URI (concepto muy similar al de URL pero más técnico) se identifica inequívocamente un recurso. De esta manera se han diseñado un conjunto de URIs, especificadas en el fichero *urls.py*, para que el controlador sea capaz de asociar cada petición del cliente al recurso correspondiente. A continuación se expone algunos ejemplos:

```
/conquistadores/partida/crear/
/conquistadores/partida/listar/
/conquistadores/partida/1/modificar/
/conquistadores/partida/1/equipo/1/eliminar/
/conquistadores/partida/1/territorio/5/info/
/conquistadores/partida/2/equipo/1/territorio/5/conquistar/
```

Como se observa, cada recurso tiene asociado un identificador numérico que lo diferencia entre sus iguales, y al final de la URI la acción a realizar sobre este. Siguiendo la filosofía REST, el conjunto de URLs diseñado no da ningún tipo de detalle sobre la implementación del servicio web.

Un recurso puede tener múltiples representaciones y son elegidas por los clientes cuando realizan la petición HTTP, cada cambio de representación es un cambio de estado en el cliente. En este proyecto se distingue dos formatos para representar los recursos: HMTL para los clientes web, y JSON para los clientes Android.

Todas las comunicaciones entre los conectores (cliente, servidor, cache, etc.) tienen que ser sin estado. La información necesaria para entender una petición del cliente debe ser autocontenida, de tal manera que no dependa de comunicaciones precedentes. Por este motivo, un buen diseño del conjunto de direcciones es esencial para respetar esta norma.

Finalmente se hace uso de los principales métodos de petición que proporciona el protocolo HTTP para definir las operaciones sobre los recursos: *GET* (recupera la representación del recurso), *POST* (crea un nuevo recurso), *PUT*

(actualiza o crea un recurso) y *DELETE* (elimina el recurso especificado en la URI). Actualmente la mayoría de los navegadores web sólo soportan los dos primeros, por lo que es muy complicado seguir esta norma. En este proyecto, el método *GET* no cambia su cometido y el método *POST* se ha utilizado para la creación, modificación y eliminación de cualquier recurso.

Debido al diseño REST llevado a cabo, se adquieren las siguientes ventajas:

- Claridad en la construcción de las URLs haciendo que sean fácilmente entendibles para los usuarios.
- Debido a la simplicidad del diseño, se obtiene un mayor rendimiento y eficiencia del sistema.
- Fácil de implementar ya que no necesita herramientas auxiliares.
- Baja dependencia entre cliente y servidor.

3.5. Diseño de la aplicación Android

En primer lugar se detallará el modelo de clases de la aplicación. Posteriormente se explicará la utilización de LibreGeoSocialApp en el cliente. Por último, se analizará los mecanismos que ofrece el *framework* Android para la creación de las interfaces gráficas.

3.5.1. Modelo de clases

Mediante el modelo de clases, que se observa en la figura 3.6, se visualiza las diversas relaciones entre las clases que componen el cliente. Al igual que el modelo de datos del servidor, se utilizará el lenguaje UML para la especificación del diagrama. Debido al gran número de clases, se agruparán y comentarán las más importantes. Para una mayor información, se recomienda visualizar la cabecera de cada clase contenidas en el soporte informático de este proyecto.

- **Partida, Equipo, Marcador, Territorio, Mensaje:** Clases que representan el modelo de datos del servidor en la aplicación.
- **Conquistadores:** Clase encargada de visualizar la pantalla de inicio del juego.

- **PartidaListar, RecursoListar:** Muestran el listado de las partidas y equipos respectivamente. La clase *RecursoListar* también obtiene los elementos asociados a la partida seleccionada.

- **PartidaEmpezar:** Clase que muestra la pantalla con los nombres de la partida y equipo elegidos. Una vez que el usuario compruebe los datos, empezará el juego.

- **Estadisticas:** Clase que obtiene información del evento y equipos participantes. Estos datos junto al tiempo restante del juego, son mostrados al usuario.

- **Mapa:** Muestra un Google Maps con la ubicación de los participantes y territorios de la partida.

- **MiEquipoItemizedOverlay, EquipoItemizedOverlay, TerritorioItemizedOverlay:** Clases encargadas de personalizar y representar los diversos elementos geolocalizados en el mapa. También controla los eventos producidos cuando el usuario interacciona con los iconos.

- **OpcionesMenuEquipo, OpcionesMenuTerritorio:** Implementan la lógica de las acciones que el usuario puede realizar sobres los equipos y zonas respectivamente.

- **TerritorioRivalConquistar:** Clase que muestra la pantalla donde se selecciona la cantidad de tropas para la conquista de una región rival.

- **TropaMover:** Presenta la pantalla en la que el jugador determina la cantidad de unidades que desea desplazar entre sus recursos.

- **Mensajería, MensajeNuevo, MensajeListar, MensajeDetalle:** Implementan el sistema de mensajería dentro de la aplicación. Muestran el menú, el formulario para la creación de un nuevo mensaje, el listado e información detallada de los mensajes recibidos y enviados.

- **ServicioEnfrentamiento, ServicioLocalizacion, ServicioMapa, ServicioMensajeria, ServicioRecursoEquipo:** Clases encargadas de implementar los servicios, ejecutados en segundo plano, para la actualización de los diferentes elementos del mapa y mantener el correcto funcionamiento de la lógica del juego.

- **Utilidades:** Dedicada para establecer conexión con el servidor, usando las librerías de LibreGeoSocialApp. También proporciona los métodos para la finalización de los servicios y actividades una vez concluido el evento.

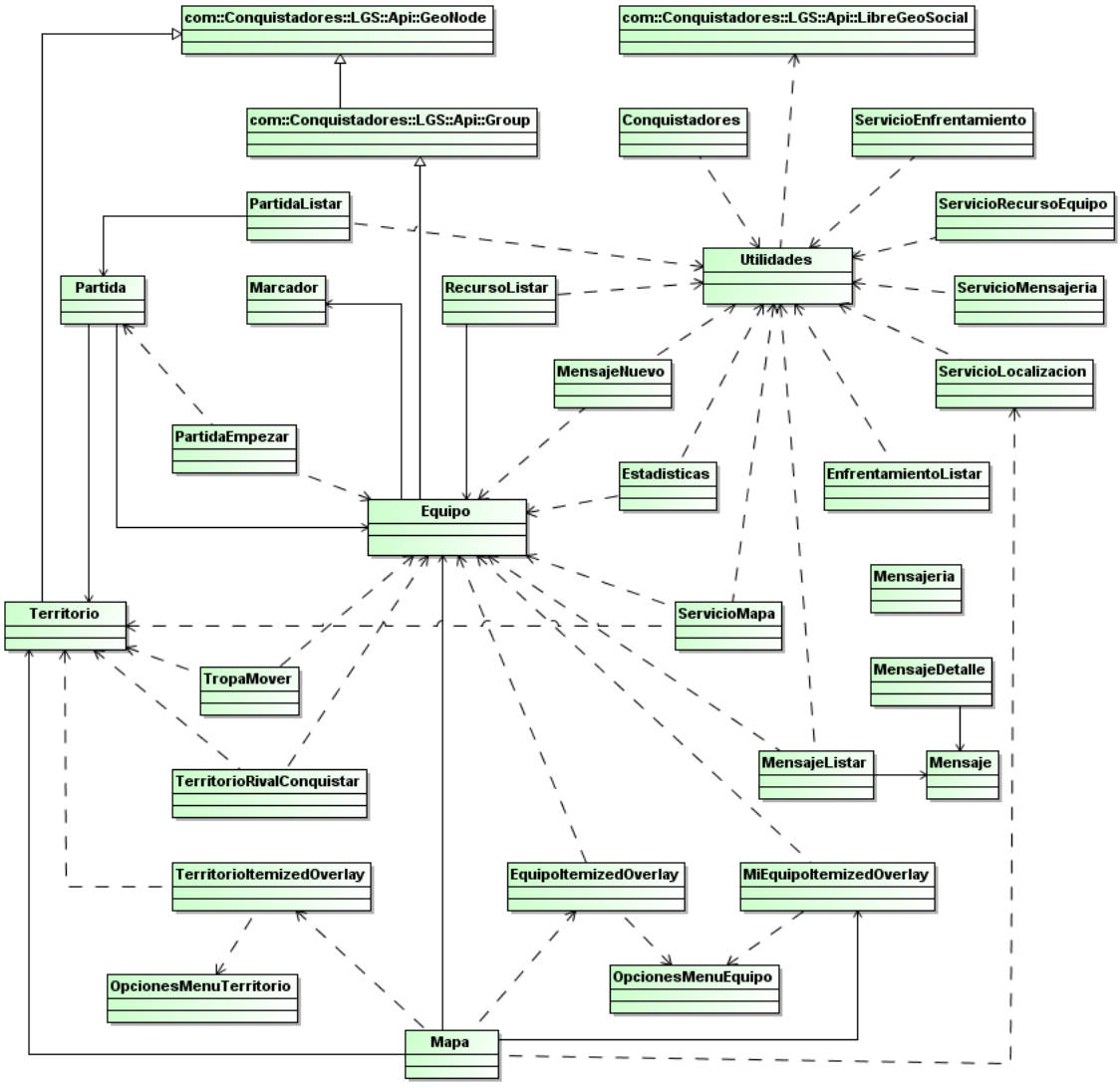


Figura 3.6: Diagrama del diseño de clases de la aplicación Android.

3.5.2. Utilización de LibreGeoSocialApp

Para un desarrollo más fácil y no tener que realizar una implementación de la aplicación desde cero, con el mayor grado de complejidad que esto conlleva, se han utilizado algunas librerías de la aplicación integrada en LibreGeoSocial, como se observa en la figura 3.7. Estas librerías han sido fundamentales en la

realización de las múltiples conexiones HTTP con el servidor y para la creación, y posterior almacenamiento, de los equipos con sus coordenadas geográficas (latitud y longitud).

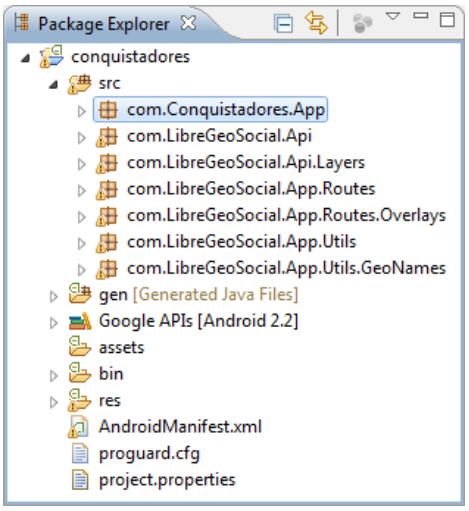


Figura 3.7: Manejo de LibreGeoSocialApp para el desarrollo de la aplicación.

3.5.3. Interfaz de usuario con XML

La interfaz en la aplicación define la estructura y los elementos que aparecen en la pantalla para el usuario. Android proporciona al programador diferentes formas para su diseño. La primera manera es la creación de las piezas de diseño en tiempo de ejecución, es decir, crear las vistas y manipular las propiedades en las clases Java. El principal inconveniente de esta técnica es el gran acoplamiento que se produce, haciendo que una pequeña modificación en la interfaz conlleve invertir muchas horas en el diseño y cambios en el código.

Otra alternativa para la construcción de estos componentes gráficos es mediante el uso de ficheros XML. Este mecanismo nos permite separar la presentación del código que controla el comportamiento de la aplicación. Además, el *plugin* ADT para Eclipse [28], proporciona una herramienta gráfica para la modificación o vista previa del XML. Debido a la serie de ventajas que ofrece esta opción, es la elegida para desarrollar las múltiples interfaces.

3.6. Implementación del servidor

El desarrollo del servidor es una parte fundamental, entre otras cosas, para el buen funcionamiento y control de la parte cliente. En la implementación de esta

parte se ha necesitado el uso e integración de diferentes herramientas: la utilización del *framework web* Django, como base de datos PostgreSQL y el servidor web HTTP Apache. Las características de estas aplicaciones se encuentran detalladas en el capítulo 1.

El principal motivo de la elección de Django es que también es utilizado en LibreGeoSocial. Esto nos permite una completa integración de nuestro proyecto y una implementación menos costosa, ya que se puede reutilizar librerías. Además de la multitud de ventajas, explicadas en el apartado 1.6.

Sobre la elección del gestor de la BBDD se empleó PostgreSQL porque también es utilizada en la red social móvil, pero también hay otros motivos por los que se ha decidido utilizarlo. En primer lugar, Django recomienda usar este gestor si se va a utilizar bases de datos espaciales, es decir, aquellas que pueden almacenar elementos con características geográficas. Para poder habilitar esta función de la base de datos es necesario la utilización del módulo PostGIS [29]. Otra razón de su elección, es que resulta extremadamente fácil integrar Python y PostgreSQL mediante el adaptador Psycopg [30]. De esta manera, la BBDD puede interpretar las órdenes dadas en este lenguaje de programación y transformarlas en lenguaje SQL, sin que el desarrollador tenga que intervenir en este proceso.

Durante el progreso de este proyecto se ha utilizado el servidor de desarrollo que incorpora Django. Gracias a este tipo de servidor, la implementación del sitio web y la aplicación Android se han desarrollado de manera más ágil. Como desventaja, es que no se recomienda su uso fuera del entorno de desarrollo y como el sistema va a ser probado en un entorno real, puede ocurrir que este tipo de servidor no funcione correctamente debido a un mayor grado de exigencia. Por este motivo y por mi experiencia personal, se decidió usar Apache como servidor de producción web. Esta decisión implica también la utilización del módulo mod_wsgi [31] para hospedar aplicaciones programas en Python.

Un aspecto fundamental en la implementación del servidor es el mecanismo para controlar posibles casos de concurrencia a lo largo de una partida. Durante el análisis de la aplicación, se detectó que varios usuarios podían acceder de manera simultánea a un mismo recurso alojado en la base de datos. Por lo que es obligatorio identificar y tratar estas condiciones de carrera en el lado del servidor, para que la experiencia de los jugadores sea totalmente satisfactoria. Django, por si mismo, no ofrece ningún mecanismo para gestionar la concurrencia, por lo que se tuvo que sopesar entre diferentes modelos y elegir uno para controlar esta misma.

A continuación se explican los diferentes mecanismos que se han considerado para tratar este problema:

- **Control de concurrencia pesimista (PCC)**: Este tipo de modelo implica aislar en la BBDD alguna parte de la tabla que está siendo usada. Esto conlleva que otro usuario que necesite leer o modificar algo relacionado con el bloqueo, no podrá hasta que el recurso quede liberado. Este tipo de modelo es útil en entornos donde los periodos de bloqueos sean cortos y/o el coste computacional de proteger los datos mediante cerrojos, sea menor que deshacer una transacción. El problema de este mecanismo es que requiere grandes cantidades de recursos por parte del servidor y deja de ser escalable cuando los usuarios desean interactuar con un mismo elemento. Django permite usar este modelo mediante la aplicación django-locking [32].

- **Control de concurrencia optimista (OCC)**: Mediante esta técnica se permite realizar multitud de transacciones sin que sea necesario bloquear algún tipo de fila o tabla. Cuando se termina la transacción, se evalúa si ha habido algún conflicto durante el proceso. En caso afirmativo, se desecha la transacción. Este tipo de concurrencia es útil en situaciones donde la probabilidad que se de algún conflicto sea baja, ya que si no, se realizaría trabajo para posteriormente descartarlo. También el uso de esta técnica aumenta el rendimiento del sistema, debido a que los datos siempre están disponibles para los clientes.

Considerando ambas técnicas, al final se decidió implementar el control de concurrencia optimista. El principal cambio que han sufrido las tablas susceptibles a algún tipo de condición de carrera, ha sido la inserción de una nueva columna que determina la versión de los datos de esa fila. Básicamente cuando un cliente solicita algún dato de estas tablas, también obtiene la versión. En el momento de modificar alguna fila se compara que la versión almacenada en la base de datos es la misma que se solicitó en la anterior transacción. Si las versiones coinciden, se realizan las modificaciones y se incrementa el valor de la columna "versión". En caso contrario, se descarta la nueva petición y se le informa de lo sucedido.

Para un correcto funcionamiento de esta técnica, se han utilizado operaciones atómicas (*F()* y *update()*) incluidas en la API de Django. Este tipo de operaciones realizan completamente su acción (lectura, modificación, etc.) en la base de datos sin que nadie pueda interrumpirla. Esto es esencial cuando se trabaja con concurrencia.

Se puede identificar de una manera muy clara que tipología de clientes harán uso del servidor, cada uno de una manera diferente. El primer tipo son los organizadores, que mediante la implementación de una interfaz web y el uso de cualquier navegador web, podrán gestionar los diferentes elementos que conforman una partida de *Los Conquistadores* y efectuar su monitorización. Por otro lado se encuentra la implementación de una aplicación Android, explicada en el apartado 3.7, destinada a los usuarios que posean un terminal móvil con dicho sistema operativo con el que podrán participar.

La información recibida por parte del servidor estará codificada en diferentes formatos dependiendo del tipo cliente que interactúe con el servidor. Si se trata de un navegador web, la información será enviada y recibida mediante el formato HMTL, ya que estos datos se visualizarán en páginas web. En cambio, si establece comunicación la aplicación, se usará el formato JSON debido al fácil parseo de esta tecnología en el lenguaje de programación Java y porque es utilizado en LibreGeoSocial.

3.6.1. Creación de la interfaz web

Antes de explicar las distintas acciones que el usuario puede realizar mediante la interfaz web, quiero destacar que se ha utilizado una hoja de estilo en cascada (CSS) [33] recomendado por el *World Wide Web Consortium* (W3C), para dar formato a los documentos HMTL. De esta forma se consigue separar el contenido de la presentación pudiendo cambiar el aspecto visual de la página, realizando cambios en la plantilla sin la necesidad de modificar el código.

Seguidamente se detallan las numerosas acciones que un organizador puede realizar en este sitio web, para la gestión de las partidas y los diversos recursos que forman parte de esta.

Sistema de autenticación

La primera pantalla que el usuario visualizará cuando intente acceder a cualquier dirección del sitio web, siempre y cuando no se encuentre autorizado, será el acceso al sistema de autentificación (ver figura 3.8) solicitada al servidor mediante una petición *GET*. Este como respuesta devuelve el fichero *login.html*, el cuál será mostrado al cliente. Cuando se introduzca el nombre de usuario y su contraseña en el formulario, se enviarán estos datos mediante una petición *POST*. El servidor verificará si existe algún recurso (organizador o jugador) que coincida

con los mismos datos recibidos. En caso de que se produzca algún tipo de fallo en el momento de la validación, el servidor enviará la respuesta y le mostrará, en la misma página, que el nombre de usuario o contraseña son incorrectos. De esta manera se intenta aumentar la seguridad del sitio web debido que no se especifica qué tipo de parámetro ha fallado.

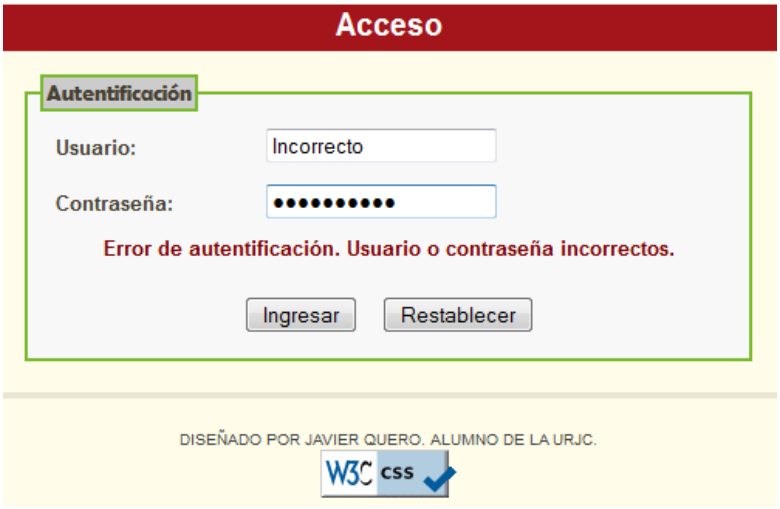


Figura 3.8: Formulario de autentificación informando de un acceso erróneo.

En caso de que el proceso haya tenido éxito y dependiendo del tipo de usuario que ingrese en el sitio web, podrá realizar determinadas acciones. Un simple jugador solo obtendrá información de los diferentes elementos que conforman una partida (equipo, territorio, etc.) pero no podrá crear, modificar y/o eliminar recursos en la base de datos. En cambio, un usuario autorizado como organizador tendrá pleno control sobre las partidas. Debido a que el sitio web se desarrolló con el objetivo de crear y gestionar las partidas, nos centraremos en las acciones que pueden realizar este último tipo de cliente.

Una vez certificado, se le redirecciona a la pantalla de bienvenida (ver figura 3.9), donde el organizador podrá navegar, mediante un menú horizontal desplegable situado en la parte superior, por todas las páginas del sitio web. Este menú siempre estará disponible y variará sus opciones según el recurso en el que se encuentre. Seguidamente se visualizará una breve descripción de todos los elementos que constituyen el juego, y por último un listado con el nombre de todas las partidas que ha creado junto con sus acciones para una fácil y rápida gestión.

Figura 3.9: Pantalla de bienvenida mostrando información sobre el juego.

Gestión de las partidas

El organizador podrá crear, modificar o eliminar cualquier partida a su gusto. Lo más lógico es que en primer lugar se cree una partida para que albergue los diferentes elementos del juego, como por ejemplo, territorios, equipos, mensajes, etc. Para llevar a cabo esta acción se debe seleccionar la opción "Crear" del menú desplegable "Partida", realizándose una petición *GET* al servidor mediante la dirección '/conquistadores/partida/crear/'. Acto seguido se visualizará un formulario para su creación en el sistema. Los campos requeridos en este formulario son: título, lugar, duración, núm. de espías, cantidad de escolta, crecimiento de tropas y una descripción de la partida.

En el momento de enviar cualquier tipo de formulario, la librería LiveValidation [34], gracias al uso de la tecnología AJAX, comprueba en el lado del cliente si hay algún campo vacío o mal rellenado (una cadena de texto en una casilla que únicamente requiere valores numéricos, etc.). En la figura 3.10 se aprecia las alertas generadas por esta librería. Aunque en el lado del servidor se han implementado medidas para detectar y avisar de posibles errores de los datos recibidos, usar este mecanismo de seguridad conlleva una serie de ventajas. La primera es el ahorro de tráfico entre cliente-servidor, también se consigue una interfaz web más amigable debido a que el usuario es informado rápidamente de los errores y por último, se evitan errores en tiempo de ejecución.

Cabe mencionar que el envío de este formulario, se realizará mediante una

Figura 3.10: Formulario para la creación de una partida junto con las alertas generadas por LiveValidation.

petición *POST* sobre la misma dirección anteriormente mencionada. Como norma general, el servidor podrá realizar diferentes acciones sobre una misma URL, dependiendo del tipo de petición (*GET* o *POST*) recibida.

Una vez creada o seleccionada una partida del listado, se le redirigirá a una nueva página mediante la URL '/conquistadores/partida/<partida_id>/info/' (ver figura 3.12). En ella, el organizador visualizará la información más relevante de la partida en cuestión. También se le da la posibilidad, mediante el botón "Modificar", de realizar cualquier cambio en sus atributos, como por ejemplo el nombre, fecha, lugar de celebración, etc. Si finalmente decide realizar alguna modificación se le redireccionará a otra página, similar a la que se utilizó en el momento de crearla, con la salvedad de que el formulario se encontrará debidamente cumplimentado con los datos de la partida almacenados en la base de datos. Igualmente, está interfaz también utiliza la misma librería JavaScript para validar los campos antes de ser enviados.

El organizador podrá visualizar un listado, como se puede observar en la figura 3.11, de todas las partidas que se encuentren almacenadas, sin importar quien las creó. Para esto se debe seleccionar la opción "Listar" del menú "Partida". Se realizará una petición *GET* a la URL '/conquistadores/partida/listar/', mos-

trando de forma tabulada el listado. Cada fila de la tabla, recogerá el nombre, estado y las diferentes acciones que se pueden aplicar a cada una de estas.

Figura 3.11: Pantalla con el listado de las partidas.

El estado de un evento indica si está disponible para jugar, y esto es esencial para el cliente Android. Básicamente las partidas con el estado "cerrada" no estarán visibles, por lo contrario, las que si son seleccionables serán las que tengan el estado "abierta". El organizador puede alterar el estado con tan solo pulsar la correspondiente acción de la columna "Opciones". Esta acción, mediante una petición *GET*, solicita al servidor que modifique el campo correspondiente del recurso en cuestión.

La eliminación de cualquier partida por parte del organizador es perfectamente posible. Para llevar a cabo esta acción, basta con decidir la fila y pulsar la opción "Eliminar". El resultado final será la eliminación de la partida y de todos los elementos que dependan de esta, es decir, equipos, territorios, mensajes, etc. De esta forma se consigue mantener la estructura de los datos almacenados limpia y ordenada, evitando que queden recursos sueltos de una partida borrada. Una vez que el servidor ha tratado la petición correctamente, se volverá a la misma página pero con el listado actualizado.

Como ya se ha mencionado anteriormente, también será posible visualizar los datos de un evento. Para ello es necesario pulsar sobre el enlace "Info de la partida" y acto seguido se redireccionará a una nueva página (figura 3.12) donde se mostrará dicha información. Siempre que el organizador se encuentre en esta nueva página se desbloquearán, en el menú desplegable, las nuevas opciones para poder administrar los recursos asociados a esta partida. La gestión de estos elementos se explica en subapartados posteriores.

Figura 3.12: Pantalla con la información de la partida seleccionada.

Finalmente cabe destacar que en el caso de que el servidor detectase un error al realizar cualquier acción sobre este tipo de recurso u otros, el usuario será redirigido a una página creada expresamente para este motivo. En ella se le detallará la naturaleza del error y si es posible, actuar para solventar el fallo.

Gestión de los equipos

Dentro de la información de una determinada partida, el organizador puede gestionar el listado y datos de los equipos que van a participar en ella. Lo primero en lo que nos vamos a centrar es en la creación de un equipo, dado que inicialmente una nueva partida no contiene ningún elemento. Para esto, basta con pulsar la opción "Crear" del menú "Equipo", enviando una petición *GET* a la URL '/conquistadores/partida/<partida_id>/equipo/crear/'. Como respuesta, se mostrará una nueva página indicando al usuario el nombre de la partida junto al correspondiente formulario para la creación del nuevo grupo. Únicamen-

te se requiere un nombre y seleccionar el color que lo identificará tanto en el seguimiento de la web como en la aplicación.

Un equipo tiene que ser un recurso geolocalizado y para que cumpla esta condición se ha tenido que relacionar cada elemento con un *Group* de LibreGeoSocial. Esta asociación implica que no pueden existir dos grupos con el mismo nombre, por lo que si durante el proceso de creación el servidor detecta que el nombre ya existe, informará al organizador del problema mediante una nueva pantalla (ver figura 3.13). También se ha limitado su cantidad, debido a que no hay más de seis colores para su identificación. En el caso de que se intente crear un equipo cuando no queden colores disponibles o no se haya seleccionado alguno, gracias a la librería LiveValidation, se le advertirá en la misma página que esa acción no se puede producir.

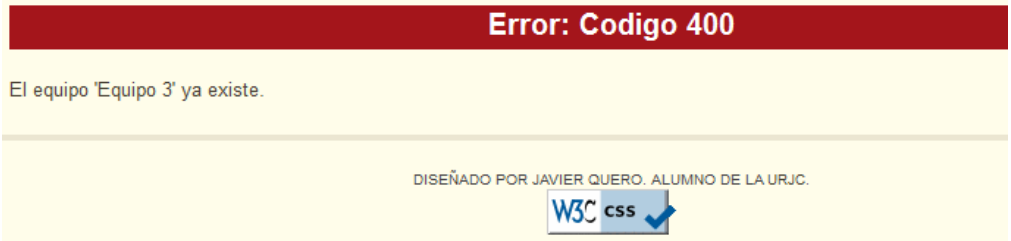


Figura 3.13: Notificación de error al crear un equipo cuyo nombre ya existe.

Una vez realizado satisfactoriamente el proceso de creación, se visualizará la página que contiene la información del equipo, como se observa en la figura 3.14, a través de la URL '/conquistadores/partida/<partida_id>/equipo/<equipo_id>/info/'. En esta página aparece el nombre, color, números de espías y escolta. También su marcador asociado con la puntuación y la cantidad de territorios que posee en ese instante. Por último, se dará la posibilidad de cambiar su nombre, mediante el botón "Modificar". Esta opción es similar a la modificación de una partida u otro elemento, por lo que no es necesario explicar nada nuevo respecto a esta acción.

Para poder ver todos los equipos que están inscritos a una partida seleccionada, tan solo hay que elegir la opción "Listar" del menú "Equipo". Similar al listado de partidas, se presentará una tabla con todos los grupos junto con sus respectivas acciones. Si se decide eliminar alguno, se procederá a su borrado en la base de datos junto a su *Group*, además su color volverá a estar disponible. En el caso de comprobar la información, se mostrará la misma página resultante que en la creación exitosa de un equipo.

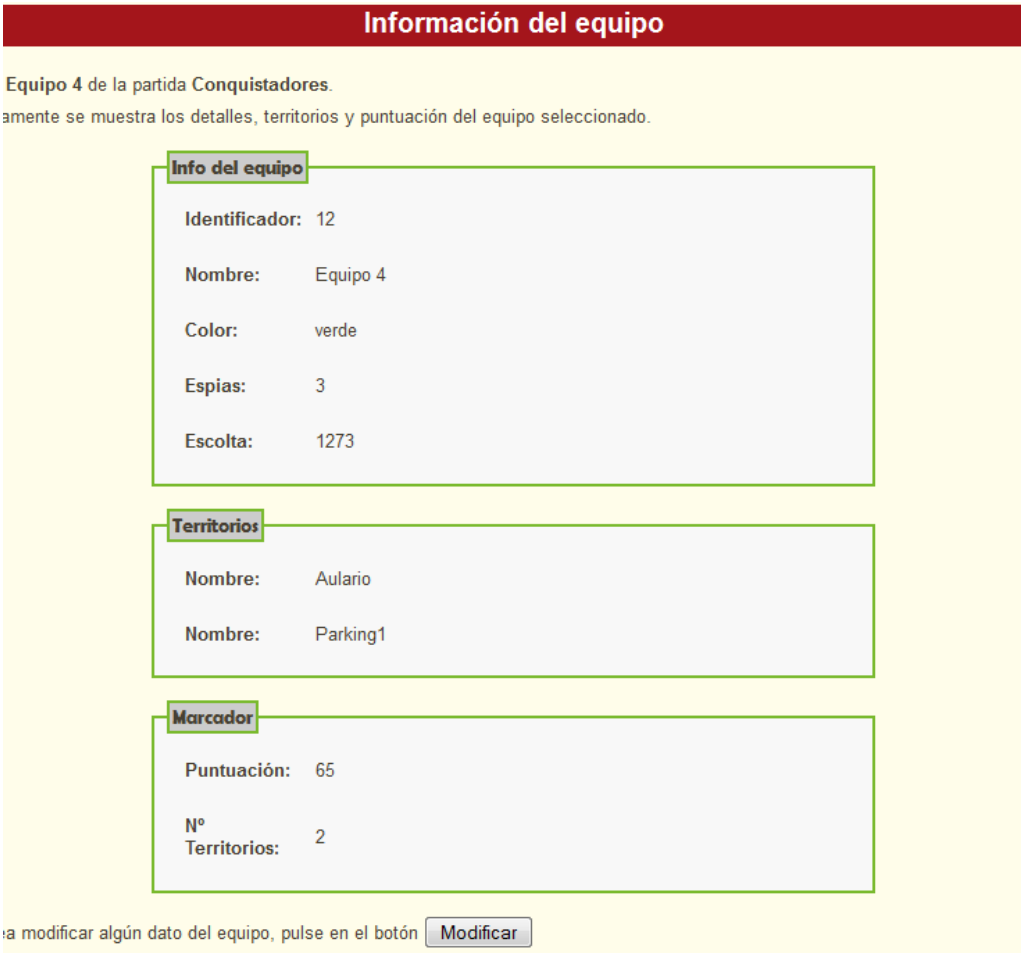


Figura 3.14: Pantalla con la información del equipo seleccionado.

Gestión de los territorios

Una vez establecida la partida y los equipos que van a participar, es necesario añadir el elemento principal sobre el que se basará el juego. Para la creación de los territorios, el organizador tendrá que pulsar sobre la opción "Crear" del menú "Territorio". Al igual que los casos anteriores, se abrirá una nueva página web con el siguiente contenido:

- Un único formulario para la creación del territorio y la tropa que tendrá asociada, cuyo objetivo será defender la zona.
- Un Google Maps para obtener, de una forma sencilla y cómoda, las coordenadas latitud y longitud necesarias para establecer su ubicación.

De una manera similar al anterior elemento, un territorio también será un componente geolocalizado, pero en este caso su ubicación geográfica será fija y

no cambiará durante el transcurso del juego. En el momento de su creación, como se observa en la figura 3.15, se requerirá un nombre y las coordenadas geográficas. Los dos campos del formulario encargados de recoger su posición, estarán bloqueados sin la posibilidad de ser rellenados por el usuario. La manera de cumplimentarlos será mediante la interacción de un Google Maps [35], que se explicará posteriormente. También será obligatorio establecer la tropa encargada de proteger la zona, para este objetivo se encuentran los campos tipo y cantidad que permiten personalizar el nombre y cantidad inicial de este recurso en cada territorio.

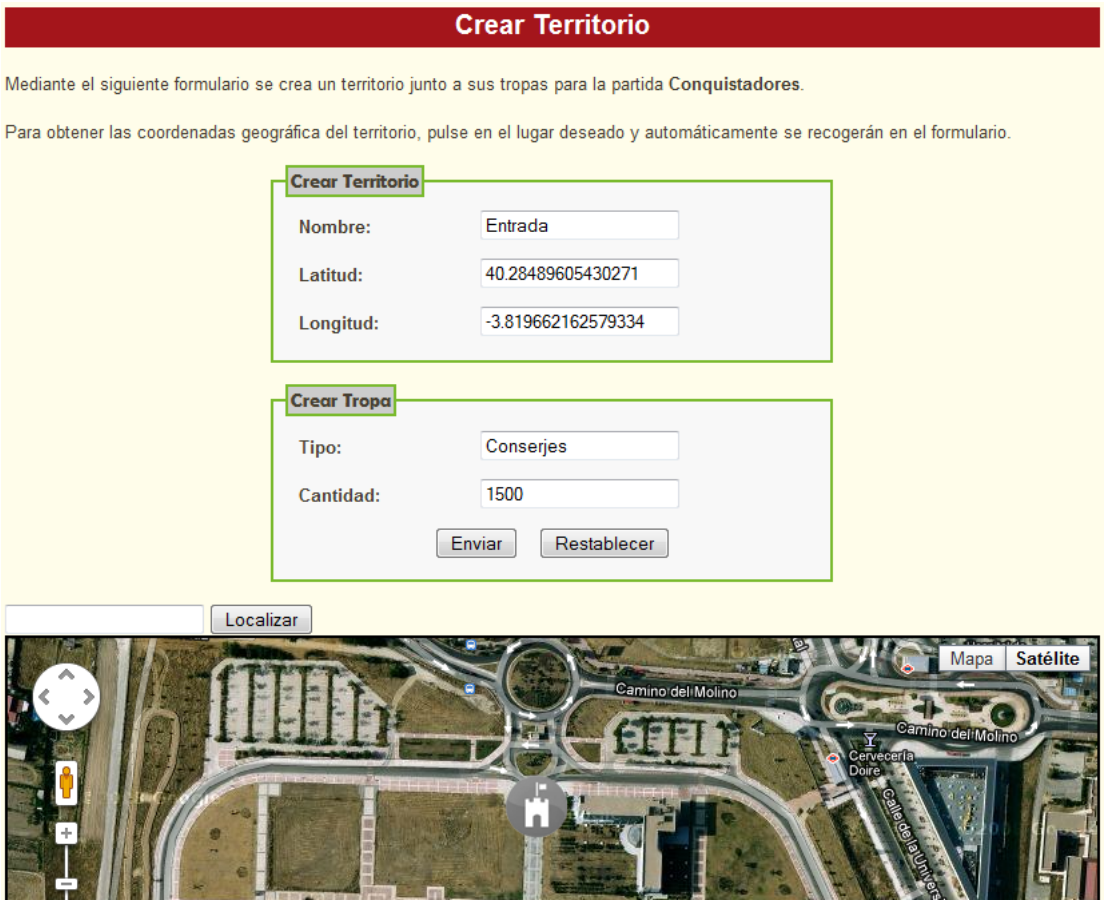


Figura 3.15: Pantalla para la creación de un territorio y su tropa, junto al Google Maps para la obtención de las coordenadas.

Con la inserción del Google Maps se ha buscado facilitar la ubicación del territorio, sin la necesidad de recurrir a herramientas externas del sitio web. Mediante una serie de controles incrustados en la interfaz del mapa que posibilitan el cambio de zoom, desplazamiento o la elección del tipo de vista, el usuario podrá seleccionar de manera rápida, sencilla y precisa la ubicación del nuevo te-

rritorio. El emplazamiento elegido quedará representado mediante un icono, con tan solo apretar el botón izquierdo del ratón sobre la zona del mapa, rellenándose de manera automática los campos latitud y longitud del formulario. Este proceso se puede repetir todas la veces que se desee hasta encontrar la situación idónea del territorio en la partida.

Al ser un elemento geolocalizado, también ha sido necesario usar la librería de LibreGeoSocial para su correcta creación en la base de datos. Todos los territorios inicialmente son neutrales, es decir, no tienen ningún propietario y sus tropas no serán resistencia en su primera conquista. En caso de producirse algún error durante el proceso, el organizador será informado mediante la correspondiente página de error.

Si todo va bien, se redireccionará a una nueva página (véase la figura 3.16) donde se expondrá la información del territorio y tropa recién creados. Como en los casos anteriores, en esta misma página se da la posibilidad de modificar algún

Figura 3.16: Pantalla con la información del territorio y tropa asociada.

dato de estos dos recursos. En tal caso, se le mostrará una página similar a la creación del territorio donde se visualizará el formulario totalmente cumplimentado y el Google Maps con el icono representando su ubicación actual.

Por último, mediante la opción "Listar" del menú "Territorio", se mostrará el registro completo de todos los territorios que contiene la partida. A través de las opciones que nos proporciona este listado, se puede obtener información o proceder a la eliminación de algunos de sus elementos. El borrado de un territorio también lleva implícito la eliminación de su tropa.

Mensajería

Durante el desarrollo del sitio web y aplicación Android, se detectó la necesidad de que el organizador y los equipos participantes puedan comunicarse entre ellos, sin la exigencia de utilizar otros programas externos. Por este motivo se ha decidido implementar un sistema de mensajería interno, dando servicio al sitio web y aplicación, con el cual se pueda intercambiarse mensajes de una manera sencilla, rápida y sin costes adicionales.

El funcionamiento de este sistema, a grandes rasgos, se basa en el almacenamiento de los mensajes en la base de datos. Cuando el sitio web o aplicación los solicita, se envían mediante su correspondiente formato para ser visualizados. En este apartado nos centraremos en explicar el funcionamiento de este servicio mediante la interfaz web. La implementación en los clientes Android se explicará en el subapartado 3.7.4.

Para crear un nuevo mensaje a través del sitio web, el organizador tiene que seleccionar la opción "Crear" del menú "Mensaje". Acto seguido se visualizará una nueva página para su redacción y posterior envío, como se puede observar en la figura 3.17. Además del campo para redactar el texto, el organizador puede seleccionar si el mensaje va destinado a un equipo concreto o por lo contrario, va dirigido a todos los participantes de la partida. Este tipo de elección resulta muy útil dependiendo de las circunstancias que van surgiendo en el transcurso del evento. En el momento de enviar el mensaje, se comprobará si se ha seleccionado el destinatario o el campo de redacción se encuentra vacío. Si se cumple alguno de estos casos, el JavaScript de validación informará del error para su posterior corrección.

Figura 3.17: Pantalla para el envío de mensajes a los equipos.

Realizado satisfactoriamente todo el proceso, se volverá a la página de seguimiento, donde aparecerá el listado de todos los mensajes que se han enviado durante la partida.

Seguimiento de la partida

Establecidos los equipos, territorio y tropas que conforman la partida, el sitio web da la posibilidad, a cualquier usuario del sistema, de observar y controlar la partida en tiempo real. Para esto, se dispone de una nueva página accesible mediante la opción "Seguimiento" del menú.

Como se observa en la figura 3.18, la primera sección consta de un Google Maps en el que se representarán la ubicación de los diferentes elementos geolocalizado de la partida. Gracias a que se utilizan iconos de diversos colores para simbolizar los territorios y equipos, el usuario puede asociarlos e identificar rápidamente el dueño de cada zona. Cuando se pulsa sobre cualquier marcador del mapa, aparecerá un cuadro de texto con la información más relevante. En el caso de los territorios, se visualizará su nombre, el equipo propietario (si lo tuviese), el tipo y cantidad de tropa que defienden el lugar. Si se selecciona un equipo se obtendrá su nombre, cantidad de escolta y espías que posee actualmente. Por cada refresco de la página, se solicita al servidor todos los datos actualizados de los marcadores para que el usuario pueda seguir la partida en vivo.

Figura 3.18: Pantalla con el mapa para el seguimiento de la partida.

En el siguiente bloque mediante el uso de una tabla, el usuario podrá visionar la clasificación ordenada de la partida (ver figura 3.19). Esta información está compuesta por el nombre, número de territorios y puntuación de cada equipo. El grupo vencedor será aquel que ocupe la primera posición de esta tabla al final de la partida.

Clasificación

equipos con su puntuación.

Id. equipo	Equipo	Nº Territorios	Puntuación
12	Equipo 4	2	65
10	Equipo 2	2	50
11	Equipo 3	1	30
9	Equipo 1	1	15

Figura 3.19: Pantalla con la clasificación de la partida.

La tercera y última sección de esta página, que se ve en la figura 3.20, es la relacionada con la gestión y visualización de todos los mensajes intercambiados durante el juego. De cada mensaje se tiene su emisor, destinatario, fecha de

creación y cuerpo. Para una mejor disposición y claridad en su lectura, se han distribuido en dos columnas. La columna de la izquierda albergará los mensajes enviados por el organizador y la columna de la derecha aquellos enviados por los equipos.

Figura 3.20: Pantalla con el listado de mensajes intercambiados a lo largo del juego.

El listado de los mensajes estará ordenado cronológicamente, es decir, los últimos se posicionarán en la parte superior, desplazando los más antiguos hacia abajo. De esta manera se evita la perdida de tiempo localizando los últimos mensajes, consiguiendo una comunicación más fluida y rápida con el jugador. Por último, los organizadores serán los únicos que podrán borrarlos del sistema mediante la opción "Eliminar mensaje".

Como se ha comentado un poco más arriba, para poder visualizar los datos actualizados y realizar un mejor seguimiento de la partida, es necesario refrescar periódicamente la página. Por este motivo se ha implementado un sencillo *script* que realiza dicha función sin que el usuario tenga que intervenir, ahorrándole las molestias que pueda ocasionarle la realización de esta tarea.

3.7. Implementación de la aplicación Android

En este apartado se explicará el desarrollo de la aplicación *Los Conquistadores*, una aplicación desarrollada en Android. Mediante un mapa (ofrecido por la API de Android) y usando la localización GPS, los equipos participantes tendrán que ir conquistando los territorios que posean los rivales, además de otras funcionalidades como duelos entre equipos, espionaje de territorios, etc. El objetivo final es sumar la mayor cantidad de puntos para alzarse con la victoria.

Una vez que el usuario ejecute el programa, la primera *Activity* y por consiguiente la primera pantalla que se encontrará será la de bienvenida a la aplicación (ver figura 3.21).

Una vez que pulse el botón, se comprobará si se encuentran activos el servicio de GPS y la conexión de datos, ambos imprescindibles para el correcto funcionamiento de la aplicación. En el caso de que alguno no se encontrase en marcha, se avisará al usuario mediante un dialogo alertándolo de la situación. Este tipo de diálogo y otros similares que el usuario verá durante el transcurso del juego, se implementan usando la clase *AlertDialog* de la API de Android. Estos diálogos son pequeñas ventanas que aparecen delante de la *Activity* perdiendo su foco e interrumpiendo al usuario para informarle del hecho que ha ocurrido.

Si todo está en orden, acto seguido se solicitará al servidor la información de todas las partidas disponibles en la base de datos en ese momento, al usuario se

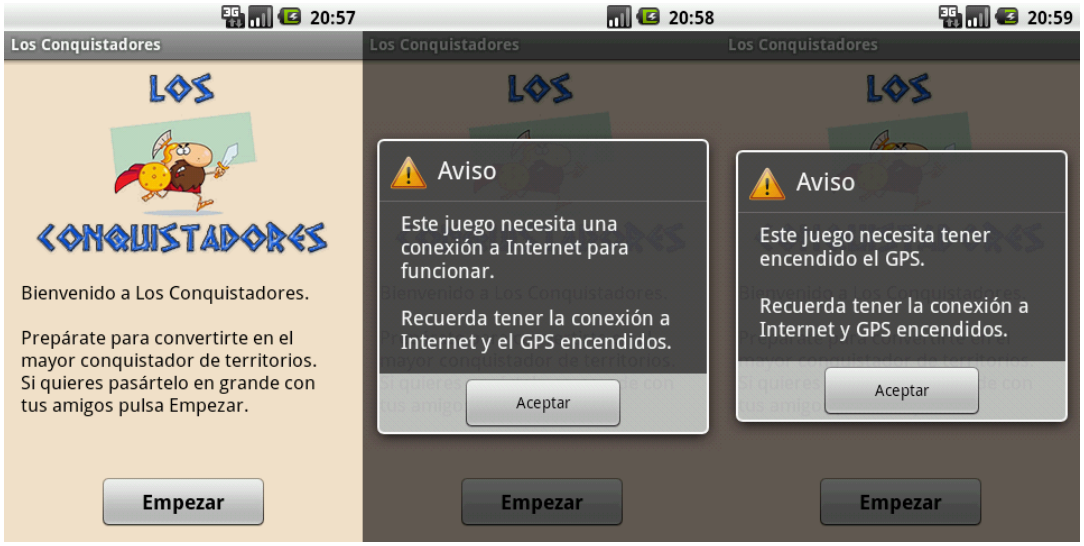


Figura 3.21: Pantalla de bienvenida y notificaciones sobre el estado de la conexión de datos y GPS.

le presentará una lista (ver figura 3.22) con los datos más importantes de cada partida: nombre, lugar, duración, etc. La GUI que muestra esta pantalla ha sido implementada mediante herencia de la clase *ListActivity* de Android (*Activity* que muestra una lista de elementos y permite manejar el evento cuando el usuario pulsa sobre un elemento de esta). Cuando el usuario pulse sobre la partida a la que desea unirse, se pasará a una nueva pantalla (el paso de una *Activity* a otra se realiza mediante la clase *Intent*) que listará los equipos disponibles para la partida elegida (ver figura 3.22). En este momento se aprovecha para solicitar más información a parte de los equipos, son los territorios que conforman la partida. Puede darse el caso de que falten algunos de los recursos mencionados arriba (partidas, equipos o territorios) que imposibiliten el inicio del juego, en esta situación se informará al usuario mediante un *AlertDialog* y se retornará a la pantalla de inicio.

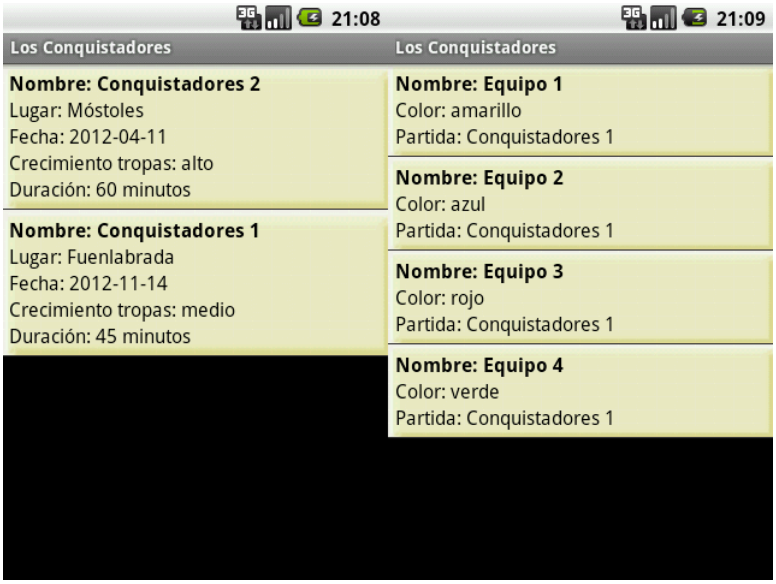


Figura 3.22: Pantallas con el listado de las partidas y equipos disponibles.

La obtención de los equipos es similar a la de las partidas, la aplicación obtiene la información del servidor en formato JSON. De aquí en adelante, cada vez que la aplicación obtenga cualquier respuesta por parte del servidor será en este formato. Cada clase implementa un método donde obtiene y trata la información recibida, de esta forma se crearán objetos (partida, equipo, territorio, mensaje, etc.) en Java que se asemejarán a la estructura de la base de datos del servidor (archivo *models.py*). Estos objetos se almacenarán en *ArrayList* (estructura de datos proporcionada por la API de Java) para posteriormente poder consultarlos y/o modificarlos según los acontecimientos que surjan a lo largo de la partida en

curso. De esta forma la aplicación puede acceder a estas estructuras sin realizar nuevas peticiones, ya que gran parte de esta información como el nombre de la partida y equipos, posicionamiento geográfico de los territorios, etc. nunca van a ser alterados. Con esto se logra reducir las peticiones del cliente, lo que se traduce en un ahorro en términos de consumo en la tarifa de datos de los terminales móviles.

Cuando el usuario haya seleccionado los elementos con los que desea competir, la siguiente pantalla será la última antes de comenzar el juego. En esta *Activity* (ver figura 3.23) se mostrará el nombre de la partida y equipo seleccionado, junto a la descripción que el organizador escribió en el momento de crear la partida. Con esta información se pretende minimizar las posibles situaciones en la que un usuario seleccione una partida o un equipo por error antes de comenzar. Si fuese este el caso, basta con pulsar el botón de retroceso con lo que se volverá a la pantalla inicial de la aplicación. Una vez que el jugador ha comprobado que su elección ha sido correcta, tan sólo tendrá que pulsar el botón "Empezar" para dar comienzo al evento. También y sin que el usuario tenga conocimiento, se iniciarán

Figura 3.23: Captura con la información de la partida seleccionada.

los principales servicios de la aplicación: de localización, de recursos del equipo, de mensajería y de enfrentamiento. La implementación y utilidad de cada servicio será explicado detalladamente más adelante.

En este momento ha empezado la partida y lo primero que verá el usuario será una pantalla con tres pestañas que estarán todo el tiempo visible en la parte

superior. La creación de estas pestañas se realizó mediante la instancia de la clase *TabHost* proporcionada por la API de Android, gracias a los métodos que aporta esta clase se puede crear una lista de pestañas, las cuales al pulsar realizan un *Intent* a otra *Activity*, es decir, el usuario visualizará una nueva pantalla. La utilización de pestañas permite una organización sencilla y clara de la GUI, de tal manera que al usuario le es más intuitivo moverse entre las diferentes pantallas que configuran la aplicación.

A continuación una pequeña descripción de cada pestaña donde en apartados posteriores se entrará en más detalle de la composición y acciones que se pueden realizar en cada una de ellas.

- La primera y que siempre se mostrará por defecto, son las estadísticas del equipo participante, como su nombre, color, puntuación, etc. explicada en el subapartado 3.7.1.

- La segunda pestaña nos permite visionar un mapa de Google y es aquí donde se desarrollará toda la acción del juego. En el subapartado 3.7.2 se explicará la creación y manejo del mapa, así como la obtención y el dibujado de los elementos. En el subapartado 3.7.3 se detallarán las principales acciones que el usuario puede realizar.

- Por último y no menos significativo, se encuentra la pantalla de mensajería explicada en el subapartado 3.7.4.

Con el fin de ayudar al equipo ante las posibles dudas que le surjan durante el desarrollo de la partida, totalmente comprensible si es la primera vez que entra en contacto con la aplicación, todas las pantallas que se mostrarán a partir de ahora tendrán un menú de ayuda (ver figura 3.24). Lo que se recoge en este tipo de menú son las principales acciones que se pueden realizar dependiendo en que pantalla se encuentre. Para poder visionar dicha ayuda tan solo hay que pulsar en el botón físico menú y seleccionar la opción "Ayuda".

Cada vez que una *Activity* realiza una petición HTTP ya sea para la obtención o envió de cualquier dato al servidor, se muestra en la pantalla una barra de progreso implementada mediante la clase *ProgressDialog*, como se puede observar en la figura 3.24. Se ha tomado esta medida para que el usuario no tenga la percepción errónea de que la aplicación no responde cuando se está llevando a cabo algún tipo de acción. Para poder realizar la implementación de los *ProgressDialog* es necesario crear un *thread* de Java que se encargue de realizar la petición

y obtener la respuesta, mientras el hilo que controla la *Activity* muestra la barra de progreso.

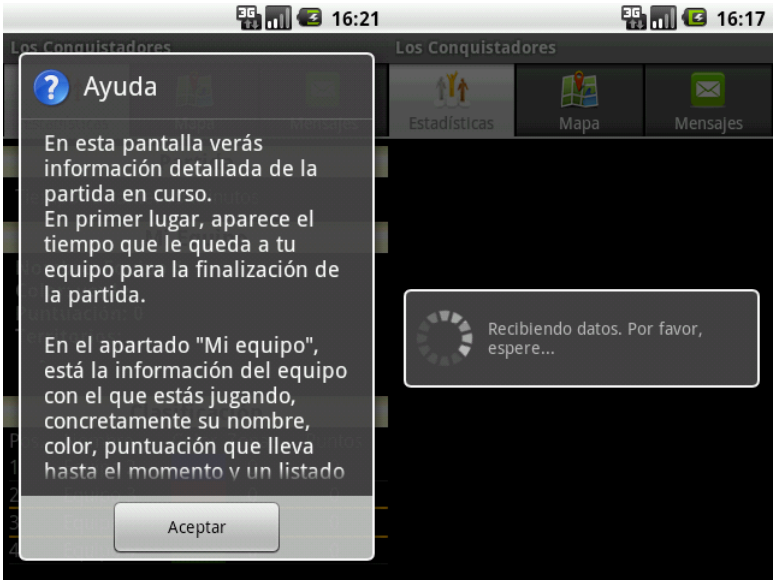


Figura 3.24: Capturas del menú de ayuda y ejemplo de un *ProgressDialog*.

También cabe mencionar que durante la ejecución de la aplicación se pueden producir errores como el fallo de conexión o la imposibilidad de tratar correctamente la respuesta enviada por el servidor a una petición del cliente. Estos hechos se informarán al usuario mediante la aparición de un *AlertDialog* con un icono de advertencia. De esta manera se asegura que el usuario tiene constancia del error que se ha producido, ya que para poder continuar tiene que interaccionar con la ventana y si depende de él, poder tomar las medidas oportunas para solucionar el problema.

La partida finalizará cuando se cumpla el tiempo fijado por el organizador. Cuando esto ocurra, se realizará un *Intent* a la clase *Estadísticas*(implementada mediante la herencia de la clase *Activity*) con la peculiaridad que lo primero que visualizará el equipo será un *AlertDialog*, como se puede ver en la figura figura 3.25, indicándole que el juego a terminado y que acuda al lugar de origen para verificar su puntuación final. Una vez que el usuario pulse sobre el botón "Aceptar" de la ventana emergente, se cerrará y podrá observar el marcador final de la partida. Para poder salir y volver a la pantalla inicial, el jugador tendrá que pulsar el botón físico "Atrás" (representado por una flecha). Por último, señalar que si el usuario lo desea también puede abandonar en cualquier momento, incluso sin que se haya cumplido el tiempo límite asignado a la partida en curso. Para poder

realizar esta acción, la *Activity* que se encuentre en primer plano, esto es, la que está visualizando el participante, tiene que ser una de las pestañas mencionadas arriba. Pulsando el botón físico "Atrás", se le mostrará un *AlertDialog* para que confirme si realmente desea abandonar la partida. Si el equipo acepta, se volverá a la pantalla inicial cerrando todos los servicios que la aplicación estaba ejecutando en según plano.

Figura 3.25: Finalización de la partida y notificación para abandonar el juego.

3.7.1. Estadísticas de la partida

La pestaña estadísticas esta implementada mediante una *Activity* y como ya se ha comentado un poco más arriba, es la primera pantalla que se le muestra al usuario cuando empieza la partida. El objetivo de esta pestaña es mantenerle informado de sus avances a lo largo del juego y lo más importante, la progresión de los otros equipos participantes.

Como se puede observar en la figura 3.26, lo primero que se muestra es el tiempo restante para la finalización del juego, dato a tener muy en cuenta por el usuario para que pueda amoldar su estrategia según el tiempo que le reste. A continuación, se expone información resumida del equipo con el que se participa, estos datos son el nombre y color, puntuación que ha logrado conseguir hasta el momento y un listado de los nombres de los territorios que posee. Estos nombres pueden ser de utilidad para una localización más rápida de un determinado territorio en el mapa del juego. Por último, una tabla de todos los participantes,

ordenados según su puntuación, con la posición, nombre, color y el número de territorios. De esta forma el jugador conoce en todo momento el avance de los otros equipos, y si lo considera oportuno, hacer un cambio en su forma de plantearse la partida según los datos visualizados en esta pantalla.

Figura 3.26: Pantalla de la pestaña estadísticas.

Cada vez que se pulse sobre esta pestaña, la aplicación hará una petición HTTP usando el método *GET*, y si todo va bien el servidor le enviará en formato JSON todos los marcadores de los equipos. Posteriormente se tratará dicho objeto y la clase *Estadísticas* mostrará toda la información.

3.7.2. Mapa de la partida

En la pestaña mapa los equipos participantes pasarán la mayor parte de tiempo, porque es donde el usuario interactuará con los principales recursos del juego, es decir, con los territorios y los demás equipos. Para utilizar el servicio de Google Maps en la aplicación es necesario que el desarrollador se registre en este servicio aceptando los términos y condiciones de uso para la obtención final del Maps API Key [36].

Todas las aplicaciones Android tienen que estar firmadas con un certificado que le vincule a su desarrollador. Hay dos maneras de firmar una aplicación:

- El SDK de Android proporciona el fichero *debug.keystore* para generar claves de prueba. De esta forma, Google pretende facilitar a los desarrolladores

la creación y prueba de la aplicación.

- Si el objetivo es distribuir o subir la aplicación a Google Play, es necesario firmarla con las claves generadas por nuestro propio keystore.

La creación de la huella digital de certificado (MD5) se ha obtenido mediante la herramienta keytool del SDK de Java, que hace uso del keystore de nuestro ordenador. A continuación se ingresa el MD5 en la web de registro para obtener el Maps API Key. Esta llave será introducida en el correspondiente fichero xml para la creación del mapa y poder hacer uso de este en la aplicación. Por último, es necesario añadir en el fichero *manifest.xml* la referencia a la librería utilizada para la creación del mapa.

Las clases que permiten crear y gestionar los mapas en Android se encuentran localizadas en el paquete *com.google.android.maps*. A continuación se listarán las principales clases utilizas en la implementación del mapa junto con una breve descripción.

- **GeoPoint**: Representa un punto en el mapa, compuesto por la latitud y longitud.
- **MapView**: Es la vista donde se muestra el mapa obtenido del servicio Google Map.
- **MapController**: Clase para manejar los desplazamientos y zoom en el mapa.
- **MapActivity**: Clase base que proporciona los elementos necesarios a una *Activity* que muestre un *MapView*.
- **Overlay**: Permite representar los diferentes elementos (capas) sobre el mapa.
- **ItemizedOverlay**: Clase que hereda de *Overlay* y sirve para manejar un listado de *OverlayItem*. Entre otras cosas permite definir la representación y tratar los eventos de pulsado sobre un elemento del mapa.
- **OverlayItem**: Es el componente básico de cualquier *ItemizedOverlay*. Este elemento contiene el *GeoPoint* del elemento a representar.

La clase *Mapa* hereda de *MapActivity* consiguiendo que tenga las funcionalidades de una *Activity* con el añadido de visualización y manejo de mapas. Para ello

es necesario la creación de dos objetos básicos inherente a todo Google Map: el *MapView* y *MapController*. El primero de ellos se obtiene a través de la vista del fichero xml que contiene el atributo *MapView* y el Maps API Key ya mencionado en este apartado. Para conseguir el objeto *MapController* es necesario invocar el método *getController()* del objeto *MapView*, y de esta manera el desarrollador puede fijar el nivel inicial de zoom en el mapa.

Se ha utilizado el control de manejo, proporcionado por el método *setBuiltInZoomControls()*, que permite al usuario seleccionar el nivel de zoom que le resulte más cómodo durante el juego. También para su beneficio se ha incluido, en la parte inferior izquierda de la pantalla, un botón cuya funcionalidad es centrar el mapa en el *GeoPoint* asociado al equipo y fijar el nivel de zoom por defecto (ver figura 3.27). De esta forma el usuario puede desplazarse a zonas lejanas del mapa y volver rápidamente a su posición tan solo pulsándolo.

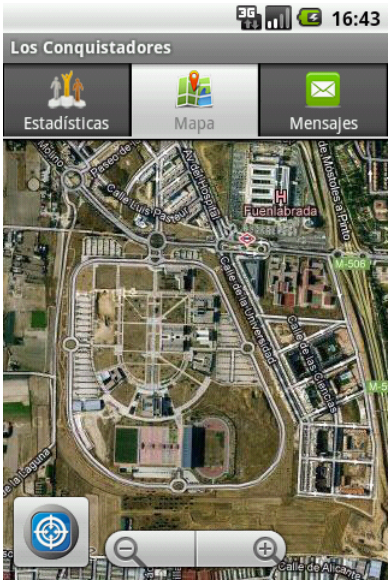


Figura 3.27: Captura del mapa con el control de zoom y el botón de centrado.

Obtención de los elementos del mapa

Una vez visualizado el mapa y que es posible hacer acciones básicas como desplazamientos y control del zoom, es el momento de solicitar los principales elementos de la partida (equipos y territorios) para posteriormente añadirlos al mapa. Para esta tarea se han creado dos nuevos servicios, como recordatorio, un *Service* en Android es un componente de la aplicación que no requiere de la

interacción del usuario y que se ejecuta en segundo plano (*background*) realizando una determinada tarea.

El primero de ellos se ha implementado en la clase *ServicioLocalizacion*, en grandes rasgos, este servicio siempre estará activo obteniendo la localización geográfica (latitud y longitud) del equipo y la almacenará en una variable estática global para que sea accesible por otras clases mediante la invocación del método estático *getLocation()*. Otra acción esencial que desarrolla este servicio es la de enviar al servidor dicha localización, mediante la utilización del método *POST* soportado por el protocolo HTTP, para su almacenamiento y posterior lectura por parte de los clientes Android. Hay dos maneras de conseguir la localización de un terminal móvil [37]:

- Acceso mediante puntos WiFi o en su defecto por la triangulación de antenas de las operadoras móviles: Esta técnica tiene la ventaja de ser soportada por cualquier teléfono móvil, su consumo de batería es menor y la localización se puede obtener tanto en lugares abiertos como en interiores, pero tiene un gran inconveniente y es que su falta de precisión puede llegar a ser considerable.

- Utilización del servicio GPS del terminal móvil: La ventaja de este método es su gran precisión, pero como puntos negativos cabe destacar su mayor consumo de batería, sólo funciona en exteriores y dependiendo de algunas circunstancias el retorno de la localización puede demorarse.

En este proyecto se ha primado la precisión frente a otros factores, debido a que es fundamental en el óptimo desarrollo del juego. Muchas acciones dependen de la distancia del equipo con los otros recursos, por eso y aun conociendo las desventajas comentadas, se ha utilizado el GPS como método para conseguir la ubicación. Para poder tener acceso a este servicio del terminal, es necesario añadir el permiso *ACCESS_FINE_LOCATION* en el fichero *manifest.xml* del proyecto. La localización de un equipo ira actualizándose por medio de la clase *MyLocationListener* cada cierto periodo de tiempo o si se recorre una cierta distancia, estos parámetros están configurados en la aplicación a 5 segundos y 12 metros respectivamente. Se tomó esta medida para disminuir el consumo de batería pero con el inconveniente de una leve pérdida en el periodo de las actualizaciones de los equipos.

Antes de implementar el anterior servicio, se pensó en utilizar la clase *MyLocationOverlay* de la API de Android. Esta clase proporciona al desarrollador los

métodos necesarios para obtener y dibujar la posición del usuario, pero al final esta idea no se llevo a cabo porque su comportamiento no se ajustaba a las necesidades del proyecto. Cuando esta clase trata de redibujar la posición del equipo, el usuario puede encontrarse en una pantalla diferente a la del mapa y esto puede causar problemas en la ejecución de la aplicación al querer añadir capas en una *Activity* que no encuentra en primer plano. Otro problema que se divisó fue el mayor uso del GPS comparado con la solución adoptada, esta clase no permite configurar que tiempo o distancia tiene que transcurrir para adquirir las nuevas coordenadas, si no que la actualización es constante por lo que ocasiona un mayor gasto energético.

El segundo servicio está implementado en la clase *ServicioMapa*, al contrario que el anterior, el funcionamiento de este se ve ligado al *MapActivity*. Si el usuario no se encuentra en la pantalla del mapa este servicio se detendrá ya que no es necesario su funcionamiento, reduciendo el uso del procesador y alargando la duración de la batería. En caso contrario, cada vez que el jugador vuelva a la *Activity* del mapa, este servicio se iniciará y será el encargado de comunicarse con el servidor en periodos de 5 segundos para obtener los posibles cambios que han sufrido los territorios y las coordenadas latitud y longitud de los equipos participantes. Cabe recordar que la ubicación de los territorios no cambia por lo que no es necesario solicitar dichas coordenadas continuamente. Esta información solo se obtiene una vez y es al principio de la aplicación, cuando se selecciona la partida.

Una vez recibida la información del servidor, se procederá a la modificación de los *ArrayList* equipos y territorios de la clase *Partida*. De esta forma se mantiene actualizadas las estructuras de datos alojadas en memoria de la aplicación para su uso posterior. Si el servicio no tiene ningún impedimento en la obtención de los datos, ordenará a la clase *Mapa* el repintado de los equipos y territorios mediante el envío de un mensaje en *broadcast*. Para ello se registra previamente un *BroadcastReceiver* con la misma acción que llegará del mensaje enviado por el servicio. Como se recomienda cada vez que se registra un *BroadcastReceiver* en el *onResume()*, es necesario anularlo en el método *onPause()*, ya que no se va a recibir mensajes dirigidos a ese *broadcast* evitando sobrecargar el sistema innecesariamente.

En caso de tener algún problema ya sea en el envío o recepción de la respuesta por parte del servidor, se notificará al usuario el error producido mediante un *Toast* de la API de Android. Este tipo de notificaciones se muestran en la pantalla durante un breve periodo de tiempo para luego desaparecer sin que se

requiera ninguna acción por parte del usuario y sin interferir en las acciones que se están realizando. Las características de los *Toast* los hacen ideales para mostrar mensajes sencillos pero se desaconseja su uso para realizar notificaciones con un determinado grado de importancia.

Dibujar los elementos del mapa

Una vez creado el mapa es el momento de representar sobre este los elementos conseguidos por los dos servicios anteriores. Por tanto los recursos que se van a dibujar y que el usuario verá son los siguientes (ver figura 3.28):

- La posición del equipo constituido por un punto azul, un círculo que determina la zona de acción y el indicador de dirección, incluido después de la primera prueba (apartado 4.2.1) para mejorar la orientación.
- Un icono con la imagen de un castillo representando los diferentes territorios de la partida cuyo color es el del equipo propietario.
- Un icono de usuario representando a los restantes equipos con el color que tienen asociado.

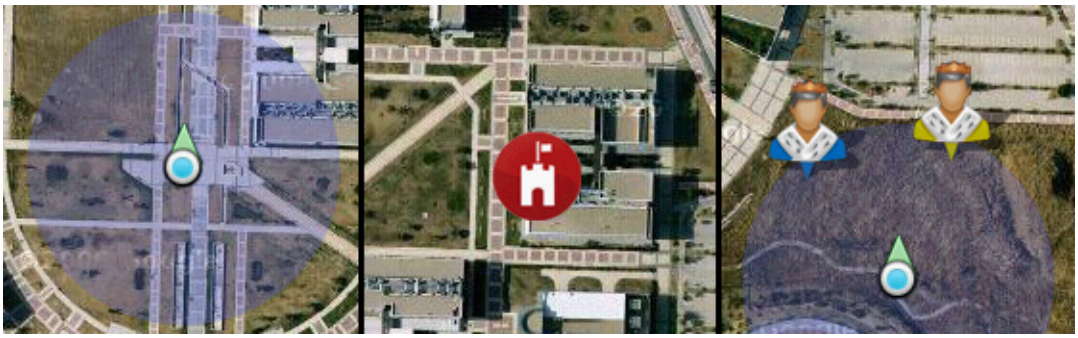


Figura 3.28: Captura de los diferentes iconos en el mapa.

En un mismo mapa se pueden representar múltiples objetos *Overlay* independientes gracias a la implementación de diferentes clases. A continuación, se explicarán las clases creadas que permiten controlar los eventos y dibujar de una manera personalizada al equipo del usuario, los territorios y los demás equipos participantes.

Para el dibujado de la ubicación del usuario se ha implementado la clase *MiEquipoItemizedOverlay* que extiende de *ItemizedOverlay<Item>*. Esta última clase

se encuentra en el paquete *com.google.android.maps* y permite representar una lista de capas en un Google Maps con la posibilidad de individualizar el elemento a dibujar. Para personalizar el icono que representa la ubicación del equipo, fue necesario sobrescribir el método *draw()* de la clase *MiEquipoItemizedOverlay*. De esta forma se pudo añadir el círculo concéntrico, que permite determinar si se encuentra a la suficiente distancia para la interacción con el recurso, y la flecha rotatoria que facilita la orientación en el mapa.

Análogamente, se crearon las clases *TerritorioItemizedOverlay* y *EquipoItemizedOverlay* para el dibujado de los territorios y resto de los participantes. Estas clases también heredan de *ItemizedOverlay<Item>* por lo que permiten personalizar los iconos que representan a cada elemento. Cabe destacar que los equipos rivales solo serán visualizados al usuario si se encuentran dentro de su círculo de acción. Para conocer la distancia que separa a ambos equipos, se utilizó el método estático *distanceBetween()* de la clase *Location*.

Manejar los elementos dibujados

No solo *ItemizedOverlay<Item>* permite dibujar múltiples elementos en el mapa, también permite manejar los eventos que se realizan sobre ellos, como por ejemplo, cuando se pulsan. El método *onTap()* se encargará de capturar la pulsación del usuario sobre un elemento y mostrar un menú implementado mediante un *AlertDialog* (visualizar figura 3.29) en el que se mostrarán las diferentes opciones que el recurso permite efectuar al jugador. Para la realización de estos menús emergentes se ha creado la clase *MenuFila* cuyo objetivo es definir la estructura de cada opción, formada por un icono y un titulo que la identificará unívocamente. Posteriormente se crea un *array* de objetos *MenuFila* que será utilizada para la creación de un *ArrayAdapter*, este adaptador proveerá de datos al control de selección. De esta manera el usuario puede visualizar e interactuar con las diferentes opciones del menú.

3.7.3. Acciones realizadas sobre el mapa

En los siguientes subapartados se analizarán las diferentes acciones que un usuario puede realizar sobre los territorios y demás equipos de la partida.

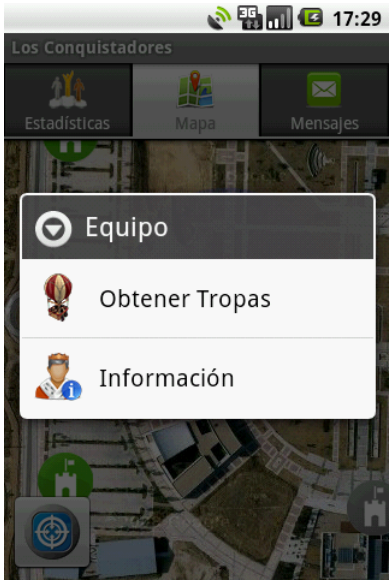


Figura 3.29: Captura del menú asociado al equipo del usuario.

Conquista de un territorio

Los territorios son los elementos más importantes en el juego y por eso es fundamental una buena estrategia para su conquista y mantenerlos de los ataques que recibirán por parte de los rivales. Se pueden identificar dos tipos de territorios: los neutrales y los que tienen propietario. Inicialmente todos estarán en un estado neutral, es decir, no pertenecerán a ningún equipo y su conquista será inmediata.

Para realizar la conquista de un territorio, el jugador pulsará sobre el icono que lo representa e inmediatamente se le mostrará un menú con las acciones disponibles, entre las que se encuentra la opción "Conquistar". Cuando el usuario pulse sobre esta, la aplicación determinará si se encuentra a la distancia suficiente para poder realizar la acción. El cálculo de esta distancia se realizará mediante el método estático *distanceBetween()* de la clase *Location*. En el caso de que el territorio se encontrase fuera de la zona de acción, el equipo no podría realizar la conquista y se lo notificará mediante un *Toast* (ver figura 3.30).

La obtención de un territorio neutral será inmediata y no será necesario que la tropa asociada al equipo entre en juego, porque se entiende que es un elemento pacífico y no es necesario establecer un combate para conseguirlo. Una vez que se ha pulsado la opción en el menú y se encuentra a una distancia correcta, se utilizará el método *POST* para la establecer la conexión. Cuando se reciba la petición por parte del servidor, se comprobará que dicho recurso no ha sido modificado, es decir, que sigue siendo un territorio neutral. Es esencial realizar este

tipo de comprobación debido a las posibles condiciones de carrera que puede sufrir un elemento compartido, en este caso un territorio. Por ejemplo, si dos equipos realizan simultáneamente una conquista sobre una misma zona, lo deseable es que el primero que la realizó se la quede y el segundo se le informe, como puede verse en la figura 3.30, de que su conquista no pudo realizarse con éxito. No tratar este tipo de fenómenos puede ocasionar situaciones no deseadas y que la experiencia del juego se vea afectada muy negativamente.

Figura 3.30: Pantallas con las notificaciones que impiden la conquista de un territorio.

Si la comprobación se ha llevado con éxito, el servidor asignará al territorio el nuevo dueño y comunicará la respuesta a la aplicación para notificar mediante un *Toast* el éxito de la conquista. En este momento el usuario contará con una nueva zona y las tropas que contiene.

En la conquista de una zona con propietario entran en juego las tropas del equipo atacante y las del territorio defensor. Cuando el jugador seleccione la opción "Conquistar" del menú se realizará una petición *GET* con el fin de obtener todos los datos necesarios para la simulación de la batalla. Una vez recibida la respuesta del servidor, la aplicación abrirá un nueva *Activity*, como se puede ver en la figura 3.31. En ella, el usuario puede ver el nombre del territorio defensor, el número total de unidades que tiene el equipo junto a la cantidad máxima de la que puede disponer, y un campo para que el usuario introduzca el número exacto de tropas que participarán en la contienda. Cuando el usuario pulse el botón de esta pantalla, se comprobará que el campo mencionado arriba no esté vacío o

contenga un número superior al de las tropas disponibles. En tal caso, el problema será notificado mediante un *Toast*. Si todo está en orden se realizarán los cálculos del enfrentamiento en la aplicación, con los datos obtenidos previamente del servidor y la cantidad introducida por el usuario. A continuación se listan, de forma resumida, las principales normas que regirán la simulación:

- El territorio defensor siempre contará con una bonificación del 20 % en el número de sus tropas.
- En todas las conquistas, a cada parte se le asignará un porcentaje de victoria según la cantidad de sus unidades. A mayor número de tropas, más posibilidades de victoria.
- Si alguna de las partes implicadas duplica el número de unidades a su adversario, ganará la batalla automáticamente.
- En todos los combates, las tropas restantes de cada bando dependerán de sus respectivos porcentajes de victoria. A mayor porcentaje, más unidades sobrevivirán.
- Si el equipo atacante es el ganador, conseguirá el territorio y las tropas resultantes de la batalla se quedarán en este. Las unidades supervivientes derrotadas pasarán a formar parte de la escolta del equipo defensor.
- Si el territorio se defiende no cambiará de propietario. El resto de la tropa atacante se unirá a la escolta del equipo.

Una vez realizada la simulación, la aplicación enviará el resultado de la contienda (ganador y el número de tropas restantes de ambos bandos) mediante una petición *POST*. Lo primero que comprobará el servidor será si se han producido condiciones de carrera, es decir, si algún elemento implicado durante el proceso de la conquista se ha visto modificado. Si esto se produce, se enviará la respuesta al cliente y mediante un *AlertDialog* se le notificará al usuario que la acción no se ha podido realizar. Si por el contrario la comprobación es correcta, se realizarán las correspondientes modificaciones según el resultado de la simulación y el usuario será notificado, mediante un *Toast*, del resultado en el enfrentamiento (ver figura 3.31).

Finalmente cabe destacar que por cada conquista lograda, ya sea un territorio neutral o rival, se le añadirán 25 puntos a su marcador. Esta acción es la que mayor puntos proporciona a los jugadores.

Figura 3.31: Capturas con el proceso de la conquista de un territorio rival.

Espionaje de un territorio rival

El espionaje será de gran ayuda para desarrollar una buena estrategia a la hora de conquistar un territorio rival. Inicialmente cada equipo contará con un número limitado de espías, que serán establecidos durante la creación de la partida en el servidor. Los espías proporcionan una valiosa información del rival, la cantidad de unidades que defienden esa zona. De esta forma el equipo atacante puede realizar un asalto más preciso y minimizar así las bajas de sus tropas. Esta acción puede ser realizada desde cualquier punto del mapa por lo que no es necesario acercarse al objetivo para llevarla a cabo.

Esta opción del juego está siempre disponible en el menú de cualquier territorio rival bajo el nombre de "Espiar". La aplicación lleva la cuenta de los espías utilizados durante el transcurso de la partida, por este motivo se comprobará si todavía se dispone de alguno cuando el usuario pulse la opción. En caso de que el número de espías del equipo sea cero, la aplicación se lo notificará mediante un *Toast* y no será necesario realizar una conexión con el servidor.

En caso contrario, se establecerá la conexión para solicitar la información requerida. Una vez recibida la respuesta y tratada en el lado del cliente, se decrementará el número de espías del equipo y se le presentará un *AlertDialog*, como se puede ver en la figura 3.32, que contiene el nombre y número de tropas que defienden.

Figura 3.32: Pantalla con la información del territorio espiado.

Movimiento de tropas

Como ya se ha explicado anteriormente, las conquistas se realizan con las unidades disponibles en el equipo y es probable que durante el transcurso de la partida, el usuario necesite reunir más tropas de sus territorios y que formen parte de su escolta para contar con una mayor probabilidad de victoria en la batalla. También es útil poder distribuir las unidades entre las diferentes zonas de su posesión, por si el jugador considera oportuno defender uno o varios territorios desprotegiendo algo más el resto. Cabe destacar que al igual que la acción anterior, esta también puede ser ejecutada sin tener en cuenta la distancia que separa a ambos.

Esta función se encuentra en los menús del propio equipo y de sus territorios con el nombre "Obtener tropas". Si el usuario la selecciona en el menú de su equipo y no posee ningún territorio de donde se puedan adquirir unidades, la aplicación le informará de este hecho mediante un *AlertDialog*. En caso de que el equipo tenga una o más zonas, la aplicación realizará una petición *GET* requiriendo la información de todas sus posesiones y de su equipo, para ser mostrada en la nueva *Activity* que se abrirá seguidamente (visualizar figura 3.33).

En esta pantalla, el usuario puede elegir el origen de donde se van a adquirir las tropas mediante el uso de un *Spinner* de la API de Android. El destino será el recurso sobre el que se realizó la acción en el mapa y se mostrará el nombre y cantidad de las tropas que este contiene. También aparece la cantidad máxima

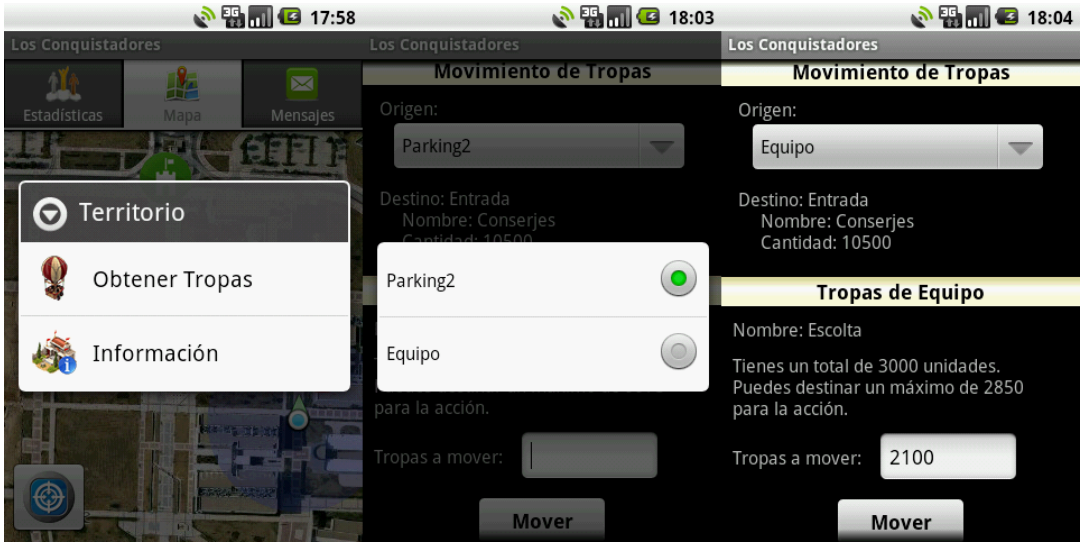


Figura 3.33: Pantallas con el menú del territorio y movimiento de tropas.

de unidades que podrá disponer según el origen seleccionado y un campo para que introduzca la cantidad exacta que será trasladada. Toda esta información es mostrada al jugador para que obtenga una idea general del estado de sus territorios y equipo, con el objetivo de facilitar el intercambio. Cuando el usuario pulse el botón "Mover" de la GUI, se comprobará que el campo (implementado con la clase *EditText*) no se encuentre vacío o contenga un valor superior al de las unidades disponibles para la acción. En ese caso, se le notificará el error a través de un *Toast* como se puede ver en la figura 3.34.

Figura 3.34: Notificación al introducir valores incorrectos en el EditText.

Una vez confirmado que todo está en orden, se enviará una petición *POST* al servidor para que realice los cambios oportunos. Como en algunos de los casos ya explicados, se verificará si se ha producido algún problema de concurrencia mientras la acción transcurría y si todo está correcto, poder llevar a cabo las modificaciones necesarias. Finalmente el servidor enviará la respuesta para que el cliente notifique al usuario si la acción se realizó correctamente o no.

Información del equipo y territorios del usuario

A lo largo de la partida es esencial que el usuario tenga la posibilidad de obtener información relativa a su equipo y territorios todas las veces que lo desee, sin necesidad de desplazarse al objetivo. Esto es fundamental para que el jugador elija la mejor estrategia teniendo en cuenta el estado de sus recursos.

Para adquirir los datos de su equipo, el jugador tendrá que pulsar sobre su icono y seleccionar la opción "Información" del menú. Acto seguido, se solicitará al servidor los datos requeridos para que posteriormente el cliente pueda mostrar la información formada por el nombre del equipo, número de escolta y espías restantes, como se puede observar en la figura 3.35.

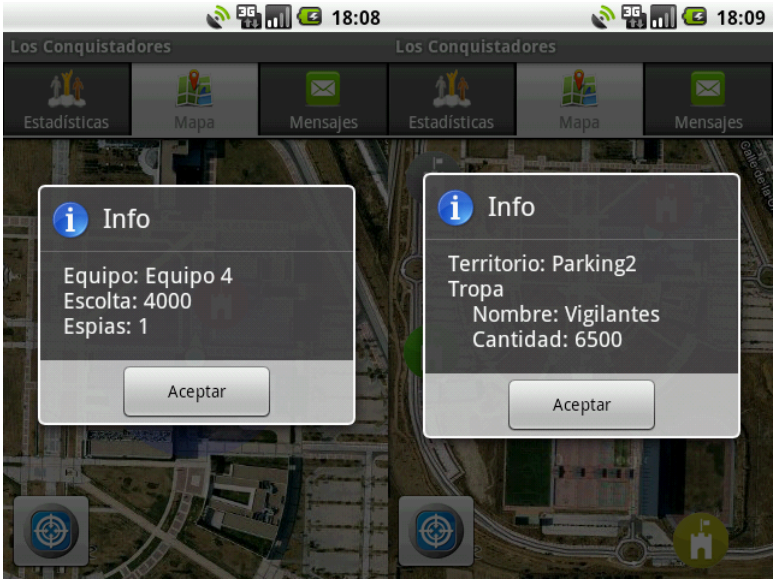


Figura 3.35: Capturas con la información del equipo y territorio del usuario.

De forma similar, el usuario obtendrá los datos de cualquiera de sus zonas, para ello solo será necesario pulsar sobre el icono del objetivo y seleccionar la opción "Información". Seguidamente se mostrará en un *AlertDialog*, el nombre del territorio y los datos de la tropa ubicada en este.

Duelo entre equipos

En cualquiera de las acciones anteriores, el usuario no interaccionaba con ningún participante, por lo que se añadió esta funcionalidad a la aplicación. Como ya se indicó el usuario no ve a los demás equipos en el mapa, excepto cuando están lo suficientemente cerca y entran dentro de su área de acción, es en este momento cuando pueden retarse.

Para que puedan enfrentarse uno con otros, tendrá que pulsar sobre el icono del contrario y seleccionar del menú la opción "Duelo". Entonces la aplicación comprobará si el usuario ha efectuado recientemente esta acción, y de ser así, se lo notificará mediante un *Toast* (ver figura 3.36) de la imposibilidad de realizar el desafío. Esta medida se ha tomado para evitar un mal uso de esta funcionalidad durante el juego, como por ejemplo que un equipo con una escolta muy superior a la de su rival, le esté lanzando duelos reiteradamente, ganándolos y sumando puntos rápidos y fáciles. Se entiende que esta no es la filosofía que se pretende buscar en el juego, ya que el rival no tendría la mínima posibilidad de defenderse en ninguna ocasión mientras el otro equipo ha conseguido una gran suma de puntos en su marcador.

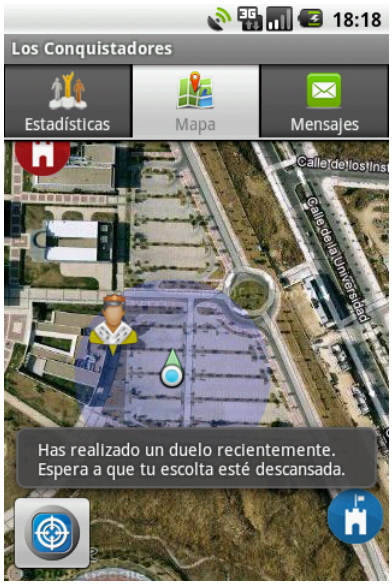


Figura 3.36: Notificación que impide la realización del desafío.

Si se permite su realización, se solicitará al servidor los datos necesarios que básicamente estarán formados por la cantidad de escolta de ambos equipos. A diferencia de la conquista de un territorio rival, el equipo no podrá elegir la cantidad

de unidades si no que será la totalidad de su escolta la que participará en el duelo. Una vez que la aplicación ha obtenido correctamente los datos, se procederá a la simulación fundamentada en las siguientes reglas:

- A cada equipo se le asignará un porcentaje de victoria según la cantidad de su escolta. A mayor número, más posibilidades de ganar.
- Si alguna de las partes implicadas duplica el número de unidades a su adversario, ganará el desafío automáticamente.
- Las tropas restantes de cada bando dependerán de sus respectivos porcentajes de victoria. A mayor porcentaje, más unidades sobrevivirán.

Cuando la simulación haya terminado, el cliente Android conectará con el servidor para enviar el identificador del equipo ganador y la escolta resultante de ambos. Por su parte, el servidor detectará posibles casos de concurrencia y si se producen, desechar el resultado del duelo y comunicar del problema al cliente. Por último, la aplicación procesará la respuesta JSON y notificará al usuario el resultado del enfrentamiento mediante el uso de un *Toast* (ver figura 3.37).

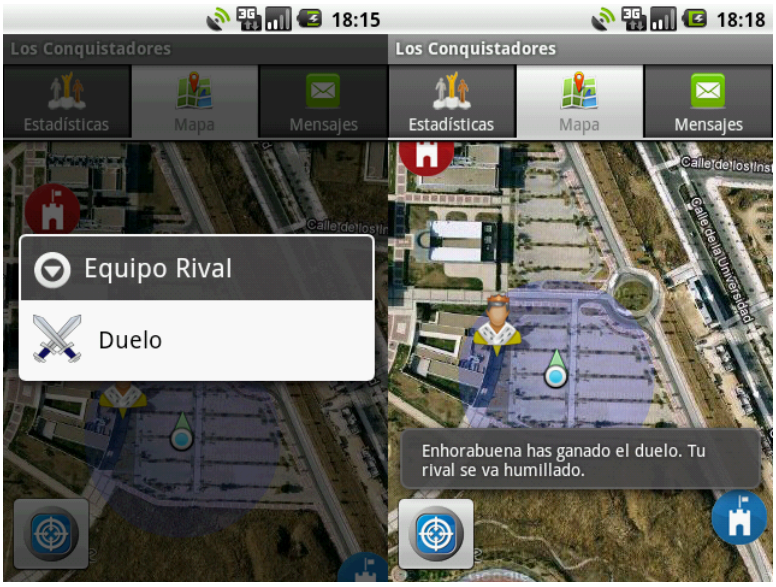


Figura 3.37: Pantallas con la realización exitosa de un duelo.

Este tipo de acción aportará al ganador, aunque no haya sido el que la efectuó, un total de 15 puntos. Junto a las conquistas son las dos únicas operaciones que permiten incrementar los marcadores de los jugadores.

3.7.4. Mensajería

Se ha implementado un sistema de mensajería interno en la propia aplicación para facilitar la comunicación del usuario con el organizador y demás participantes. Otro objetivo que se desea conseguir con este sistema es que el jugador puede desarrollar nuevas estrategias, como la formación de alianzas virtuales con los otros equipos y conseguir recortar puntos a rivales con mejor posición en la clasificación. En la figura 3.38, se puede advertir la pestaña de Mensajería con las diferentes opciones que ofrece al usuario: escribir un nuevo mensaje, listar los mensajes recibidos o enviados.

La GUI destinada para la creación de un nuevo mensaje se ve en la figura 3.38. El jugador puede elegir el destinatario mediante el uso del *Spinner* y escribir el texto del mensaje en el cuadro destinado a ello. Cuando se pulse el botón para remitirlo, la aplicación comprobará si el cuerpo del mensaje está vacío y si es así, no se procederá a su envío y se le informará. Si todo esta correcto, se mandará el mensaje al servidor para su almacenamiento y posteriormente generar una respuesta para el cliente. Siempre el usuario será notificado mediante un *Toast*, tanto si el envío se ha realizado correctamente o se produjo algún problema durante el proceso.

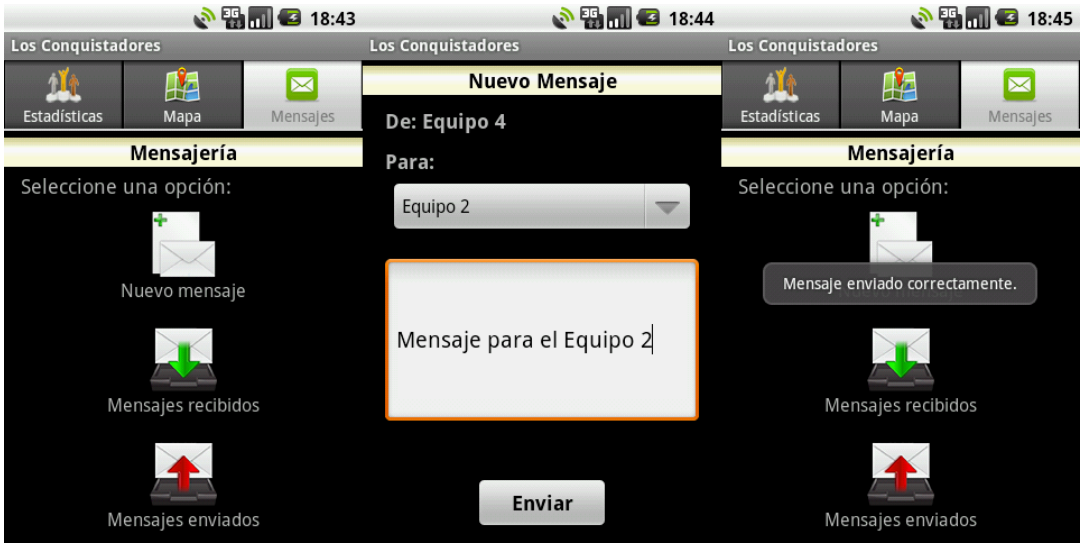


Figura 3.38: Capturas de la pestaña Mensajería y el envío de un nuevo mensaje.

Cuando se quiera consultar los mensajes recibidos o enviados a lo largo de la partida, se abrirá una nueva pantalla implementada por la clase *ListActivity* con

el listado de los mensajes obtenidos del servidor, ordenados según la fecha de creación. En cada componente del listado se podrá distinguir el emisor, destinatario y a modo de resumen, la primera línea del cuerpo. Si el usuario pulsa sobre un elemento, se mostrará una nueva *Activity* donde podrá ver más detalles del mensaje seleccionado, como la fecha, hora y el texto completo (ver figura 3.39). También se le dará la opción de responder pulsando el botón de esta GUI, pasando a la pantalla para la creación de uno nuevo.

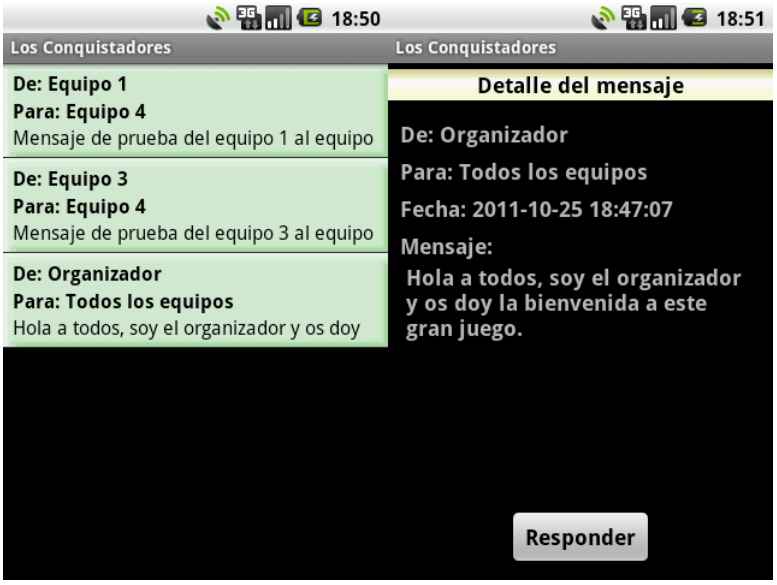


Figura 3.39: Pantallas con el listado de mensajes recibidos y detalle de un mensaje.

Todavía no se ha mencionado una parte fundamental de este sistema y que facilita enormemente a los usuarios saber cuando han recibido un nuevo mensaje, sin que tengan que estar comprobando continuamente su bandeja de entrada. Se trata de la creación de un nuevo *Service* implementado en la clase *ServicioMensajeria*, cuyo objetivo es el de comprobar periódicamente si el equipo ha recibido nuevos comunicados por parte del organizador o de otros participantes y en tal caso, notificárselo.

Para realizar esta tarea el cliente Android solicitará, mediante una petición *GET*, el identificador del último mensaje recibido. Cuando la aplicación compare este nuevo valor con el antiguo almacenado (obtenido en la anterior consulta), se podrá determinar si el equipo tiene nuevos mensajes y si es así, crear una nueva notificación en la barra de estado, como se observa en la figura 3.40. Cuando el usuario pulse sobre la notificación accederá al listado de mensajes recibidos, ya explicado anteriormente.

Figura 3.40: Notificación de la llegada de un nuevo mensaje en la aplicación.

3.7.5. Otros servicios implementados

Para la correcta funcionalidad y lógica de la aplicación, se han diseñado e implementado una serie de servicios que se ejecutan en segundo plano sin que el usuario tenga conocimiento de ello. Algunos servicios como el de localización, mapa y mensajería ya han sido explicados detalladamente a lo largo de este apartado, pero todavía quedan por explicar dos servicios más. A continuación se analiza el funcionamiento de estos componentes:

Servicio del equipo

Este servicio se encargará periódicamente de solicitar al servidor, mediante una petición *POST*, el incremento de unidades en cada territorio y escolta del equipo, según el factor de crecimiento fijado en la creación de la partida. Además de sumar en su marcador 10 puntos por cada territorio en su poder. Una vez recibida y procesada la respuesta en la aplicación, se generará un *Toast* para informar al usuario de las modificaciones en sus territorios, equipo y marcador.

Servicio de enfrentamientos

Se trata de un servicio que periódicamente comprobará si el usuario ha sufrido algún tipo de enfrentamiento, para ello se realiza una petición *GET* al servidor con

el fin de obtener los últimos identificadores de los duelos y conquistas recibidos, y almacenarlos en el cliente. Posteriormente se comparará estos últimos valores con los obtenidos en la anterior consulta, en el caso de que se detecten nuevos enfrentamientos se generará una nueva notificación en la barra de estado.

Cuando el usuario pulse sobre la notificación se abrirá una nueva *ListActivity*, como se puede ver en la figura 3.41, obteniendo del servidor el listado de las últimas conquistas y/o duelos recibidos. En esta pantalla el jugador visualizará un icono y una descripción que identificarán cada tipo de enfrentamiento. En el caso de los duelos se indicará si ha conseguido vencer o no, y en las conquistas si ha logrado mantener el territorio atacado.

Figura 3.41: Notificación de un nuevo enfrentamiento y el listado de los últimos ataques recibidos.

Capítulo 4

Pruebas del sistema

En este capítulo se presentan las diferentes pruebas realizadas al sistema con el objetivo de verificar que cumplen los requisitos marcados al inicio y encontrar los errores que se han podido cometer durante la fase de implementación. Se van a distinguir dos tipos: las realizadas durante el desarrollo del PFC y las ejercidas en un entorno real con voluntarios y dispositivos físicos.

4.1. Pruebas en el entorno de desarrollo

Una vez implementado el sistema, es necesario realizar una serie de pruebas para comprobar su correcto funcionamiento y solventar los errores que se van encontrando. Estas se llevaron a cabo en la misma máquina donde se realiza el proyecto, facilitando enormemente su desempeño.

Inicialmente se verifica el sitio web mediante la utilización del servidor de pruebas que incorpora Django. Para ello se arranca y a través del manejo de un navegador web, se comprueba la correcta conexión con el sitio. Posteriormente, se accede al servicio web para ejecutar las funcionalidades disponibles y al mismo tiempo se constata la gestión de la base de datos, es decir, si se crean, almacenan y obtienen correctamente los datos generados. Como es lógico, en caso de encontrar algún tipo de error, se soluciona y vuelve a repetirse este proceso.

Por otro lado se encuentran las pruebas realizadas a la aplicación móvil cuyo objetivo primordial es comprobar la correcta lógica del juego y detectar los posibles fallos. Para logar este fin se empleó el emulador proporcionado en el Android SDK. Este emulador permite lanzar la aplicación como si estuviese instalada en un dispositivo real, de tal manera que permite comprobar cómo se producen las

conexiones con el servidor y simular situaciones que de otra manera serían bastante difíciles de realizar.

Sobre este último aspecto, ha sido de vital importancia la utilización del comando *geo fix* en la consola del emulador. Indicando la latitud, longitud y altura como parámetros se pueden enviar posiciones geolocalizadas al emulador, igual que si fueran obtenidas a través del GPS. Estas pruebas fueron complementadas con las realizadas sobre un terminal físico, concretamente en un HTC Desire [38] con la versión 2.2 de Android.

4.2. Pruebas en el entorno real

Realizar pruebas en un entorno controlado facilita, de cierta manera, obtener unos buenos resultados. Por este motivo se decidió examinar el sistema en un entorno real, con voluntarios y terminales físicos, donde es más complicado tener bajo control determinados factores que pueden descubrir nuevas vulnerabilidades. A diferencia de las anteriores, en estas se utilizó el servidor de producción de Apache. El principal motivo de este cambio es dar una mayor estabilidad al sistema ya que el servidor de Django, fuera de los entornos de desarrollo, no garantiza su correcto funcionamiento.

Con el objetivo de conocer la valoración y aceptación de los participantes, se les pidió que completaran una encuesta *on-line*. Además de proporcionar estos aspectos tan útiles, también le permite conocer al desarrollador las sugerencias propuestas con el fin de mejorar la aplicación. A continuación se explica el proceso empleado en cada una de las pruebas realizadas, así como el análisis de los resultados obtenidos.

4.2.1. Primera prueba

Con la realización de esta primera prueba se busca comprobar el buen funcionamiento del sistema en general y detectar los máximos de errores para solventarlos en futuros experimentos. También se pretende recabar la mayor información sobre el manejo y diseño de la interfaz por personas foráneas al proyecto, así como analizar las opiniones de los voluntarios.

La primera prueba tuvo lugar el día 16 de Noviembre de 2011 en el Campus de Fuenlabrada de la Universidad Rey Juan Carlos. El grupo de voluntarios

acudió al punto acordado sobre las 15:00h, formado por tres estudiantes de Ingeniería Informática Superior y un ingeniero recién titulado en la misma carrera. Una vez echa las presentaciones se realizó una explicación sobre la dinámica y las principales funcionalidades del juego, que aproximadamente duró 25 minutos. Al final de esta se resolvieron las dudas surgidas con el objetivo de maximizar la experiencia de los participantes. Posteriormente se repartieron los terminales móviles, en concreto cuatro HTC Magic [39], totalmente configurados (conexión de datos, GPS y aplicación instalada) para iniciar la aplicación sin contratiempos. Dado el perfil de los voluntarios, no fue necesario explicar el funcionamiento básico de estos dispositivos.

Explicados los puntos más importantes, se dio comienzo a la partida, previamente creada y alojada en una máquina virtual proporcionada por el departamento del GSyC. Esta constaba de un total de ocho territorios distribuidos a lo largo del Campus y con una duración de 45 minutos. En su transcurso, el organizador se dedicó a controlar la partida e intentar resolver los problemas que pudieran surgir. Poco minutos después del comienzo, un jugador retornó al punto de reunión con problemas graves en el juego por lo que siguió el evento a través del sitio web. Los demás pudieron desarrollar la actividad sin mayores contratiempos.

Al finalizar el tiempo establecido, los jugadores regresaron y se les pidió que realizasen una encuesta para obtener sus impresiones y propuestas sobre la aplicación. Al mismo tiempo se recogieron los móviles y en términos generales, comentaron que la experiencia les había parecido original y divertida. Seguidamente se va a comentar los resultados de esta primera encuesta, recopilada en el anexo A.1.

Resultados de la encuesta

La totalidad de los jugadores vieron muy positivas las explicaciones dadas al comienzo, ya que les fueron bastante útiles para entender mejor la dinámica del juego y fomentar la competitividad. Además, los menús de ayuda fueron consultados por la mayor parte ellos y les sirvieron para solucionar alguna duda durante el transcurso de la partida. En términos generales, la interfaz gráfica fue valorada como muy intuitiva, sencilla de manejar y con un diseño original. Esto redunda en una mejor inmersión por parte de los participantes y más probabilidades de éxito del producto.

La información mostrada en la pestaña “Estadísticas” fue valorada satisfactoriamente e incluso un voluntario sugirió añadir el número de duelos ganados. En cambió, el 75 % de los participantes vieron la necesidad de cambiar la vista

utilizada en la pestaña "Mapa". Esto es debido a que algunos jugadores expresaron cierta dificultad a la hora de orientarse y opinaron que esta modificación ayudaría a reducir este problema. Otro aspecto estimado como original es la implementación de las acciones mediante menús emergentes, esta decisión facilita la interacción con los diferentes recursos.

El sistema de puntuación presentado fue del agrado de los participantes, exceptuando que uno de ellos propuso premiar con 5 puntos cada territorio defendido exitosamente. Sobre el tiempo de reparto de nuevas tropas, tanto en los territorios y en la escolta, hubo opiniones dispares: dos de los voluntarios lo consideraron correcto y no harían ninguna modificación, otro propuso incrementarlo ligeramente y el resto se decantó por reducir este periodo.

Los resultados obtenidos del servicio mensajería fueron favorables, gran parte lo calificaron como fácil en su manejo y sólo uno reveló que añadiría algún tipo de sonido en la notificación generada cuando se recibe un mensaje.

Sobre las dificultades encontradas, comentar que dos equipos tuvieron algún tipo de problema con la aplicación. El más grave interrumpió la participación de uno de ellos debido al bloqueo de su terminal. El otro error, más leve y sin mayores consecuencias, estuvo relacionado con el *thread* encargado de gestionar el tiempo restante de la partida.

Exceptuando al usuario que sufrió el fallo más grave, el resto percibió un buen rendimiento y fluidez de la aplicación. Otro detalle a mencionar es el gasto normal de la batería, similar al de cualquier otro juego con las mismas características. Estos aspectos fueron tenidos muy en cuenta durante el desarrollo, debido a que las limitaciones hardware de los terminales podían llegar a ser un inconveniente. Valoraron muy positivamente la aplicación y la calificaron de bastante divertida, incluso uno de ellos expresó su deseo de volver a repetir la experiencia. En las preguntas finales, se les dieron la libertad de exponer sus opiniones y sugerencias, muy tenidas en cuenta para mejorar o añadir nuevas funcionalidades.

Por último, antes y después de la partida, se tomaron el saldo de las tarjetas y se calculó que el gasto producido por cada uno de los terminales fue de 0.81 euros. Esto era de esperar debido a la tarifa contratada de Vodafone, que permite utilizar la misma conexión durante dos horas sin limitación en el tráfico de datos. Destacar que se trata de un coste económico y al mismo tiempo se consigue pasar un buen rato con los amigos al aire libre.

4.2.2. Segunda prueba

Una vez analizados los resultados y obtenido una serie de conclusiones, era necesario realizar una nueva prueba donde verificar las últimas modificaciones introducidas y correcciones de los errores detectados. También es el momento ideal para volver a comprobar la robustez del sistema y recoger la opinión de un nuevo grupo sobre la aplicación y otros temas relevantes.

La segunda prueba se celebró el día 19 de Enero de 2012 en el Campus de Fuenlabrada de la Universidad Rey Juan Carlos. Un requisito esencial era encontrar un grupo que no tuviese nada que ver con el anterior, por este motivo el tutor colaboró anunciando el evento en el foro de la ETSIT (Escuela Técnica Superior de Ingeniería de Telecomunicaciones). El resultado fue un conjunto de cinco asistentes, todos ellos estudiantes de ingeniería de la misma escuela. Una vez reunidos y realizadas las presentaciones, sobre las 15:10h comenzó una breve exposición de las características y funcionalidades del juego. A la finalización, se resolvieron las dudas y se repartieron los móviles, en concreto tres HTC Magic y un HTC Desire. A diferencia de la prueba anterior, en esta se formó un equipo de dos personas, dato relevante y recogido en la posterior encuesta.

Resueltos estos preliminares, la partida se inició sin mayores contratiempos. Las características de esta fueron similares a la del anterior experimento, es decir, 45 minutos de juego, ocho territorios repartidos por el Campus y cuatro equipos. Como es habitual, se efectuó su seguimiento con el fin de guiar a los participantes en caso que les surgiesen alguna duda. Lo único a destacar es que la partida no se pudo alojar en la misma máquina virtual ya que no se encontraba disponible, para solventar esta situación tuve que recurrir a mi ordenador personal para dar el soporte necesario.

Una vez finalizado el evento sin problemas aparentes, los jugadores retornaron y realizaron la encuesta, recogida en el anexo A.2. Antes de la despedida se ofreció una pequeña merienda a modo de gratificación, mientras todos comentábamos la experiencia vivida.

Resultados de la encuesta

Antes de empezar con el análisis, mencionar que el equipo formado por los dos participantes, realizó una sola encuesta consensuando sus respuestas. En general, las explicaciones previas al comienzo fueron valoradas muy favorablemente, aunque dos equipos las consideraron poco prácticas durante el juego. Este hecho se

puede explicar si se toman en consideración los menús de ayuda disponibles en la aplicación, ya que exceptuando un grupo, los demás los consultaron y les fueron útiles en algún momento de la partida.

El diseño de las diferentes pantallas ha sido catalogado por la mayoría como bueno, y el manejo de estas no ha sido una gran inconveniente. Sobre la funcionalidad de las notificaciones, ya sean de mensajes o enfrentamientos, dos equipos la valoraron como regular. Este aspecto puede ser mejorable añadiendo más información o algún tipo de sonido que alerte de su aparición.

Los dos cambios introducidos en la pestaña "Mapa", vista satélite e indicador de dirección en el icono del equipo, han sido muy bien acogidos. El objetivo que se buscaba era ayudar a los participantes a una orientación más rápida en el entorno y viendo los resultados, parece que se ha conseguido solventar este problema. Además, debido al consumo normal de la batería en la primera prueba, se optó por mejorar la precisión del GPS, este cambio fue valorado satisfactoriamente y seguro que reforzó positivamente la experiencia. En esta ocasión no se pudo obtener información sobre el impacto de las mejoras introducidas en el gasto de la batería, puesto que la mayoría de los equipos no se fijaron en este aspecto, lo que sí quedó demostrado fue el excelente rendimiento y fluidez de la aplicación.

Ningún equipo reportó errores o cierres inesperados de la aplicación, solo dos de ellos indicaron que durante el movimiento de tropas o ataque a un territorio tuvieron ciertas dificultades. Hablando con ellos durante el refrigerio, se llegó a la conclusión de que estos hechos no eran fallos sino que se trataban de avisos del control de concurrencia, informando que el recurso había sufrido modificaciones y no podía realizarse la acción.

Para los participantes, uno o dos integrantes por equipo es lo idóneo. Quizás grupos más numerosos pueden tener más dificultades en su organización y en la definición de una estrategia durante el juego. Los usuarios llegaron a divertirse bastante y mayoritariamente volverían a repetir la experiencia vivida. Este hecho se constató en las calificaciones obtenidas, que fueron de 7 a 10, y en la reunión distendida que tuvo lugar a la finalización de esta encuesta. En las últimas preguntas los participantes dejaron sus sugerencias, así como los aspectos positivos y negativos de la aplicación.

Al igual que el experimento anterior, se calculó el gasto de cada terminal y se obtuvo el mismo coste de 0.81 euros. Por último, debido a los resultados correctos y el alto grado de aceptación por parte de los voluntarios, quedó patente que no era necesaria una tercera prueba por lo que se acordó su supresión con el tutor.

Capítulo 5

Conclusiones del proyecto

En este capítulo se exponen los principales logros alcanzados, conclusiones y dificultades encontradas durante el desarrollo de este Proyecto Fin de Carrera. También se presenta la planificación final y se esbozan algunas de las posibles líneas de futuro con el fin de mejorar y hacer progresar el sistema implementado.

5.1. Conclusiones finales

En el capítulo 2 se marcaron una serie de objetivos para la elaboración del presente proyecto, a lo largo del cual se han ido cumpliendo satisfactoriamente. En primer lugar, se ha logrado implementar exitosamente cada componente, es decir, el servidor web y la aplicación Android. De igual manera se ha demostrado, mediante las pruebas realizadas, la gran madurez, estabilidad y aceptación del sistema construido. A continuación se listan las principales conclusiones extraídas del trabajo realizado.

- **Descubrir y asimilar las principales características de Android:** Inicialmente se tenía un gran desconocimiento de esta plataforma, pero durante el desarrollo de este PFC se ha conseguido asimilar sus características más relevantes. Comprender su arquitectura, componentes y funcionalidades han sido fundamentales para la correcta implementación de la aplicación. Este objetivo se ha podido cumplir gracias a la amplia y útil documentación expuesta por Google para los desarrolladores, siendo complementada con la consulta de manuales y libros.
- **Conocer nuevas tecnologías y conceptos:** Otro gran reto ha sido aprender las numerosas tecnologías y herramientas ignoradas hasta entonces.

Django ha sido de gran utilidad en el desarrollo del servidor web, ya que me ha permitido crear esta parte de una forma más rápida y organizada. La utilización de este *framework* implicó el aprendizaje del lenguaje de programación Python. También fue necesario manejar PostgreSQL, herramienta que facilitó la creación y gestión de la base de datos utilizada. A través del estudio de LibreGeoSocial se descubrieron nuevos conceptos como los de red social móvil, geolocalización y realidad aumentada. El manejo de LaTeX, utilizado en la redacción de este documento, me ha servido para conocer un lenguaje de procesamiento de texto potente y con unos resultados sobresalientes. En resumen, estos nuevos conceptos y tecnologías me han enriquecido y probablemente sean muy provechosos en mi futuro laboral.

- **Afianzar los conocimientos obtenidos:** La formación recibida durante estos años han dados sus frutos. El aprendizaje del lenguaje Java, en diferentes asignaturas de la carrera, me ha facilitado la compresión y programación en Android. Otro ejemplo son los conocimientos adquiridos en asignaturas relacionada con redes informáticas, útiles para la definición de la arquitectura y transmisión de datos entre los diversos componentes del sistema.

- **Importancia del software libre:** Las principales herramientas utilizadas cuentan con licencias que permiten su uso sin coste alguno. Además del beneficio económico, sus funcionalidades son iguales o superiores que la del cualquier software propietario. De esta manera, el proyecto se ha vertebrado gracias a este tipo de aplicaciones que han cumplido mis expectativas sobradamente.

- **Seguimiento de una metodología:** Apoyarse en una metodología de desarrollo software me ha ayudado en la organización y definición de las tareas llevadas a cabo en el proyecto. Aunque inicialmente me costó adaptarme, a la larga los beneficios obtenidos son mayores y esto se ve reflejado en los resultados conseguidos en las dos pruebas realizadas en un entorno real.

- **Cambio en la perspectiva de programación:** Lo más habitual, a lo largo de la carrera, era realizar programas destinados a ordenadores sin mayores restricciones. En este caso ha sido necesario un cambio en la mentalidad a la hora de desarrollar la aplicación, ya que el objetivo es ejecutarla en dispositivos con ciertas limitaciones hardware. Este hecho implica realizar una programación, si cabe, todavía más eficiente para adaptarla en un

ámbito donde el procesamiento de los datos, consumo de batería o tamaño de la pantalla son ajustados. Para cumplir este objetivo ha sido fundamental la API de Android, gracias a la cual permite llevar a cabo este tipo de implementación y aprovechar las últimas tecnologías que incorporan estos terminales móviles.

5.1.1. Dificultades encontradas

Aunque el proyecto se ha realizado sin grandes problemas, siempre aparecen contratiempos que dificultan su desarrollo. A continuación se mencionan algunas de estas dificultades.

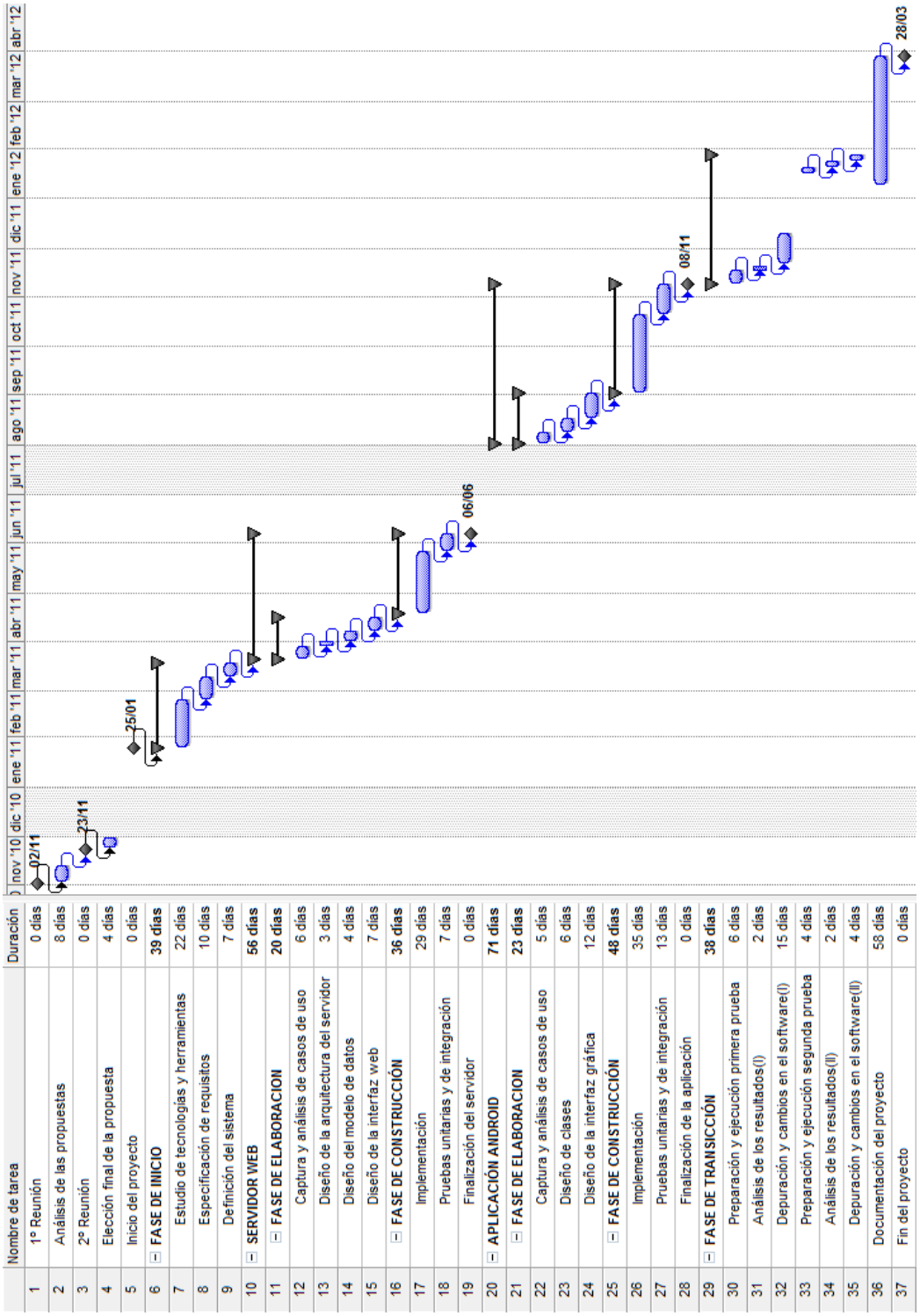
- Debido a que Android funciona en multitud de dispositivos con características diferentes, fue una tarea laboriosa adaptar los elementos gráficos a las resoluciones de cada terminal móvil. Para solucionar este problema, Google pone a disposición de los desarrolladores una guía con las prácticas correctas en el diseño de la GUI [40]. Como consecuencia de seguir estas indicaciones, se multiplicaron cada uno de los iconos e imágenes con el fin de proporcionar la mejor visualización en cualquier ámbito.

- Otra tarea en la que se dedicó un esfuerzo superior, fue el diseño de la hoja de estilo empleada en el servidor web. Lo más difícil fue acondicionar dicho CSS a la temática de la aplicación, es decir, seleccionar los colores e iconos más adecuados que aportasen un diseño atractivo y funcional para los usuarios. Al mismo tiempo, se buscaba la mayor compatibilidad del sitio web con los navegadores más importantes del mercado. El servicio de validación de CSS del W3C [41] resultó bastante práctico en esta tarea, ya que me ayudó a encontrar errores y posibles riesgos en la usabilidad.

- Uno de los cometidos más arduo fue la instalación y posterior configuración de todos los *framework* y aplicaciones utilizados en el proyecto. En este aspecto cobran una mayor relevancia los módulos (mod_wsgi y psycopg) encargados de comunicar las diferentes herramientas para su correcto funcionamiento. Este problema se acrecentó cuando se tuvo que repetir este proceso en varias ocasiones, una de ellas para configurar la máquina virtual destinada a la primera prueba, y debido a su inoperancia para la segunda prueba tuve que preparar mi ordenador personal.

5.2. Tiempo dedicado al proyecto

En la figura 5.1 se muestra el diagrama de Gantt con la planificación final del proyecto. Se pueden visualizar las fases y las principales tareas que se realizan, así como el tiempo empleado en cada una de estas.



	Nombre de tarea	Duración
1	1º Reunión	0 días
2	Análisis de las propuestas	8 días
3	2º Reunión	0 días
4	Elección final de la propuesta	4 días
5	Inicio del proyecto	0 días
6	**FASE DE INICIO**	**39 días**
7	Estudio de tecnologías y herramientas	22 días
8	Especificación de requisitos	10 días
9	Definición del sistema	7 días
10	**SERVIDOR WEB**	**56 días**
11	**FASE DE ELABORACION**	**20 días**
12	Captura y análisis de casos de uso	6 días
13	Diseño de la arquitectura del servidor	3 días
14	Diseño del modelo de datos	4 días
15	Diseño de la interfaz web	7 días
16	**FASE DE CONSTRUCCIÓN**	**36 días**
17	Implementación	29 días
18	Pruebas unitarias y de integración	7 días
19	Finalización del servidor	0 días
20	**APLICACIÓN ANDROID**	**71 días**
21	**FASE DE ELABORACION**	**23 días**
22	Captura y análisis de casos de uso	5 días
23	Diseño de clases	6 días
24	Diseño de la interfaz gráfica	12 días
25	**FASE DE CONSTRUCCIÓN**	**48 días**
26	Implementación	35 días
27	Pruebas unitarias y de integración	13 días
28	Finalización de la aplicación	0 días
29	**FASE DE TRANSICIÓN**	**38 días**
30	Preparación y ejecución primera prueba	6 días
31	Análisis de los resultados(I)	2 días
32	Depuración y cambios en el software(I)	15 días
33	Preparación y ejecución segunda prueba	4 días
34	Análisis de los resultados(II)	2 días
35	Depuración y cambios en el software(II)	4 días
36	Documentación del proyecto	58 días
37	Fin del proyecto	0 días

Figura 5.1: Diagrama de Gantt con la planificación del proyecto.

5.3. Trabajos futuros

A pesar de haber cumplidos todos los objetivos marcados de una forma bastante exitosa, hay líneas de trabajo que pueden seguirse para mejorar el proyecto. Algunas de estas propuestas son:

- Incrementar la variedad de tropas y que posean una serie de atributos (puntos de vida, ataque y defensa) que las hiciesen únicas. De esta manera se pondría a disposición del usuario unidades especializadas para la conquista o defensa de los territorios. Con esta idea se pretende involucrar aun más al jugador, y que decida que tropas son más idóneas según las circunstancias de la partida.

- Añadir un sistema de evolución en equipos, territorios y tropas. Según se vaya aumentado el nivel, los diferentes elementos adquirían ventajas en los futuros enfrentamientos. Una posible forma de llevar a cabo esta propuesta sería mediante la ganancia de experiencia en los combates, de tal forma que los recursos vencedores obtendrían una mayor recompensa de puntos. Cada incremento de nivel sería más exigente que el anterior.

- Materializar las alianzas que se producen entre los equipos para que la aplicación las reconozca y el resto de usuarios tengan constancia de su formación. Esta nueva funcionalidad permitiría a los miembros de una alianza compartir unidades entre ellos, realizar ataques conjuntos o ayudarse en la defensa de una zona. Este concepto imprimiría una mayor actividad y cambios inesperados en la dinámica del juego.

- Permitir el uso de varios terminales móviles por equipo, posibilitando la realización de acciones simultáneas en diferentes puntos geográficos. Cada usuario participaría con su dispositivo y conocería, en todo momento, las ubicaciones de sus compañeros.

- Distribuir la aplicación en Google Play, así se daría a conocer el juego y se obtendría la valoración de una gran comunidad de usuarios, opiniones muy valiosas para seguir mejorando el sistema. Para maximizar su éxito, sería recomendable añadir un modulo de registro en la página web, para permitir a los usuarios crear y seguir sus propias partidas. Con esto se consigue eliminar la dependencia que existe actualmente con el administrador del sitio web, ya que es el único que puede realizar estas tareas.

Bibliografía

[1] Jorge Fernández. Sitio web oficial LibreSoft Gymkhana. `http://gymkhana.libresoft.es/indice.html`. GSyC/LibreSoft. URJC. Último acceso: Marzo 2012.

[2] Sitio web oficial de Fable 3 Kingmaker. `http://www.fable3kingmaker.com/Index.aspx`. Último acceso: Marzo 2012.

[3] Sitio web oficial de Parallel Kingdom. `http://www.parallelkingdom.com/`. Último acceso: Marzo 2012.

[4] Ian Sommerville. *Ingeniería del Software*. Pearson Educación, 2005.

[5] Sitio web oficial de Python. `http://python.org`. Último acceso: Marzo 2012.

[6] Python wiki. Organizaciones que utilizan Python. `http://wiki.python.org/moin/OrganizationsUsingPython`. Último acceso: Marzo 2012.

[7] Sitio web oficial de Django. `https://www.djangoproject.com/`. Último acceso: Marzo 2012.

[8] Django. Documentación de GeoDjango. `https://docs.djangoproject.com/en/dev/ref/contrib/gis/`. Último acceso: Marzo 2012.

[9] Django. Documentación de Django - FAQ General. `https://docs.djangoproject.com/en/dev/faq/general/`. Último acceso: Marzo 2012.

[10] Sitio web oficial de Apache HTTP Server. `http://httpd.apache.org/`. Último acceso: Marzo 2012.

[11] Netcraft. January 2012 Web Server Survey. `http://news.netcraft.com/archives/category/web-server-survey/`. Último acceso: Enero 2012.

[12] Sitio web oficial de PostgreSQL. http://www.postgresql.org/. Último acceso: Marzo 2012.

[13] Marcos López Sanz. Gestión de la Información en Juegos y Realidad Virtual. http://www.kybele.etsii.urjc.es/MIGJRV/GIJRV/%5BGIJRV-2006-2007%5DTema5-Bases%20de%20Datos%20distribuidas.pdf. Kybele. URJC. Último acceso: Marzo 2012.

[14] PostgreSQL. Organizaciones que utilizan PostgreSQL. http://www.postgresql.org/about/users/. Último acceso: Marzo 2012.

[15] GSyC/LibreSoft. Sitio web oficial LibreGeoSocial. http://www.libregeosocial.org/. Último acceso: Marzo 2012.

[16] Morfeo Project. Software Libre y Realidad Aumentada en la feria de Albacete 2010. http://www.morfeo-project.org/archives/software%2Dlibre-y-realidad-aumentada-en-la-feria-de-albacete-2010. Último acceso: Marzo 2012.

[17] Android Developers. Sitio web oficial de Android para desarrolladores. http://developer.android.com/index.html. Último acceso: Marzo 2012.

[18] Ed Burnette. *Hello, Android: Introducing Google's Mobile Development Platform*. Pragmatic Bookshelf, 2010.

[19] Mark Murphy. *Beginning Android 2*. Apress, 2010.

[20] Sitio web oficial de Eclipse. http://www.eclipse.org/. Último acceso: Marzo 2012.

[21] Blog oficial de Google. Introducing Google Play: All your entertainment, anywhere you go. http://googleblog.blogspot.com/2012/03/introducing-google-play-all-your.html. Último acceso: Marzo 2012.

[22] Distimo. Google Android Market Tops 400,000 Applications. http://www.distimo.com/blog/2012_01_google-android-market%2Dtops-400000-applications/. Último acceso: Marzo 2012.

[23] Apple. Apple's Mac App Store Downloads Top 100 Million. http://www.apple.com/pr/library/2011/12/12Apples-Mac-App%2DStore-Downloads-Top-100-Million.html. Último acceso: Marzo 2012.

[24] Ivar Jacobson, Grady Booch, and James Rumbaugh. *El Proceso Unificado de Desarrollo de Software*. Addison Wesley, 2004.

[25] Esperanza Marcos, Belén Vela, and Juan M. Vara. *Diseño de Bases de Datos Objeto-Relacionales con UML*. Dykinson, 2005.

[26] Sitio web oficial de JSON. `http://www.json.org/json-es.html`. Último acceso: Marzo 2012.

[27] Sergio Saugar. REST: La Arquitectura de la World Wide Web. `http://zenon.etsii.urjc.es/grupo/docencia/as/material/tema6.pdf`. URJC. Último acceso: Marzo 2012.

[28] Android Developers. Plugin ADT para Eclipse. `http://developer.android.com/sdk/eclipse-adt.html`. Último acceso: Marzo 2012.

[29] Sitio web oficial de PostGIS. `http://postgis.refractions.net/`. Último acceso: Marzo 2012.

[30] Sitio web oficial del adaptador Psycopg. `http://initd.org/psycopg/`. Último acceso: Marzo 2012.

[31] Google Inc. Sitio web oficial del módulo mod_wsgi. `http://code.google.com/p/modwsgi/`. Último acceso: Marzo 2012.

[32] Concurrency control with django-locking. `http://stdbrouw.github.com/django-locking/`. Último acceso: Marzo 2012.

[33] World Wide Web Consortium (W3C). Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. `http://www.w3.org/TR/CSS2/`. Último acceso: Marzo 2012.

[34] Sitio web oficial de la librería JavaScript LiveValidation. `http://livevalidation.com/`. Último acceso: Marzo 2012.

[35] Google Developers. Versión 3 de Google Maps JavaScript API. `https://developers.google.com/maps/documentation/javascript/?hl=es`. Último acceso: Marzo 2012.

[36] Google Code. Sitio web para la obtención de un Maps API Key. `http://code.google.com/intl/es-ES/android/add-ons/google-apis/mapkey.html`. Último acceso: Marzo 2012.

[37] Android Developers. Obtaining User Location. http://developer.android.com/guide/topics/location/obtaining-user-location.html. Último acceso: Marzo 2012.

[38] HTC. Especificación técnica del HTC Desire. http://www.htc.com/la/product/htc-desire/specification2.html. Último acceso: Marzo 2012.

[39] HTC. Especificación técnica del HTC Magic. http://www.htc.com/la/product/htc-magic/specification.html. Último acceso: Marzo 2012.

[40] Android Developers. Icon Design Guidelines. http://developer.android.com/guide/practices/ui_guidelines/icon_design.html. Último acceso: Marzo 2012.

[41] World Wide Web Consortium (W3C). Servicio de validación de CSS. http://jigsaw.w3.org/css-validator/. Último acceso: Marzo 2012.

Apéndice A

Resultados de las Encuestas

Seguidamente se muestran las encuestas realizadas por los participantes en las diferentes pruebas, con sus respectivos resultados.

A.1. Encuesta de la primera prueba

Respuestas enviadas: 4
Preguntas: 35

1. Valore las explicaciones sobre el manejo de la aplicación en el terminal móvil antes de empezar a jugar.

- Insuficientes: 0
- Regulares: 0
- Buenas: 0
- Muy buenas: 4 (100.00 %)

2. ¿Cómo le ayudaron las explicaciones durante el transcurso de la partida?

- Poco: 0
- Suficiente: 0
- Bastante: 2 (50.00 %)
- Mucho: 2 (50.00 %)

3. ¿Cómo de sencillo le resultoó el manejo del terminal móvil?

- Poco: 0
- Normal: 0
- Bastante: 1 (25.00 %)
- Mucho: 3 (75.00 %)

4. ¿Qué le ha parecido la información o ayuda que se puede consultar a lo largo de las diferentes ventanas de la aplicación?

- No la he consultado: 1 (25.00 %)
- Necesaria pero poco útil: 0
- Necesaria y suficiente: 3 (75.00 %)
- Excesiva: 0

5. Las barras de progresos se ha incluido para evitar la sensación de cuelgue de la aplicación al realizar una determinada acción. ¿Le han resultado útiles?

- Poco útiles: 0
- Útiles: 3 (75.00 %)
- Muy útiles: 1 (25.00 %)

6. Valore la GUI (interfaz gráfica) de la aplicación en cuanto al manejo y sencillez.

- Mala: 0
- Normal: 0
- Buena: 1 (25.00 %)
- Muy buena: 3 (75.00 %)

7. ¿Qué le parece, en general, el diseño de las diferentes pantallas de la GUI?

- Poco original: 0
- Normal: 1 (25.00 %)
- Original: 3 (75.00 %)
- Muy original: 0

8. ¿Qué le parece el formato, tamaño de letra y color del texto presentado en cada GUI?

- Malo: 0
- Normal: 0
- Bueno: 3 (75.00 %)
- Muy bueno: 1 (25.00 %)

9. Valore, en función de su contenido y estética, cómo se muestran las diferentes notificaciones o avisos del juego a lo largo de la partida.

- Poco correcto: 0
- Normal: 0
- Correcto: 4 (100.00 %)
- Muy correcto: 0

10. ¿Le resulta útil la información que se muestra en la pestaña "Estadísticas"?

- Sí: 3 (75.00 %)
- No: 1 (25.00 %)

11. En caso de haber contestado negativamente la pregunta anterior, ¿qué cambiaría o que información cree que falta?

-
-
- Creo que le quedaría bien que apareciesen reflejados también los duelos ganados, pero el resto está muy bien.
-

12. La vista utilizada en la pestaña "Mapa", ¿la cambiaría por la vista satélite?

- No conozco la vista satélite de los mapas de Google: 0
- No, así está correcto.: 1 (25.00 %)
- Si, la cambiaría por la vista satélite: 3 (75.00 %)

13. En caso de haber contestado afirmativamente la pregunta anterior, ¿cuáles son los motivos?

-
-
-
-

14. Valore la precisión del GPS durante el juego.

- Mala: 0
- Normal: 1 (25.00 %)
- Buena: 1 (25.00 %)
- Muy buena: 2 (50.00 %)

15. Valore si le parece original la decisión de implementar las acciones como menús emergentes pulsando sobre el recurso en el mapa, en vez de pulsar sobre el botón físico "menú" del terminal.

- Poco original: 0
- Normal: 0
- Original: 2 (50.00 %)
- Muy original: 2 (50.00 %)

16. El radio de acción de cada equipo para interactuar con los diferentes recursos del mapa está fijado en 75 metros, ¿modificaría esta longitud?

- Sí: 0
- No: 4 (100.00 %)

17. En caso de haber contestado afirmativamente la pregunta anterior, indique el radio que considere mejor para jugar y el motivo del cambio.

-
- Considero que, según la distancia del mapa total actual, está bien en 75 metros.
-
-

18. Valore el diseño e información mostrada en la GUI utilizada para la conquista de un territorio rival.

- Malo: 0
- Regular: 0
- Bueno: 4 (100.00 %)
- Muy bueno: 0

19. Valore el diseño e información mostrada en la GUI utilizada para el movimiento de tropas entre los territorios y/o equipo.

- Malo: 1 (25.00 %)
- Regular: 0
- Bueno: 2 (50.00 %)
- Muy bueno: 1 (25.00 %)

20. ¿Le resulta útil que la aplicación siempre le informe cuando su equipo o algún territorio de su propiedad ha sido atacado?

- No, lo veo innecesario: 0
- Sí, pero solo cuando he perdido el territorio: 0
- Sí, así es más completo y se obtiene más información.: 4 (100.00 %)

21. Por cada conquista exitosa se dan 25 puntos, por cada territorio en posesión 10 puntos y por cada duelo ganado 15 puntos. ¿Le parece apropiado este sistema de puntuación?

- Sí: 3 (75.00 %)
- No: 1 (25.00 %)

22. En caso de haber contestado negativamente la pregunta anterior, ¿Por qué modificaría el sistema de puntuación? ¿Qué nuevos valores fijaría?

- Deberia haber puntuaciones por defender con exito territorios. Por ejemplo 5 puntos o algo similar.
-
-
-

23. Cada 3 minutos, se incrementan las tropas de los equipos y de cada territorio con un propietario. ¿Qué le parece este periodo de tiempo?

- Lo reduciría ligeramente: 1 (25.00 %)
- Bien, lo dejaría como está: 2 (50.00 %)
- Lo incrementaría ligeramente: 1 (25.00 %)

24. Valore el uso del servicio de mensajería en el envío de mensaje a otros participantes o al organizador.

- No lo usé: 0
- Complejo: 0
- Normal: 1 (25.00 %)
- Fácil: 3 (75.00 %)

25. Sobre el servicio de mensajería, ¿le parece correcto el aviso generado en la barra de notificaciones cuando el sistema detecta un nuevo mensaje recibido?

- No es necesario este tipo de aviso: 0
- Sí, pero le añadiría algún tipo de sonido para completarlo: 1 (25.00 %)
- Si, así está bien: 3 (75.00 %)

26. Durante el transcurso de la partida, ¿se ha producido algún error o cierre inesperado de la aplicación?

- Sí: 2 (50.00 %)
- No: 2 (50.00 %)

27. En caso de haber contestado afirmativamente la pregunta anterior, indique bajo que circunstancia se ha producido el error.

-
- Al comenzar la aplicación el mapa no me ha aparecido en ningún momento y el terminal se ha quedado bloqueado. He seguido el curso del juego con el moderador y de ahí mis impresiones de la encuesta.
- Lo único que me ha pasado es que en algún momento el tiempo restante se me ha reiniciado, lo demás ha funcionado perfectamente.
-

28. Valore el consumo de la batería a lo largo de la partida.

- No me he fijado en este detalle: 1 (25.00 %)
- Bajo: 0
- Normal, como cualquier otro juego en un terminal móvil: 3 (75.00 %)
- Alto: 0

29. Valore, de forma general, el rendimiento de la aplicación en su terminal móvil.

- Malo: 1 (25.00 %)
- Regular: 0
- Bueno: 2 (50.00 %)
- Muy bueno: 1 (25.00 %)

30. Indique si le parece divertido el juego.

- Me aburrí: 0
- Normal: 0
- Bastante: 3 (75.00 %)
- Mucho, repetiría la experiencia: 1 (25.00 %)

31. En términos generales, ¿qué le parece la aplicación?

- Malo: 0
- Regular: 0
- Buena: 2 (50.00 %)
- Muy buena: 2 (50.00 %)

32. ¿Está familiarizado con el sistema operativo (SO) Android?

- Nada familiarizado: 1 (25.00 %)
- Algo he leído, pero no lo he utilizado: 1 (25.00 %)
- Soy usuario de un móvil y/o tablet con este SO: 2 (50.00 %)
- Desarrollo aplicaciones en este SO: 0

33. Indique los aspectos más positivos del juego.

- Originalidad, y el hecho de que sea geolocalizado, es decir que "uno mismo" conquista los territorios.
- MOVILIDAD en contraste con la mayoría de juegos actuales; lo cual unido a algún tipo de herramienta para inserción de mapas y puntos podría hacerla portable para cualquier lugar; por ejemplo, el campo, o algún sitio de la ciudad concreto.
- Me gustaría destacar la facilidad de manejo y lo intuitivo que resultan todas las acciones a realizar a lo largo del juego, enseguida te haces con el juego y disfrutas con el. También creo necesario recalcar la fluidez con la que funciona todo, todo funciona lo rápido que debe funcionar. Además es muy fácil acceder a las distintas acciones, visión de estadisticas y mensajería.
- Una buena aplicación para jugar al aire libre y conocer a gente. En general, na interfaz intuitiva y fácil de manejar.

34. Si los hubiese, ¿qué aspectos añadiría, modificaría o eliminaría del juego? ¿Por qué?

- Añadiria:
  - Tipos de tropas distintos, para una mayor profundidad. Ej: Soldado, Caballeria, Arqueros, etc.
  - Que se incrementaran los espias tambien cada cierto tiempo. Por ejemplo el doble que las tropas para no abusar de ellos.

  Mejoras:
  - En la pantalla de estadisticas, cuando hay varios territorios conquistados y varios jugadores, no se visualiza bien la clasificación.
- Como he expresado hace un momento, añadiría esa herramienta de inserción de mapas y puntos aunque soy consciente de la gran complejidad que implicaría ésta. Es decir, convertir al propio usuario en creador de su propio mundo.
- Creo que algo más de prfoundidad a la hora de gestionar las zonas conquistadas, como subir el nivel de las mismas, de modo que sea más difícil conquistarlas, y quizás también algun tipo de unidad adicional, que haya unidades más defensivas y otras más ofensivas.
- Tiene bastante variedad de acciones pero añadiría alguna variedad más de tropas. También añadiría una especie de brújula para mejorar la orientación mientras se juega.

35. En último lugar, deje cualquier comentario que considere oportuno o plantear alguna sugerencia que no se haya recogido en esta encuesta.

-
- Me ha parecido muy correcta y creo que a partir de ahí se pueden añadir todas las complejidades y modificaciones que se quieran (distintos tipos de tropas, diferenciación de los territorios,...) pero lo básico y más importante ya estaría hecho.
- La experiencia ha sido buena y la aplicación me ha parecido muy divertida, muy recomendable.
- Una aplicación divertida de manejar, a pesar de los fallos que ha tenido mis compañeros, ha sido una buena experiencia.

A.2. Encuesta de la segunda prueba

Respuestas enviadas: 4
Preguntas: 24

1. Valore las explicaciones dadas sobre la aplicación antes de iniciar la partida.
- Insuficientes: 0
- Regulares: 0
- Buenas: 3 (75.00 %)
- Muy buenas: 1 (25.00 %)

2. Indique en qué grado le han sido útiles las explicaciones previas durante el juego.
- Nada: 0
- Poco: 2 (50.00 %)
- Bastante: 0
- Mucho: 2 (50.00 %)

3. ¿Qué le ha parecido el manejo de las diferentes pantallas en la aplicación?
- Difícil: 0
- Normal: 2 (50.00 %)
- Fácil: 1 (25.00 %)
- Muy fácil: 1 (25.00 %)

4. Valore el diseño de los diferentes elementos de la GUI (menús de información, iconos, menús emergentes, exposición de la información, etc.).
- Mala: 0
- Normal: 1 (25.00 %)
- Buena: 2 (50.00 %)
- Muy buena: 1 (25.00 %)

5. ¿Le ha resultado de alguna utilidad los diferentes menús de información y ayuda presentes en la aplicación?
- No la he consultado: 1 (25.00 %)
- No, es insuficiente: 0
- Si, es suficiente: 2 (50.00 %)
- Si, bastante completa y útil: 1 (25.00 %)

6. ¿Le ha parecido conveniente la información proporcionada por la pestaña "Estadísticas"?
- Sí: 3 (75.00 %)
- No: 1 (25.00 %)

7. En caso de haber contestado negativamente la pregunta anterior, ¿Qué tipo de información ha echado en falta?
-
-
-
- Se necesita, alguna instrucción de como obtener tropas

8. El indicador de dirección junto a la vista satélite del mapa, son cambios introducidos para ayudar al equipo en la orientación. ¿Le han resultado beneficiosos estos cambios?
- Poco: 0
- Normal: 0
- Bastante: 0
- Mucho: 4 (100.00 %)

9. Valore la precisión del GPS para establecer su posición a lo largo de la partida.
- Mala: 0
- Regular: 0
- Buena: 2 (50.00 %)
- Muy buena: 2 (50.00 %)

10. Por cada conquista exitosa se dan 25 puntos, por cada territorio en posesión 10 puntos y por cada duelo ganado 15 puntos. ¿Le parece apropiado este sistema de puntuación?

- Sí: 4 (100.00 %)
- No: 0

11. En caso de haber contestado negativamente la pregunta anterior, ¿Por qué modificaría el sistema de puntuación? ¿Qué nuevos valores fijaría?

-
-
-
-

12. Valore la frecuencia con la que se incrementan las tropas del equipo y de los territorios.

- Lo incrementaría: 1 (25.00 %)
- Correcta: 2 (50.00 %)
- Lo reduciría: 1 (25.00 %)

13. Valore la utilidad del servicio interno de mensajería en la aplicación.

- No lo usé: 1 (25.00 %)
- Poco: 1 (25.00 %)
- Bastante: 2 (50.00 %)
- Mucho: 0

14. En términos generales, ¿qué le parece la estética y la funcionalidad de las diferentes notificaciones (enfrentamientos y mensajería)?

- Mala: 0
- Regular: 2 (50.00 %)
- Buena: 1 (25.00 %)
- Muy buena: 1 (25.00 %)

15. Según su experiencia en el juego con otros compañeros, ¿cuál sería el número ideal de integrantes para formar un grupo?

- 1: 2 (50.00 %)
- 2: 2 (50.00 %)
- 3: 0
- 4 o más: 0

16. Indique si se ha divertido y, en tal caso, ¿repetiría la experiencia?

- No: 0
- Normal, pero no volvería a jugar: 0
- Sí, probablemente repetiría: 2 (50.00 %)
- Mucho, seguro que volvería a jugar: 2 (50.00 %)

17. Durante el transcurso de la partida, ¿se ha producido algún error o cierre inesperado de la aplicación?

- Sí: 2 (50.00 %)
- No: 2 (50.00 %)

18. En caso de haber contestado afirmativamente. Explique brevemente y bajo qué circunstancias se ha producido el error.

-
-
- Daba error a veces, cuando intentabas mover tropas
- error de se ha interrumpido el movimiento de tropas por motivos.

19. ¿Qué le ha parecido el consumo de batería durante el tiempo que ha jugado?

- No me he fijado: 3 (75.00 %)
- Bajo: 1 (25.00 %)
- Normal: 0
- Alto: 0

20. Valore el rendimiento y fluidez de la aplicación en su teléfono.

- Malo: 0
- Normal: 1 (25.00 %)
- Bueno: 2 (50.00 %)
- Muy bueno: 1 (25.00 %)

21. Teniendo en cuenta todo lo anterior, califique de forma general la aplicación.

- 1: 0
- 2: 0
- 3: 0
- 4: 0
- 5: 0
- 6: 0
- 7: 1 (25.00 %)
- 8: 1 (25.00 %)
- 9: 1 (25.00 %)
- 10: 1 (25.00 %)

22. Indique los aspectos más positivos del juego.

- La correcta posicion en el mapa y la distancia de conquista.
- Es fácil de usar y muy adictivo.
- Es muy dinámico, muy original por la geolocalizacion
- localización gps buena
  iniciativa buena

23. Indique algún aspecto negativo del sistema.

- El sistema de anuncios de batallas o territorios en disputa. Si le das varias veces a un territorio, te sales varias veces las ventanas de opciones. Mejor que se limitara a 1.
- Ninguno!
- errores cuando movian las tropas y retardos en la localizacion
- necesita más información, cuando una tropa le aproxima a atacar, no supe cuando alguien me va atacaba. No entredí muy bien lo de espías, no sabía como usarlos

24. Por último, deje cualquier comentario que considere oportuno o plantee alguna sugerencia que no se haya recogido en la encuesta.

- El numero de tropas creo que deberia estar presente al lado de mi posicion y de mis territorios. Seria mucho meor que te indicara si alguien te ha batido en duelo y el resultado de este, asi como informarme de los territorios que me ha conquistado o que lo han intentado.
- Daría mayor valor en la puntuación o mayor número de producción de tropas a los territorios alejados (como el de la zona de las canchas de deporte) para compensar la caminata.
- Los mensajes deberian emerger en la pantalla sin tener que ir a mirarlos.

  Creo que deberian informarte de que se han batido en duelo contigo

  El numero de tropas de tu equipo deberia estar siempre visible en el mapa y debajo de cada icono de territorio propio, el numero de tropas que tiene

  Que se notificara los territorios que te arrebatan

  La opcion de tener varios moviles por equipo
- tal vez un menú de instrucciones previas o ayuda en el juego