Universidad Rey Juan Carlos

Escuela Técnica Superior de Ingeniería Informática

Ingeniería Informática

Proyecto Fin de Carrera

Desarrollo de una Aplicación con Geolocalización para el Sistema Operativo Móvil Android: GoldenStreets

Autor: Rubén Quero Castrillo

Tutor: Gregorio Robles Martínez

Curso académico: 2011/2012

Agradecimientos

En primer lugar agradecer a mis padres la plena confianza que pusieron en mí para emprender esta carrera de ingeniería, así como a mis hermanos y familiares.

También quiero dar las gracias a todas las personas (profesores, amigos, etc.) que me han apoyado tanto directamente como indirectamente en el transcurso de esta etapa de mi vida.

Especialmente agradecer a mi tutor Gregorio Robles Martínez quien me ha ayudado y proporcionado consejos en todo en momento.

Resumen

En la actualidad la gran mayoría de los teléfonos smartphones disponen de importantes características, asemejándose a un ordenador, e incluyendo otras funcionalidades (GPS, cámara, conectividad, etc.). Esto junto a la irrupción del sistema operativo Android en el mundo de los móviles, favorecen el desarrollo de aplicaciones para adaptar los juegos clásicos, como pueden ser Monopoly y Pac-Man, entre otros.

Este Proyecto Fin de Carrera tiene como principal finalidad, la implementación de una plataforma software basada en la localización del usuario, gracias a la red social móvil LibreGeoSocial, siguiendo la metodología del Proceso Unificado de Desarrollo. Con este sistema se pretende ofrecer a los diferentes participantes, en grandes rasgos, una nueva forma y experiencia a la hora de jugar, explotando los diferentes recursos que ofrecen los terminales móviles y el sistema operativo Android.

Para su realización se ha diseñado una arquitectura Cliente-Servidor, con la idea principal del proceso encargado de proporcionar los diferentes servicios, como el almacenamiento de la estructura de datos, sea el Servidor. Además este pondrá a disposición del organizador un servicio web, desarrollado con Django, con el objetivo de facilitar la gestión y seguimiento de la partida. Accesible por el Cliente mediante los principales navegadores webs. El otro Cliente está enfocado a los jugadores que podrán participar mediante una aplicación destinada a los móviles Android, con la cual podrán realizar numerosas acciones. Además de servir su desarrollo para adquirir y ampliar los conocimientos sobre las características del sistema operativo creado por Google. Ambos clientes se comunicarán mediante el protocolo HTTP enviando peticiones al servicio web e intercambiando los datos en formato JSON.

Finalmente se verifica el correcto funcionamiento del sistema implementado, comprobado con la realización de diferentes pruebas en un escenario real. Con estas se ha conseguido descubrir y corregir los errores detectados, así como obtener la importante opinión de los numerosos voluntarios que participaron y que han sido muy útiles para la maduración y progreso del sistema desarrollado en este Proyecto Fin de Carrera.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Este capítulo pretende ser el punto de entrada para comprender el resto del documento. Ofreciendo las motivaciones que han llevado a realizar el presente PFC, así como una breve descripción del resto de contenidos.

1.1. Motivación

En la actualidad el sistema operativo Android para móviles y tablets, está creciendo a un ritmo imparable, convirtiéndose en la plataforma más utilizada por estos dispositivos. También, dispone de una tienda de software en línea, accedida mediante el terminal móvil, denominada Google Play [11], poniendo a disposición de los usuarios multitud de aplicaciones de diferentes categorías, tanto gratuitas como de pago.

La proliferación y evolución de las nuevas tecnologías, servicios web, conectividad y que los terminales móviles presentan cada vez mayores prestaciones y más funciones para el beneficio del usuario. Abre un nuevo camino para las aplicaciones Android que se desarrollan en estos dispositivos, permitiendo aprovechar de manera eficiente sus características, ofreciendo una nueva experiencia al usuario. El desarrollador puede crear un entorno más allá del ordenador y consola, siendo el mundo real el escenario de la aplicación o juego. Además la utilización de los terminales móviles evita la utilización de recursos materiales para la realización de las funciones requeridas, lo que conlleva un cierto ahorro económico.

En conclusión, las motivaciones para la realización de este Proyecto Fin de Carrera se centran en poner en práctica gran parte del conocimiento adquirido a lo largo de

la carrera y en la completa realización de una aplicación basada en la red social LibreGeoSocial para ser accedida mediante la web y terminales Android. Tomando como un reto personal ampliar el conocimiento y aprender nuevas herramientas de desarrollo no vistas en el transcurso de la carrera, como el *framework* Django o LibreGeoSocial. Además de desarrollar con éxito un complejo sistema que cumpla con las necesidades de los participantes.

1.2. Contenido de la memoria

Este apartado ofrece una breve descripción del contenido que se puede encontrar en cada uno de los capítulos que componen la memoria.

En la **Introducción** se habla sobre los motivos que han impulsado a la realización del presente proyecto, así como una breve descripción del contenido de cada uno de los capítulos y se comentan algunas aplicaciones similares a la desarrollada.

En el segundo capítulo, **Estado del arte**, se expondrán los aspectos más relevantes de las principales herramientas y tecnologías que se han llegado a utilizar en el Proyecto Fin de Carrera, además de un breve repaso a la historia de cada una de estas.

En los **Objetivos** se dará una visión global del proyecto, que objetivos se han establecido, así como la metodología seguida para su elaboración.

Durante el cuarto capítulo, **Diseño**, se analizará y definirá el sistema propuesto para la realización del este Proyecto Fin de Carrera, detallando desde el punto de vista del diseño cada uno de los componentes que lo conforman.

El capítulo **Implementación**, se explica cómo se han realizado finalmente los componentes diseñados en el capítulo anterior, además de las distintas funcionalidades que proporciona, a sus respectivos clientes, cada uno de estos.

En el capítulo **Pruebas y resultados**, se expone la experiencia y resultados obtenidos de las diferentes pruebas realizadas en un escenario real con un grupo de voluntarios, pudiendo verificar el correcto funcionamiento, identificar los carencias y comprobar la aceptación del usuario.

En el capítulo a modo de cierre de la memoria, **Conclusiones**, se da a conocer las conclusiones finales alcanzadas, así como las posibles líneas de futuro que posibilitan evolucionar y mejorar el sistema propuesto. También se exponen el tiempo y esfuerzo empleado al desarrollar cada una de las fases del proyecto.

En la **Bibliografía** se presentan las referencias bibliográficas consultadas durante la realización del Proyecto Fin de Carrera.

Al final del documento se localizan los **Anexos**, donde se recogen los resultados obtenidos de las encuestas realizadas a los participantes en las pruebas del sistema.

1.3. Otros *location-based games*

El presente proyecto se engloba en la categoría de *location-based games*, entre otros nombres conocidos, siendo este nuevo género que está emergiendo poco a poco debido a las posibilidades que ofrecen los terminales móviles. A continuación, se exponen dos ejemplos que realizan un correcto uso de las prestaciones que ofrecen los smartphones.

1.3.1. LibreSoft Gymkhana

El proyecto LibreSoft Gymkhana [18] es un software libre y de código abierto desarrollado por el GSyC (Grupo de Sistemas Telemáticos y Computación) y con el apoyo de la red eMadrid. Se trata de un juego geolocalizado y educativo basado en la red social LibreGeoSocial, también utilizada en el presente proyecto, debido a sus innumerables usos está relacionada con las aplicaciones m-learning y turismo. Esta aplicación proporciona el soporte necesario al Organizador para crear fácilmente su propia gymkhana, a través de una secuencia de retos interrelacionados con la temática que el haya elegido para esta. Siendo el objetivo de cada participante, mediante un terminal móvil, superar satisfactoriamente el mayor número de pruebas propuestas (ver figura 1.1).

El Organizador tendrá a su disposición un sitio web para la creación de la gymkhana, accesible desde cualquier ordenador o dispositivo a través de Internet. Podrá crear desde cero una nueva indicando sus datos, entre estos el lugar o fecha de celebración. Además se deberá indicar o crear los grupos participantes formados por los usuarios de la red social LibreGeoSocial, así como los tipos y ubicación de las pruebas que deberán de superar en el transcurso de esta. En este mismo sitio web se pone a disposición del Organizador un servicio de mensajería para la comunicación con los participantes, ya sea para solventar dudas o aportar información adicional a un reto, pero también una página con la monitorización de la partida. En el seguimiento de la partida el usuario podrá ver en tiempo real la tabla de clasificación, la geolocalización de los participantes y las respuestas que han realizado de los diferentes retos propuestos.

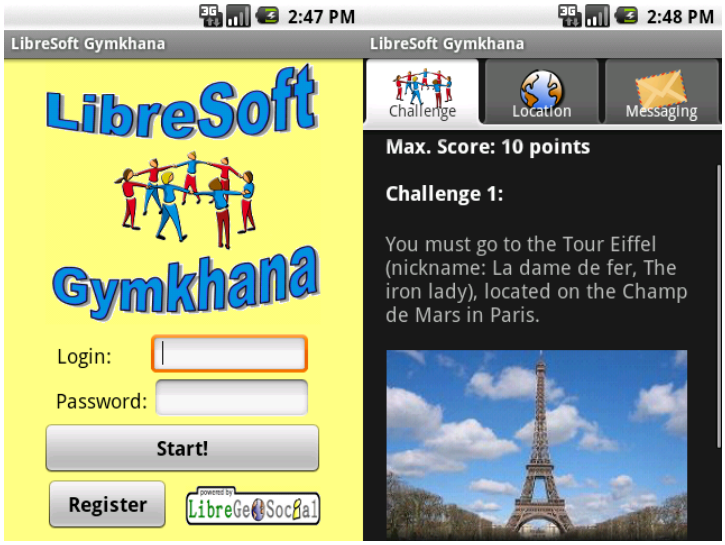


Figura 1.1: Capturas de la aplicación Libresoft Gymkhana

Un jugador puede encontrarse diferentes tipos de retos en el transcurso de una gymkhana a través de su dispositivo, cada uno de estos exigiéndole la máxima atención y esfuerzo. Las respuestas enviadas por los participantes deberán de ser verificadas por el Organizador.

- **Reto de respuesta textual:** En este tipo de prueba, el equipo tiene que resolver el reto planteado mediante una respuesta textual.
- **Reto de respuesta fotográfica:** Se deberá responder mediante una fotografía realizada a través del terminal al reto propuesto.
- **Reto de geolocalización:** Otro tipo de reto, estará enfocado a la ubicación del usuario teniéndose que dirigir a un determinado lugar, sin necesidad de enviar ninguna respuesta.
- **Reto de realidad aumentada:** Similar al textual, pero en este caso se deberá obtener la respuesta mediante el módulo de realidad aumentada que implementa la aplicación.

Finalmente indicar que con esta aplicación se consigue facilitar la organización de las gymkhanas y mejorar la experiencia vivida por los participantes, teniendo un mayor control sobro los equipos y retos.

1.3.2. BigShot

BigShot [5] es un juego MMORPG (massively multiplayer online role-playing game) basado en la localización del usuario, siendo el mundo real el escenario. El principal objetivo es adentrarse en un nuevo mundo, en el cual el jugador debe de convertirse en el jefe de la gran ciudad, para esto se podrán ocupar lugares de importancia estratégica alrededor de su posición, defendiéndolos o capturando otros a través de combates con sus propietarios.

A través de la aplicación se ofrece al usuario diferentes acciones que podrá realizar con su personaje, para aumentar sus posibilidades de éxito se le equipará con armas, ropa, etc. Según se vaya desplazando el jugador por las calles de su ciudad, mediante el mapa habilitado en la aplicación móvil podrá ir capturando diferentes tipos de edificios, como pueden ser restaurantes, bancos, joyerías y escuelas, entre otros. Al visitar cada una de estas ubicaciones se podrán recibir recompensas para incrementar el nivel de tu personaje o equiparlo, también se tendrá la opción de capturar el sitio luchando con su propietario, así como combatir con otros personajes que han visitado esa ubicación (ver figura 1.2).

Figura 1.2: Capturas de la aplicación BigShot

El usuario puede aceptar misiones para ganar experiencia y dinero, estas se deberán de realizar en un determinado periodo de tiempo. Las misiones ofrecidas pueden ser de diferentes tipos (visitar, capturar o combatir), en estas se especifican el lugar donde se deben de desarrollar, aunque también aparecerán marcadas en el mapa. Periódicamente

se podrá recolectar la renta de los edificios que posea el jugador, en el caso que no recogiese sus beneficios en unos determinados días perderá la propiedad.

El juego pone a disposición del usuario una tienda en la que podrá comprar, mediante el dinero recolectado y oro, diferentes armas para potenciar su ataque o ropa para incrementar la armadura. Cada personaje dispone de vida y energía que se irán agotando según avance en el juego, posibilitando su recuperación con la adquisición de diferentes objetos, como botiquines.

Capítulo 2

Estado del arte

El propósito de este capítulo es situar el Proyecto Fin de Carrera dentro del ámbito de tecnologías y herramientas utilizadas, proporcionando al lector las nociones básicas para la correcta comprensión del mismo.

2.1. Arquitectura Cliente-Servidor

En el mundo TCP/IP las comunicaciones entre los diferentes procesos se rigen principalmente por lo que se conoce como modelo arquitectónico Cliente-Servidor [32]. Desde el punto de vista funcional, este modelo se puede definir como una arquitectura distribuida, permitiendo a los distintos usuarios obtener acceso a la información requerida de forma transparente.

La idea de este modelo arquitectónico es que un proceso actúa como Servidor, encargado de proporcionar diferentes servicios. Otro proceso será el encargado de iniciar la comunicación, enviando un mensaje (petición) a través de la red demandando algún servicio o información, este será el Cliente. Cuando el servidor reciba dicha petición, la procesa y envía el resultado obtenido. Normalmente, el servidor y cliente, están localizados en ordenadores distintos distribuyendo físicamente los procesos y los datos de forma más eficiente. Aunque también ambos procesos pueden situarse en la misma máquina.

Se pueden distinguir diferentes tipos de clientes dependiendo de la implementación que se realice en ellos. El Cliente flaco, no implementa la lógica de la aplicación, actúa únicamente de intermediario entre el usuario y el servidor. Requiere muy pocos recursos hardware, pero ocasiona un gran tráfico de datos por la red y el servidor puede saturarse

rápidamente. En cambio el Cliente gordo, implementa la mayor parte de la lógica de la aplicación encargándose de procesar la información del usuario antes de enviarla. Requiere una mayor capacidad de proceso y capacidad de almacenamiento, reduciendo gran parte del trabajo al servidor.

Si optamos por una arquitectura basada en un cliente flaco el servidor será más complicado, en el caso que se opte por uno gordo el servidor será más sencillo teniendo una mayor complejidad el cliente.

El servidor utilizado en el sistema puede realizar diferentes roles, dependiendo del que realiza podemos distinguir: Servidor dedicado, es aquel que se encarga únicamente de atender las peticiones realizadas por los clientes. Servidor no dedicado, aquel que no dedica todo sus recursos a sus clientes, sino que también actúa como estación de trabajo procesando las peticiones del usuario local.

Entre las principales ventajas que proporciona el modelo Cliente-Servidor están:

- El esquema facilita la integración de nuevos procesos, posibilitando un crecimiento gradual, aumentando el número de clientes. Sustituir los servidores sin que los demás recursos se vean afectados o en algunos casos añadir nuevos sin necesidad de rediseñar la arquitectura.
- No es necesario transmitir información gráfica por la red al servidor, solo los datos necesarios, pues esta reside principalmente en el cliente.
- Utilización de componentes (clientes y servidores), tanto de hardware como de software diferentes, lo cual contribuye a la reducción de costos, favorece a la flexibilidad y actualización de soluciones.

Entre las principales desventajas que proporciona el modelo Cliente-Servidor están:

- El mantenimiento de los sistemas es más difícil, implica la interacción de diferentes partes de hardware y software, distribuidos en diferentes computadores.
- Los sistemas distribuidos tienden a fallar con mayor frecuencia que un sistema centralizado, ya que posee más componentes que pueden estropearse independientemente, complicando la depuración de errores.
- El servidor en sistemas distribuidos puede ser un cuello de botella presentándose congestión en la red, es decir, dificultad en el tráfico de datos.

2.2. Python

El lenguaje de programación Python [29] [30] fue creado a finales de 1989 como sucesor del lenguaje ABC por Guido van Rossum [13], debido a su experiencia en CWI (*Centrum Wiskunde & Informatica*) y motivado por la necesidad de un lenguaje de alto nivel en el proyecto Amoeba. También existía la necesidad de un nuevo lenguaje que "una la brecha entre C y el shell", por mucho tiempo este fue el principal eslogan del lenguaje de programación Python. El nombre del lenguaje viene de un programa de la BBC *Monty Python's Flying Circus*, uno de los grupos cómicos preferidos por el creador. Finalmente en 1991 el lenguaje de programación Python fue publicado bajo una licencia de código abierto (*source code*), antes de que el término fuese creado a finales de 1997.

Actualmente Python está en continuo movimiento y pleno desarrollo, siendo una interesante opción para aprender y desarrollar cualquier tipo de programa. También dispone de una amplia, clara y sencilla documentación para familiarizarse con el lenguaje. Algunas empresas que lo utilizan son *Yahoo, Google, NASA, Walt Dysney, Civilization 4*, etc. Python es un lenguaje scripting, interpretado, interactivo y orientado a objetos que ofrece una gran variedad de estructuras de datos de alto nivel. Cuya filosofía hace hincapié en una sintaxis limpia y que favorezca un código legible. A continuación se exponen y explican algunas de sus principales características:

Propósito general: Puede ser usado para varios propósitos, como puede ser, acceso a bases de datos, cálculos matemáticos, desarrollo de páginas web, etc.

Dinámico: No es necesario declarar el tipo de dato que va a contener una determinada variable, dicho tipo de dato será determinado en tiempo de ejecución, según el valor asignado a la variable. Pudiendo cambiar el tipo de la variable si se le asigna otro.

Interpretado: No se debe compilar ni enlazar el código antes de su ejecución, sino que Python se encarga de ejecutar el código mediante un intérprete. Al ser un lenguaje interpretado ahorra mucho tiempo en el desarrollo de programas pero representa una mayor lentitud en su ejecución.

Sintaxis clara: Utiliza una sintaxis muy visual, gracias a que es de obligado cumplimiento la indentación como delimitador de bloques. Esto ayuda a que los programadores adopten las mismas notaciones y que los programas desarrollados tengan una apariencia muy similar.

Multiplataforma: Actualmente está disponible para las plataformas o sistemas operativos más importantes, como pueden ser, Windows, UNIX, Mac OS, Solaris, etc.

Multiparadigma: Evita que el programador se vea forzado a adoptar un estilo en particular, permitiendo el uso de varias formas de programación: orientada a objetos, estructurada y funcional, entre otras.

Funciones y librerías: Dispone de una gran variedad de funciones incorporadas en el lenguaje para el tratamiento de los diferentes tipos de datos, gestión de memoria, etc. Además, existen numerosas librerías con las cuales podemos ampliar específicamente las funcionalidades que este ofrece.

2.3. Django

El *framework* de desarrollo web Django [6] [17] nació en 2003 por el equipo de programadores web del diario *Lawrence Journal-World*, Adrian holovaty y Simon Willison. El equipo responsable de la producción y mantenimiento de los sitios locales de noticias, exigían que se pudiesen agregar de forma rápida y sencilla nuevas características o aplicaciones enteras. Debido a esta necesidad Adrian holovaty y Simon Willison crearon un *framework* de desarrollo web multiplataforma y escrito completamente en Python, denominado más tarde como Django en honor al guitarrista de jazz, Django Reinchardt. En 2005, una vez desarrollado, se decidió liberar el *framework* como software de código abierto.

Actualmente el *framework* Django es uno de los más conocidos y utilizado gracias a sus características y potencia, junto a Ruby on Rails, Symfony, CakePHP, etc. Este posee una amplia documentación y comunidad para facilitar su aprendizaje. Algunos de los sitios web que lo utilizan son, *The Washington Post, Tabblo de Hewlett Package, The Magazine Toronto Life*, etc.

El uso de un *framework* facilita y agiliza el desarrollo de páginas web dinámicas. Permitiendo a los diseñadores y programadores abstraerse de los problemas comunes al desarrollo web, proporcionando soluciones para las tareas de programación más frecuentes. Al utilizarlo, no es necesario plantearse una estructura sino que proporciona al programador la plantilla a seguir.

Otra de las características de Django es la gran API que tiene para el acceso a la capa de almacenamiento, si se hace uso de una base de datos, ofreciendo funciones

para realizar las diferentes tareas sobre esta capa. Esto viene de su origen, ya que en un principio se pretendía desarrollar entornos de noticias. Ofrece un mapeo objeto-relacional, técnica con la cual se consigue convertir los datos utilizados en la base de datos al sistema de programación utilizado, es decir, se crea una base de datos orientada a objetos virtual, sobre la base de datos relacional.

Además ofrece automáticamente una interfaz de administración, es decir, evita al programador el trabajo de crear las distintas interfaces para agregar, modificar o eliminar los diferentes elementos contenidos en el sitio web, a través de un panel de administración. También tiene soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.

Gracias a la utilización del *framework* Django, se dispone de un sistema limpio y elegante para la generación de las distintas URLs. Creando un módulo con el mapeo de los patrones URL (simples expresiones regulares) a las funciones de devolución de llamada Python. Además, es posible separar el código de la presentación mediante un sistema de plantillas, de esta manera se puede cambiar cómodamente el aspecto visual del sitio web sin afectar al contenido, y viceversa. El sistema de plantillas está basado en etiquetas y se permite la herencia entre estas.

Incorpora un servidor web ligero, válido para la fase de desarrollo y pruebas, siendo recomendada la utilización del servidor Apache en la fase de producción. Para la utilización de Apache u otros servidores, es necesario el uso de un módulo para comunicarlo con Django, ya que hay que integrar en este el lenguaje Python usado en todo el famework. Incluye de forma predeterminada SQLite3 como base de datos, facilitando el desarrollo y pruebas, siendo recomendable el uso de PostgreSQL, debido a sus características, aunque también soporta otras bases de datos.

Finalmente, destacar que Django sigue en cierta medida el patrón *Model-View-Controller* (MVC), pero debido a la forma en la que trabaja, los desarrolladores renombran este patrón a *Model-Template-View* (MTV).

2.4. Servidor HTTP: Apache

El Servidor Apache [4] se desarrolló en 1995, cuando Robert McCool abandona NCSA (*National Center for Supercomputing Applications*). El servidor NCSA HTTP sobre el que trabajaba vio paralizado su desarrollo temporalmente, lo que ocasionó que varios webmasters formaran el Grupo Apache, lo que se conoce como hoy en día *Apa-*

che Software Foundation. La primera versión de Apache se desarrollo basándose en el servidor utilizado por el NCSA, el más utilizado en ese periodo. Esta primera versión ha sufrido mejoras como software de servidor, inicialmente solo para UNIX, fruto de esas evoluciones es la versión para Windows. El nombre Apache viene de *A patchy server*, ya que inicialmente era un servidor NCSA parcheado.

Actualmente el servidor Apache tiene su propia licencia, siendo esta descendiente de la licencia BSD permitiendo su libre distribución, siempre que se reconozca su trabajo. Posee una gran comunidad de trabajo y una amplia documentación que sirve de guía a los desarrolladores web, siendo en los últimos años el servidor web más utilizado, gracias a las numerosas características y ventajas que presenta. Como se observa en la figura 2.1, este servidor es el más utilizado frente a otras opciones en los sitios web que utilizan el *framework* Django [7].

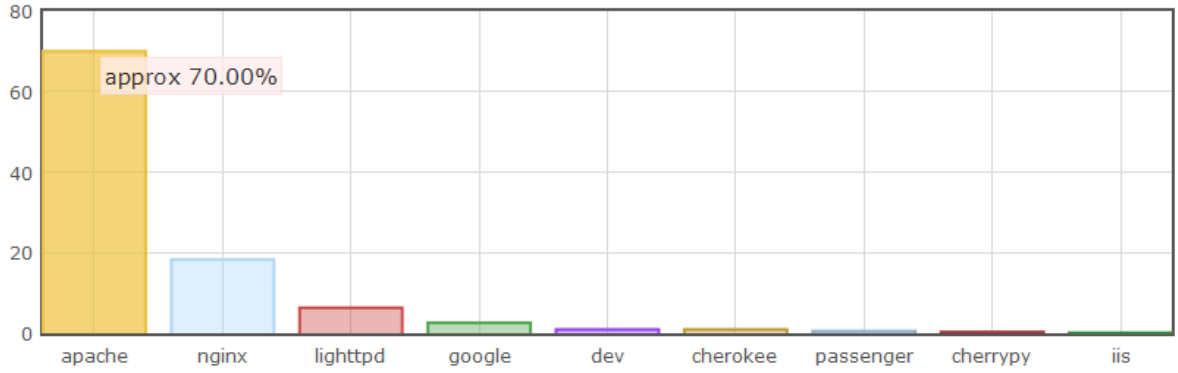


Figura 2.1: Gráfica de los servidores más utilizados en el *framework* Django

Apache es un servidor web muy potente, flexible y robusto que funciona en diferentes plataformas y entornos, lo que le hace prácticamente universal. El diseño modular de este servidor, permite a los desarrolladores adaptarlo a diferentes entornos y necesidades, gracias a la API de programación de módulos y a los numerosos complementos de apoyo que se proporcionan, ya sean para controlar el tráfico, incrementar su seguridad, etc. Además Apache permite trabajar con una gran cantidad de lenguajes, como pueden ser Python, Perl, PHP, etc., dando el soporte necesario para tener páginas dinámicas.

También permite la personalización de las diferentes respuestas ante los posibles errores que se puedan dar. Cuando se produzca un error en concreto es posible configurar el servidor para que se ejecute un determinado script. Finalmente, destacar la alta cantidad de opciones disponibles para la administración de logs (errores, acceso,

etc.), posibilitando la creación de diferentes ficheros destinados para el administrador, teniendo un mayor control sobre los sucesos que ocurren en el servidor.

La utilización de este servidor también conlleva una serie de desventajas. Al ser un producto multiplataforma, no aprovecha al máximo las posibilidades que ofrece el sistema operativo. Su configuración puede resultar bastante compleja, incluso para tareas sencillas, ya que no posee un formato estándar. Las principales vulnerabilidades de seguridad que presenta el servidor, solo pueden ser utilizadas por usuarios a nivel local, salvo si se utiliza el módulo para soportar las páginas dinámicas PHP, lo que ocasiona que algunas de estas vulnerabilidades sean accesibles remotamente.

2.5. Base de datos: PostgreSQL

La base de datos PostgreSQL [28] [19] fue creado en 1985, proyecto inicialmente liderado por Michael Stonebraker. Basándose en la experiencia obtenida con el proyecto Ingres (Base de datos relacional) y unos claros objetivos, empezó a trabajar en el proyecto Postgres. En 1994 dos estudiantes, Andrew Yu y Jolly Chen, comenzaron a trabajar sobre este proyecto, haciendo una limpieza del código, solucionando errores del mismo e implementando mejoras, entre las que destaca la sustitución del intérprete por el lenguaje SQL, obteniendo como resultado final de estos cambios lo que llamaron Postgres95, publicado bajo una licencia BSD (software libre). A partir de 1996 se creó *PosgreSQL Global Development Team*, haciéndose cargo del desarrollo del proyecto. Cambiaron el nombre a PostgreSQL, como lo conocemos en la actualidad, por el lenguaje SQL del que hace uso.

El gestor de Bases de Datos Relacionales PostgreSQL capaz de manejar complejas rutinas y reglas, es hoy en día una de las bases de datos más potentes y utilizadas del mercado, Como se observa en la figura 2.2, este sistema es uno de los más utilizados frente a otras opciones en los sitios web que utilizan el *framework* Django [7]. La estabilidad, potencia, robustez, facilidad de administración e implementación de estándares, son las características que más se han tenido en cuenta en su desarrollo. A continuación se exponen algunas de las más importantes y soportadas por SQL.

Es una base de datos full ACID (*Atomicty Consistency Isolation Durability*), es decir, posee las características para que una serie de instrucciones pueda ser considerada como una transacción.

- **Atomicidad:** Garantiza que las acciones se realicen o no se ejecuten, y en caso

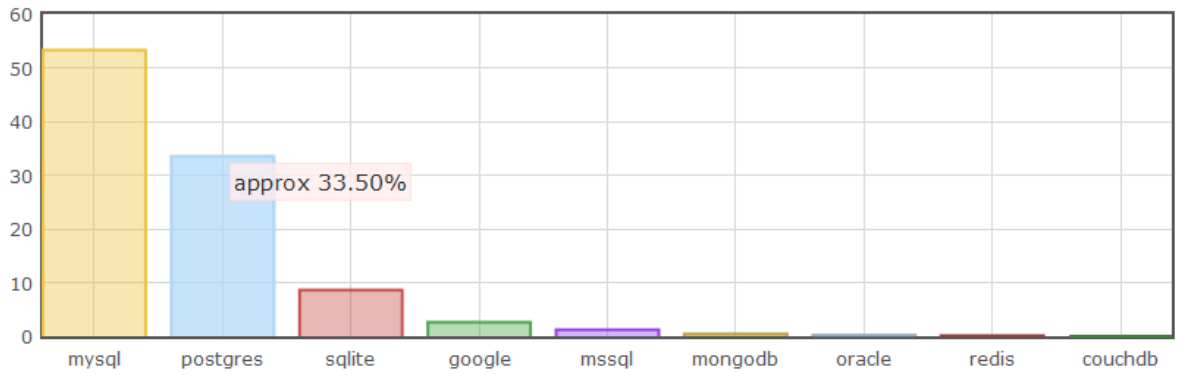


Figura 2.2: Gráfica de las Bases de datos más utilizadas en el *framework* Django

de error el sistema no se quede a medias.

- **Consistencia:** Propiedad que asegura que las reglas que hayan sido declaradas para una transacción sean cumplidas.

- **Aislamiento:** Garantiza que las transacciones que se estén realizando sean transparentes a todos los usuarios hasta que finalicen.

- **Durabilidad:** Propiedad que asegura que cuando finalice la transacción esta perdure a pesar de fallos en el sistema u otros motivos.

PostgreSQL está disponible para las principales plataformas Linux, Unix en todas sus versiones, Windows, etc. Además tiene un soporte nativo para los principales lenguajes de programación (PHP, C, C++, Perl, Python, etc.). También posee soporte para vistas, claves foráneas, integridad referencial, triggers, subconsultas, tipos de datos y operadores soportados por SQL2008 y predecesores. Cabe destacar que permite el uso de operadores, funciones, métodos de acceso y tipos de datos definidos por el propio usuario.

Soporta varios tipos de conexiones al servidor, locales o por red (TCP/IP). En el caso de utilizar una conexión a través de la red, se puede utilizar el cifrado SSL para las comunicaciones, teniendo como único requisito la instalación de OpenSSL en nuestro servidor. Además la base de datos posee utilidades para su limpieza, el proceso Vacuum es el encargado de realizar dicha tarea. Se encarga de eliminar las tuplas que han sido marcadas como borradas, ya que el motor no las elimina inmediatamente para no sobrecargar las operaciones.

El sistema de base de datos PostgreSQL integra un sistema Control de Concurrencia Multi-Versión (MVCC), siendo un control de concurrencia optimista. Gracias a esta tecnología se evita bloqueos innecesarios sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles mientras otro usuario los actualiza, de esta forma un lector nunca es bloqueado cuando quiere obtener información.

2.6. LibreGeoSocial

LibreGeoSocial (LGS) [22] es un software libre y de código abierto desarrollado por investigadores del GSYC de la Universidad Rey Juan Carlos (URJC), destinado para los terminales móviles con Android. LGS es una red social que cuenta con diferentes módulos posibilitando que los diferentes nodos que componen la red social sean geolocalizados y posicionados a diferentes altitudes, permitiendo etiquetar diferentes objetos del mundo (texto, imágenes, video,...), poniendo a disposición de los demás usuarios de la red social visualizar la información y autor, a través de su dispositivo móvil cuando apunta al objeto.

LGS [21] principalmente está estructurado en dos componentes: LibreGeoSocial, es el *framework* desarrollado en Django que facilita la creación de redes sociales en dispositivos móviles, posibilitando a los usuarios conocer el posicionamiento de los diferentes nodos que lo componen. LibreGeoSocialApp, es una aplicación desarrollada para los terminales Android que aprovecha las funcionalidades ofrecidas por el componente anteriormente mencionado, posibilitando conocer de su posición a través del GPS y señal wi-fi. Además de obtener información de los lugares más cercanos a través de diferentes canales (YouTube y Panoramio, entre otros), encontrar amigos así como ver su posicionamiento, etc.

Los módulos de realidad aumentada y georreferenciación permiten a los diferentes usuarios interaccionar de una manera diferente con el mundo real, abriendo un amplio abanico de utilidades: guías turísticas, sistemas de participación ciudadana, redes sociales, etc.

2.7. Android

En 2003 se funda en Palo Alto (California) Android Inc. de la mano de Andry Rubin, Rich Miner, Nick Sears y Chris White. La compañía estaba centrada inicialmente en

el desarrollo de software para teléfonos móviles. En 2005 Google empezó a hacerse con algunas *startup* (compañías con un futuro prometedor) de sector móvil, una de las empresas adquiridas era Android Inc., entre otras. A partir de ese momento se empezaron a escuchar numerosos rumores sobre las pretensiones que tenía Google en el mercado de la telefonía móvil, siendo el 5 de noviembre de 2007 cuando se hizo oficial el anuncio del desarrollo del sistema operativo Android que revolucionaría el mercado. No fue hasta un año después, en Octubre 2008 cuando se fabricó el primer dispositivo con el sistema operativo Android 1.0, el HTC Dream(G1). Uno de los elementos que lo diferencia del resto de sistemas operativos, es la cantidad de actualizaciones y versiones que recibe [36], como se observa en la figura 2.3.

Figura 2.3: Diagrama con las diferentes versiones de Android

Google en 2005 fundó la *Open Handset Alliance* (OHA), lo que facilitó el desarrollo e implementación de Android en terminales móviles. Formada por diferentes fabricantes y desarrolladores de hardware, software y operadores de servicio. Teniendo como objetivos acelerar la innovación en las compañías móviles y ofrecer a los consumidores una experiencia intuitiva y rica.

Android [1] [3] [31] es una plataforma de desarrollo de software y un sistema operativo para terminales móviles basado en un *kernel* Linux. Google pone a disposición de los desarrolladores una API en el lenguaje de programación Java y utilidades, gracias a la herramienta Android SDK, con las cual escribirán el código que se ejecute en los dispositivos. La mayor parte de la plataforma está disponible bajo la licencia Apache 2.0 por lo que su distribución es libre y posibilita el acceso y modificación de su código fuente.

A continuación se exponen y explican algunas de sus principales características:

- ***framework* de aplicaciones,** los desarrolladores tienen acceso a las mismas APIs del *framework* usadas por la aplicación base. La arquitectura está diseñada para simplificar la reutilización de componentes.

- **Maquina virtual Dalvik,** es la máquina que se pone a disposición de los desarrolladores para ejecutar las aplicaciones. Está optimizada para requerir pocos recursos, utilizar menos espacio y permite el funcionamiento de varias a la vez.

- **Navegador,** se integra en el sistema un navegador web, basado en el motor de código abierto WebKit.

- **Optimización de gráficos,** se incluye librerías para la representación de gráficos 2D y gráficos 3D basados en OpenGL ES.

- **SQLite,** Android utiliza el gestor de datos SQLite para el almacenamiento estructurado de los datos. También se permite la utilización de otras bases de datos o incluso utilizar las clases de la API para el almacenamiento de los datos sin hacer uso del gestor.

- **Soporte multimedia,** se da soporte a multitud de formatos de audio (MP3, MIDI, WAV, etc.), vídeo (MPEG-4, WebM, etc.) e imágenes (JPEG, PNG, GIF, etc.).

- **Soporte para hardware,** cámara, brújula, acelerómetro, GPS, pantalla táctil, etc., son soportados por el terminal dependiendo del hardware que este compuesto.

- **Rico entorno de desarrollo,** se facilita el desarrollo de las aplicaciones a través del Android SDK, ya que incluye un emulador, herramientas de depuración y perfiles de memoria, además se puede instalar un plug-in para el entorno de desarrollo Eclipse.

La arquitectura de Android, es un sistema diseñado por diferentes capas de software que facilitan al desarrollador la creación de aplicaciones. Esta distribución permite el acceso a las capas inferiores mediante el uso de librerías para que así no se tenga que programar a bajo nivel las diferentes funcionalidades necesarias que hacen uso de los componentes de hardware. Cada una de estas capas utilizan servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores, tal como muestra la siguiente figura 2.4.

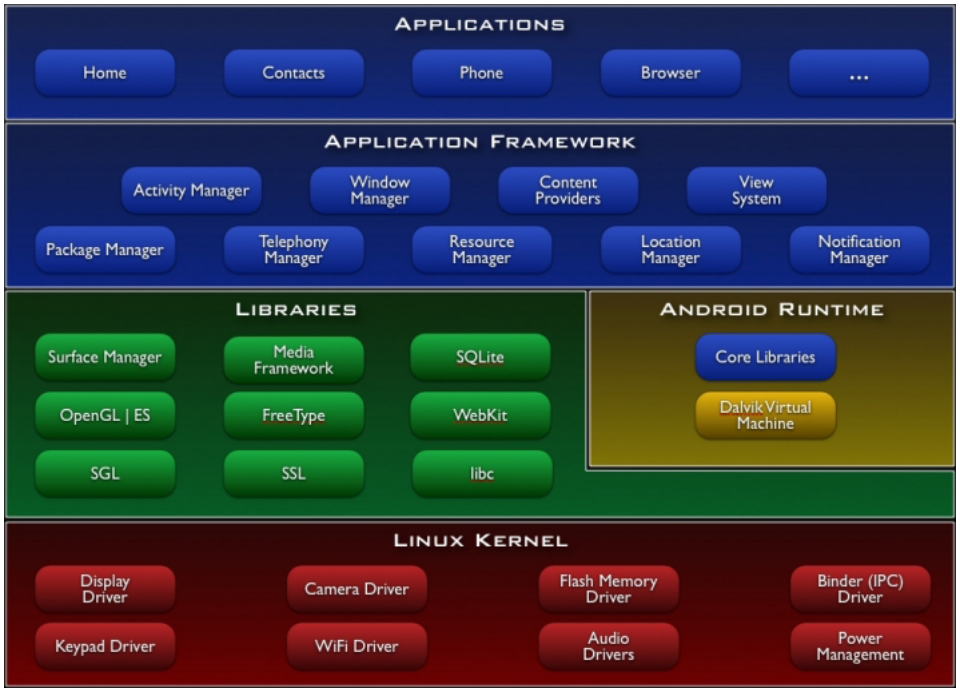


Figura 2.4: Diagrama con las capas y componentes que conforman la arquitectura de Android

Aplicaciones

Este nivel contiene, tanto las aplicaciones que incluye por defecto Android (cliente de correo, mapas, etc.) como aquellas que el desarrollador vaya añadiendo posteriormente. Todas estas pueden utilizar las librerías y servicios que proporcionan los niveles inferiores.

***framework* de Aplicaciones**

Compuesta por una completa API que ofrece al desarrollador la capacidad de crear aplicaciones muy ricas e innovadoras. La arquitectura está diseñada para que la reutilización de componentes sea sencilla, posibilitando que cualquier aplicación pueda ofrecer sus capacidades o hacer uso de otras, consiguiendo un alto grado en la reutilización de código. Los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik.

Bibliotecas

Esta capa situada sobre el *kernel* de Linux está compuesta por las bibliotecas nativas de Android. Están escritas en C o C++ y compiladas específicamente para la arquitectura hardware del teléfono. Proporcionan la mayor parte de las

funcionalidades presentes y que son utilizadas por diferentes componentes del sistema Android. Estas capacidades son accesibles a través de la capa superior (*framework* de aplicaciones). Junto al *kernel*, estas librerías constituyen el corazón de Android.

Entorno de ejecución de Android

En la misma capa que las librerías, se sitúa el entorno de ejecución, compuesto principalmente por la máquina virtual Dalvik. Las aplicaciones se desarrollan bajo Java pero al ser compiladas cambian su formato para que esta máquina virtual las ejecute. La ventaja de esto es que son compiladas una única vez, estando lista para distribuirse y ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema que requiera la aplicación. La ejecución de cada una de estas crea su propia instancia de la máquina virtual en un proceso, de esta manera un dispositivo puede lanzar múltiples instancias de dicha máquina.

***Kernel* de Linux**

El sistema operativo Android está basado en el *kernel* de Linux 2.6, similar al de cualquier distribución de Linux, solo que adaptado a las características hardware que ofrece un dispositivo móvil. Esta capa es utilizada como una abstracción de los elementos físicos, conteniendo los drivers necesarios para la correcta utilización mediante las llamadas de cualquier componente del terminal, como la cámara. Además este nivel también es el encargado de gestionar los principales servicios del sistema como pueden ser la seguridad, batería y gestión de memoria. Cuando un fabricante incluye un nuevo componente, lo primero que debe de realizar para su correcta utilización desde Android, es crear su correspondiente controlador dentro del *kernel*.

Una aplicación Android está compuesta por una combinación de bloques o componentes, cada uno de los cuales desempeña un papel específico, ayudando a definir el comportamiento general de la aplicación. Estos componentes deben de ser declarados en el fichero *AndroidManifest.xml*. En este se encuentran definidos otros datos importantes como los permisos, siendo un fichero indispensable para cualquier programa. A continuación se exponen los diferentes componentes de una aplicación Android, teniendo cada uno de estos un propósito y ciclo de vida distinto.

Activity

Es el componente encargado de construir la interfaz de usuario (GUI) y el com-

ponente más habitual de las aplicaciones Android, siendo el responsable de representar los elementos visuales y reaccionar a las acciones del usuario. Refleja una actividad llevada a cabo por el programa, además tiene asociada la interfaz representada por la clase *View* y sus derivados. Este componente se implementa mediante la clase del mismo nombre *Activity*. La mayoría de las aplicaciones poseen de varias pantallas con diferentes objetivos, es decir, estará compuesta por distintas actividades. El paso de una pantalla a otra se consigue mediante la inicialización de una nueva *Activity*. Toda actividad es inicializada mediante un *Intent*, cuando se abre una nueva, la anterior se queda pausada dentro de la pila, permitiendo al usuario navegar entre las diferentes actividades abiertas. Android permite controla por completo el ciclo de vida del componente *Activity* a través de diferentes métodos que cambia su estado, estos son los cuatro posibles: activada, pausada, parada y reiniciada.

Service

Son componentes sin interfaz gráfica que se ejecutan en segundo plano (*background*) mientras otras aplicaciones (con interfaz) están activas en la pantalla del dispositivo. Un *Service* puede realizar cualquier tipo de acciones durante un tiempo prolongado, como actualizar periódicamente datos o lanzar notificaciones. Los servicios son ejecutados en el proceso principal de la aplicación, pudiendo que ésta quede bloqueada si consume una gran cantidad de tiempo, por esto es recomendable lanzar los servicios en un *Thread*. De igual manera que una *Activity*, se implementa mediante la clase con su mismo nombre *Service* y posee un ciclo de vida propio y más simple.

Broadcast receiver

Este componente está destinado a detectar y reaccionar ante eventos o determinados mensajes generados por el sistema, por ejemplo: una llamada entrante o un SMS recibido. Este componente no tiene asociado una interfaz de usuario, al igual que un *Service*, pero se pueden crear notificaciones en la barra de estado de la interfaz del teléfono a través de la API *NotificationManager*, para avisar al usuario cuando un evento de este tipo se genera. De esta forma se pueden iniciar diferentes componentes como respuesta del evento generado. Además es posible generar en la propia aplicación mensajes broadcast dirigidos a cualquier componente de este tipo que este activo. Este componente se implementa a través de la clase *BroadcastReceiver*.

Content provider

Este elemento ofrece al desarrollador la posibilidad de almacenar los datos en el sistema, ya sea en una base de datos SQLite o en cualquier otro formato que considere. A través de este componente, se proporcionará una serie de métodos que permiten almacenar, recuperar, actualizar y compartir los datos entre aplicaciones, sin mostrar detalles sobre su forma de almacenamiento y estructura. Por defecto en la API de Android ya se encuentran definidos algunos de estos componentes y permiten compartir todo tipo de datos, información de contactos, imágenes, audio o incluso vídeo. Se implementa a través de las subclases de *ContentProvider* y se debe implementar una serie de funciones para permitir que otras aplicaciones realicen las transacciones.

Para el facilitar el desarrollo de aplicaciones en Android, Google ha creado un plug-in ADT (*Android Development Tools*) adaptado para el entorno de desarrollo Eclipse, de gratuita distribución. Este plug-in junto Android SDK facilita la creación de una manera rápida e intuitiva la creación de aplicaciones para los dispositivos con este sistema operativo.

Al crear un nuevo proyecto en esta plataforma en el IDE Eclipse [8], se genera un árbol de directorios (ver figura 2.5) que constituirá el esqueleto básico. A continuación se explica con detalle el contenido y propósito de la estructura generada.

El nodo inicial es la carpeta con el nombre del proyecto con el que será identificado dentro de Eclipse, esta contendrá diversas carpetas. En el directorio "src/" se encuentran los paquetes que forman la aplicación, es decir, el código que hemos implementado. La carpeta "gen/" se encuentra un paquete con un único fichero, *R.java*, este será generado automáticamente por el entorno de desarrollo y no debe de ser modificado por el usuario. Estableciendo en este fichero la relación entre la interfaz gráfica y su correspondiente implementación en el código a través de referencias.

Como otro nodo de la carpeta inicial, se localiza un nuevo paquete. En este se incluyen las librerías Java con las funciones propias de la versión Android, ya que al crear el proyecto se puede elegir la versión para la que se desea desarrollar la aplicación. En el caso que sea necesario añadir alguna librería externa al sistema operativo, serán contenidas en un nuevo paquete.

Contenida dentro del nodo inicial se localiza otra carpeta importante, denominada "res/", con varias subcarpetas. En primer lugar se localiza la carpeta "drawable/", donde se encuentran todas las imágenes que se utilizan en la interfaz gráfica de la

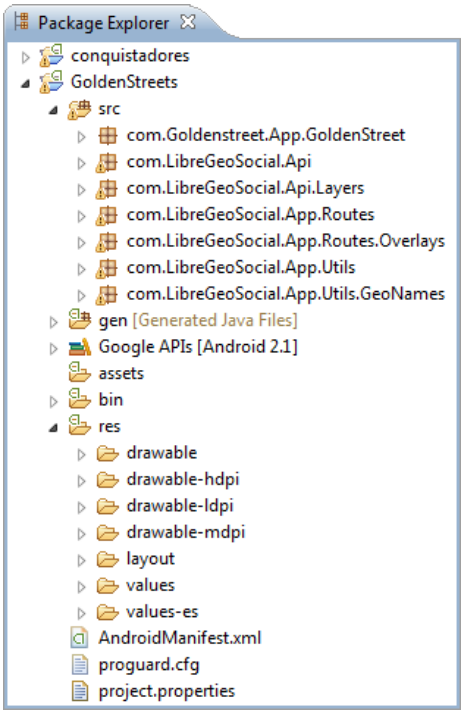


Figura 2.5: Estructura del árbol de directorios de un proyecto del IDE Eclipse

aplicación. Esta carpeta se puede dividir en tres si se desea especificar las imágenes según la resolución de los dispositivos (hdpi, mdpi y ldpi). La carpeta "layout/" contiene todos los ficheros *.xml* que definen la GUI de cada pantalla, todos ellos utilizados desde el código Java. Por último se encuentra la carpeta "values/" que contiene el archivo *string.xml*, en este se definen todas las cadenas de texto que se utilizan en nuestra aplicación. De esta forma se facilita crear aplicaciones en varios idiomas, por defecto esta carpeta está enfocada al idioma inglés, salvo que se introduzca el texto en otro idioma, en el caso que se desee traducir una aplicación se crea una nueva carpeta "values-" y el código del idioma. En el caso que se desee añadir la traducción de las cadenas de texto al español, la carpeta se denominará "values-es".

Por último destacar el fichero *AndroidManifest.xml*, escrito en XML, que es imprescindible en cualquier proyecto Android. Este archivo es generado automáticamente en la raíz del proyecto creado por Eclipse. Describe los diferentes componentes que se encuentran en la aplicación (activity, service, etc) y cómo interactúan entre ellos y con el sistema. En este fichero, también se especifican los diferentes permisos que necesita la aplicación para su correcto funcionamiento, como puede ser la versión mínima de Android, acceso al GPS, entre otros.

Capítulo 3

Objetivos

Explicadas en capítulos anteriores las motivaciones que ha llevado a la realización de este proyecto y descrito las tecnologías y herramientas utilizadas. En este capítulo, se expone la descripción del presente Proyecto Fin de Carrera y los objetivos que se pretenden alcanzar con su realización. También se comenta la metodología empleada que se ha seguido a lo largo de su desarrollo.

3.1. Descripción del proyecto

El presente proyecto se centra en el estudio de la plataforma Android y en la utilización de la red social LibreGeoSocial, desarrollada por el grupo LibreSoft [23] del GSyC [12]. Esto junto a las características que ofrecen actualmente los terminales y los servicios de telefonía, proporcionan al desarrollador las herramientas necesarias para la creación de nuevas aplicaciones con un gran potencial, como puede ser *GoldenStreets*.

El objetivo principal es desarrollar una plataforma software, a modo de Cliente-Servidor, pudiéndose considerar como un *location-based game*, que utiliza las herramientas y tecnologías expuestas en el capítulo 2. Ofreciendo a los usuarios, denominados Organizadores, la administración de los diferentes elementos de las partidas a través de un servicio web, y a los usuarios, denominados jugadores, participar en estas mediante su dispositivo móvil.

Para la consecución de este objetivo, se han planteado a continuación una serie de sub-objetivos, con los cuáles conseguimos simplificar el problema en otros más pequeños y fáciles de resolver.

Utilización y evaluación de las tecnologías empleadas:

Se pretende obtener un amplio conocimiento sobre las principales tecnologías empleadas en el desarrollo del presente proyecto, como pueden ser: Android, Python, Django, y Apache, entre otras. Para su adquisición se utilizará diferentes sitios web, libros, artículos, etc. expuestos en la bibliografía. Una vez evaluadas y analizadas si no cumplen las expectativas deseadas o no ofrecen el rendimiento requerido, se estudiarán otras alternativas.

Creación del sistema:

Es necesario ofrecer soporte a los diferentes tipos de usuarios que manejen las diferentes partes del sistema, es decir a los Organizadores como a los jugadores. Para esto, se proporciona a los primeros un servicio web con una fácil e intuitiva interfaz web, para la creación, gestión y seguimiento de los diferentes elementos de una partida. Además de poder establecer comunicación con cualquier usuario mediante un sistema interno implementado en el servicio y en la aplicación.

Además se construirá una aplicación destinada a la mayoría de versiones disponibles de Android para terminales móviles, mediante las tecnologías analizadas. Para su desarrollo será necesario tener en cuenta los requisitos del proyecto y los diferentes aspectos del diseño. A través de la aplicación se pretende dar soporte a los jugadores para que puedan participar mediante la interfaz de su terminal móvil en las partidas creadas con el servicio web.

Validación y utilización del sistema:

Verificar el correcto funcionamiento del presente proyecto en un escenario real y con diferentes voluntarios, mediante la realización de diferentes pruebas. Para descubrir y corregir los posibles errores en la implementación del sistema, así como la opinión y aceptación por parte de ellos a través de diferentes encuestas.

3.1.1. Descripción GoldenStreets

GoldenStreets es un juego geolocalizado para terminales Android, con tintes del tradicional juego de mesa Monopoly [14], en donde las calles toman otra dimensión correspondiendo con ubicaciones reales.

El objetivo principal del juego se basa, en alcanzar con el equipo utilizado el mayor patrimonio posible. Para lograr tal fin, el equipo tendrá que pensar en la mejor estrategia y desplazarse a las calles que quiera comprar y que más se adecuen a su poder

adquisitivo, pudiendo edificar en ellas para incrementar sus ingresos o sabotear las de sus adversarios para reducir sus beneficios y patrimonio. Además cada uno de ellos, podrá realizar otras acciones adicionales (comprar, construir, vender, ofertar, demoler y sabotear) para alcanzar dicho objetivo, así como establecer comunicación con sus rivales para establecer negociaciones sobre sus propiedades.

Otro elemento disponible para los equipos son las diversas cartas habilitadas en el juego, con las cuales se puede ver beneficiado o perjudicado el equipo poseedor. Este elemento será necesario en algunas ocasiones para realizar determinadas acciones sobre sus propiedades o las de sus rivales, como puede ser el sabotaje. Algunas de estas cartas serán aplicadas automáticamente por la aplicación, siendo cada una de estas de un único uso. En el transcurso de una partida cada jugador recibirá periódicamente sus cartas.

A través de la página web, el Organizador podrá personalizar y gestionar las partidas creadas, pudiendo realizar un seguimiento en tiempo real de las partidas que estén en juego y establecer comunicación con los equipos participantes, para la resolución de cualquier consulta.

3.2. Metodología empleada

Es difícil controlar los riesgos producidos en el desarrollo del software, por eso es necesario utilizar una metodología de desarrollo, en caso contrario, es posible que el software que se pretende desarrollar no se ajuste a las necesidades y funcionalidades marcadas. Para afrontar este Proyecto Fin de Carrera se ha seguido en mayor medida el Proceso unificado.

El Proceso Unificado de Desarrollo (PUD) [16] es una metodología para acometer una gran variedad de sistemas software, ajustándose a diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y tamaños de proyectos. Está dirigido por los casos de usos, ya que estos especifican las diferentes funcionalidades y representan los requisitos funcionales del sistema. Además guían el desarrollo del software a través de diferentes flujos de trabajo (requisitos, análisis, diseño, implementación y pruebas).

Este proceso es iterativo e incremental, dividiendo el trabajo pequeñas partes, también llamadas iteraciones. En cada iteración se amplía el sistema con nuevos casos de usos y nuevos riesgos, agrupándose cada una de estas en diferentes fases, por orden secuencial son las siguientes: inicio, elaboración, construcción/verificación y transición.

Cada una de estas centra más sus esfuerzos en diferentes flujos de trabajo.

La fase de inicio tiene como objetivo desarrollar el análisis de negocio hasta las justificaciones necesarias para la puesta en marcha del proyecto. Lo importante de esta fase es que se puede desarrollar de múltiples formas, pudiendo recopilar los diferentes objetivos, definir la arquitectura o realizar un profundo trabajo de investigación. Debido a esto, la fase de inicio se centra principalmente en los flujos de captura de requisitos y análisis.

Durante la fase de elaboración, se especifica con mayor detalle el diseño de la arquitectura, siendo su relación con el sistema primordial. En esta fase hay diferentes modelos del sistema, centrándose en los flujos de trabajo de análisis y diseño. Al final de esta fase el proyecto debe de estar en disposición de planificar las actividades y estimar los recursos necesarios para terminar el proyecto.

La fase de construcción, evoluciona la descripción inicial hasta obtener el sistema completo, siendo en esta donde se utilizan la mayor parte de los recursos requeridos. Se hace un mayor hincapié en los flujos de diseño, implementación y pruebas. La arquitectura en este punto es estable, pero se pueden aplicar diferentes mejoras que no afecten al sistema en sí. Al final de la fase el cliente puede obtener la versión con las funcionalidades requeridas para las pruebas.

La fase de transición es la última del PUD, los clientes realizan las pruebas del software entregado y realizan los reportes de los defectos encontrados, como cambios que consideren oportunos, siendo los desarrolladores los encargados de solventarlos. Además se forma al usuario para una correcta utilización y rápido aprendizaje. Como en la fase anterior se centre en los mismos flujos de trabajo, pero con una menor carga sobre el diseño e implementación.

La utilización de la metodología del proceso unificado de desarrollo, posibilita en varias ocasiones revisar el sistema, detectando los posibles errores y corregirlos. Además se reducen los riesgos de obtener un producto final no deseado, ya que en cada momento hay una versión del sistema funcionando que se modifica con las nuevas necesidades que vaya solicitando el cliente. Aunque su utilización para el desarrollo de un sistema pueda resultar un poco tedioso, al tener una documentación tan pesada.

Capítulo 4

Análisis y Diseño

El objetivo principal de este capítulo no es otro que el análisis y diseño realizado en el Proyecto Fin de Carrera. En el análisis se obtendrá la especificación detallada de las principales características y funcionalidades, teniendo así una sólida base sobre la que realizar el diseño y posteriormente su implementación. Los requisitos obtenidos se deben de realizar de manera satisfactoria, ya que una mala identificación puede conducir a la aplicación a un fracaso en términos de aceptación por parte de los usuarios. También se detalla el diseño de la aplicación teniendo en cuenta todas las decisiones tomadas.

4.1. Especificación de requisitos de software

Para alcanzar los objetivos propuestos se han especificado los requerimientos, sobre los cuales se basará el diseño e implementación de los diferentes clientes (web y Android) y definirán el comportamiento del sistema. Estos pueden dividirse en funcionales (RF) y no funcionales (RNF). Los funcionales definen las principales acciones que se podrán realizar sobre el sistema, como por ejemplo, administrar los diferentes elementos, visualizar los datos, etc. Los no funcionales están enfocados a las características que pueden limitar el sistema, como por ejemplo, rendimiento, fiabilidad, seguridad, etc.

Debido a la complejidad de cada cliente, se ha decidido crear un subapartado para cada uno de ellos, en el cual, se detallarán de forma tabular los requisitos de software. También se acompañará de dos diagrama de casos de usos (figura 4.1 y figura 4.2) describiendo las diferentes interacciones de los usuarios con los clientes. La utilización de estos diagramas definen el comportamiento del sistema desde el punto de vista del usuario, es decir, representan las diferentes funciones que podrán realizar.

4.1.1. Cliente Web

Autenticación	
Identificador	RF-01
Descripción	Requerirá la autenticación para acceder al servicio web, mediante la cumplimentación de un formulario.
Entrada	Datos del formulario.
Salida	Acceso al servicio web como un usuario con permisos de visualización o un Organizador, este último con todas las funciones.

Cuadro 4.1: RF-01 Autenticación.

Visualizar elementos	
Identificador	RF-02
Descripción	Visualizará el listado de los diferentes elementos (cartas, juegos, equipos, calles, edificios), dando la posibilidad de mostrar de forma detallada con tan solo pulsar sobre el enlace de su nombre.
Entrada	Datos de los elementos almacenados.
Salida	Listado o información detallada de un elemento.

Cuadro 4.2: RF-02 Visualizar elementos

Administrar elementos	
Identificador	RF-03
Descripción	Administrará los diferentes elementos (cartas, juegos, equipos, calles, edificios), posibilitando su creación, modificación y eliminación. Acciones únicamente realizables por un Organizador.
Entrada	Datos del formulario.
Salida	Creación, modificación o eliminación de los elementos sobre los que se realicen la acción.

Cuadro 4.3: RF-03 Administrar elementos

Monitorizar juego	
Identificador	RF - 04
Descripción	Visualizará la clasificación de los equipos, mapa con la ubicación de los diferentes elementos y mensajes intercambiados en el transcurso de una partida.
Entrada	Elementos que componen el juego, como son, equipos, calles, edificios.
Salida	Seguimiento del juego a través de la visualización de la clasificación de los equipos y mapa.

Cuadro 4.4: RF-04 Monitorizar el juego

Sistema de mensajería	
Identificador	RF - 05
Descripción	Permitirá el envío de un mensaje a uno o todos los equipos asociados al juego. Acción únicamente realizable por un Organizador.
Entrada	Datos del formulario.
Salida	Envío de un nuevo mensaje al destinatario o destinatarios, emitido por el Organizador.

Cuadro 4.5: RF-05 Sistema de mensajería

Validar formularios	
Identificador	RF - 06
Descripción	Comprobará la correcta cumplimentación del formulario por parte del Organizador, para su posterior envío al servidor.
Entrada	Datos del formulario.
Salida	Validación y envío al servidor de los datos ingresados en el formulario.

Cuadro 4.6: RF-06 Validar formularios

Facilidad de aprendizaje cliente web	
Identificador	RNF - 01
Descripción	Deberá de ser fácil de usar para que los usuarios aprendan, de forma rápida, los pasos requeridos para realizar las diferentes acciones.
Entrada	-
Salida	Aprendizaje rápido e interfaz web intuitiva.

Cuadro 4.7: RNF-01 Administrar los equipos

Fiabilidad cliente web	
Identificador	RNF - 02
Descripción	Informará de los errores producidos para su reconocimiento y posterior recuperación por parte de los usuarios.
Entrada	-
Salida	Visualización sencilla y clara de los mensajes de error.

Cuadro 4.8: RNF-02 Fiabilidad ciente web

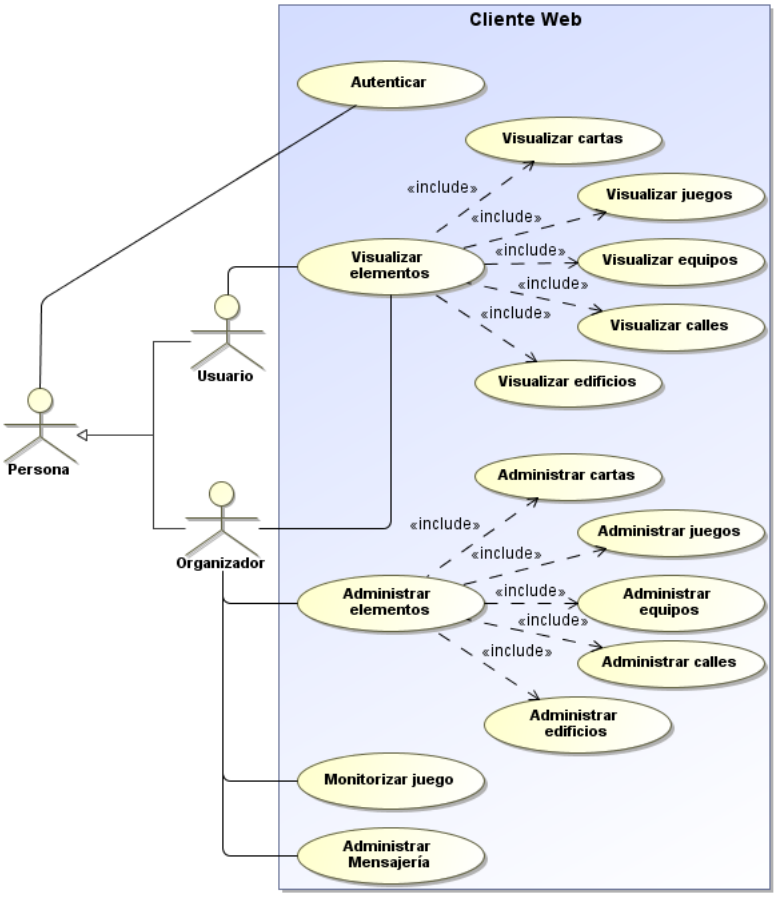


Figura 4.1: Diagrama de Casos de usos del Cliente Web

4.1.2. Cliente Android

Seleccionar datos de la partida	
Identificador	RF - 07
Descripción	Mostrará el listado de los juegos disponibles y posteriormente el listado de los equipos para su selección.
Entrada	Datos de los juegos y equipos.
Salida	Selección del juego y equipo deseado por el usuario para su participación en la partida.

Cuadro 4.9: RF-07 Seleccionar datos de la partida

Estadísticas del juego	
Identificador	RF - 08
Descripción	Mostrará el tiempo restante de la partida y la clasificación de los equipos.
Entrada	Datos del juego y equipos participantes.
Salida	Tiempo restante y clasificación de los equipos participantes.

Cuadro 4.10: RF-08 Estadísticas del juego

Sistema de cartas	
Identificador	RF - 09
Descripción	Mostrará el listado de las cartas en su posesión. Si se pulsa sobre algún elemento del listado se verá la información detallada de la carta elegida.
Entrada	Datos de las cartas.
Salida	Listado o información detallada de una carta en posesión del equipo.

Cuadro 4.11: RF-09 Sistema de cartas

Mapa del juego	
Identificador	RF - 10
Descripción	Mostrará un Google Maps con la ubicación de las calles y equipos, representados mediante diferentes iconos que determinan su estado actual.
Entrada	Datos del juego y sus elementos asociados.
Salida	Mapa de la partida con la ubicación de las calles y equipos participantes.

Cuadro 4.12: RF-10 Mapa del juego

Sistema de ofertas	
Identificador	RF - 11
Descripción	Mostrará la opción para listar las ofertas recibidas o enviadas por el equipo. Si se pulsa sobre algún elemento del listado se verá la información detallada de la oferta elegida.
Entrada	Datos de las ofertas asociadas al equipo.
Salida	Listado o información detallada de una carta en posesión del equipo.

Cuadro 4.13: RF-1 Sistema de ofertas

Sistema de mensajería	
Identificador	RF - 12
Descripción	Mostrará la opción para listar todos los mensajes del equipo y la opción para enviar un mensaje a un equipo o al Organizador.
Entrada	Datos del formulario.
Salida	Listado, información detallada o envío de un nuevo mensaje al destinatario elegido por el equipo.

Cuadro 4.14: RF-12 Sistema de mensajería

Acciones aplicación Android	
Identificador	RF - 13
Descripción	Podrá realizar diferentes acciones (comprar, vender, edificar, sabotear, demoler y realizar una oferta) dependiendo del estado de la calle y su propietario. Solo serán realizables si el jugador se encuentra a una determinada distancia.
Entrada	Datos del juego y sus elementos asociados.
Salida	Notificación informando del resultado de la acción realizada.

Cuadro 4.15: RF-13 Acciones cliente Android

Facilidad de aprendizaje cliente Android	
Identificador	RNF - 03
Descripción	Los botones, iconos, notificaciones, etc., deben de ser del tamaño adecuado para una correcta visualización por parte del usuario. Facilitando un rápido aprendizaje para realizar las diferentes acciones.
Entrada	-
Salida	Interfaz clara, sencilla e intuitiva para un rápido aprendizaje.

Cuadro 4.16: RNF-03 Facilidad de aprendizaje cliente Android

Fiabilidad cliente Android	
Identificador	RNF - 04
Descripción	Informar de los errores producidos para su reconocimiento y posterior recuperación por parte del usuario.
Entrada	Datos del error producido.
Salida	Visualización clara de los mensajes de error.

Cuadro 4.17: RNF-04 Fiabilidad cliente Android

Prestaciones	
Identificador	RNF - 05
Descripción	Deberán de minimizarse los tiempos de espera en la realización de las diferentes acciones, la respuesta del GPS, etc.
Entrada	-
Salida	Buen rendimiento de la aplicación móvil.

Cuadro 4.18: RNF-05 Prestaciones

Plataforma de destino	
Identificador	RNF - 06
Descripción	Deberá de ser funcional en dispositivos Android con la versión 1.6 o posteriores.
Entrada	-
Salida	Compatibilidad de la aplicación con los diferentes dispositivos móviles.

Cuadro 4.19: RNF-06 Plataforma de destino

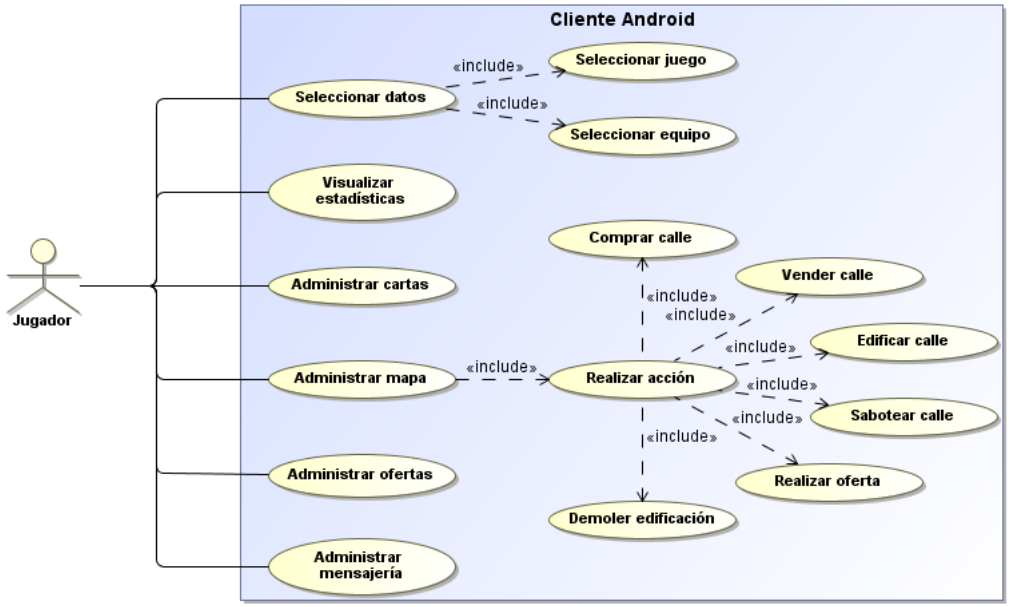


Figura 4.2: Diagrama de Casos de usos del Cliente Android

4.2. Definición del sistema

En este apartado se van a definir los aspectos más importantes relacionados con la arquitectura escogida para las necesidades del sistema. La Arquitectura de software es el diseño de más alto nivel de la estructura del sistema [15], definiendo los diferentes componentes que realizan alguna tarea de computación, de interfaz o de comunicación. Concretando, la arquitectura física que se implementará, es decir, determinando que computadora o dispositivo tendrá asignado cada componente y como se relacionan entre ellos.

La arquitectura que se ha elegido para la realización del proyecto debido a las necesidades y objetivos requeridos, se basa en el clásico diseño multinivel en capas, más concretamente, como se puede observar en la figura 4.3 , se utilizará la arquitectura en 3 capas (*Presentación*, *Negocio* y *Datos*). Cada capa solamente tiene relación con las contiguas y son independientes. El objetivo primordial de la elección de esta arquitectura se basa en la modularidad que ofrece, separando la lógica de negocio de la lógica de diseño. Este estilo posibilita el desarrollo en varios niveles y en caso de que se necesite realizar algún cambio, sólo se modifica el código de la capa requerida sin afectar a los módulos restantes. A continuación se describen las funcionalidades de cada una de las capas y las interrelaciones entre ellas:

Figura 4.3: Diagrama del diseño de la arquitectura tres capas utilizada en el sistema

- **Capa de presentación:** Esta capa también es conocida como la interfaz gráfica encargada de informar y capturar las acciones del usuario, se comunica únicamente con la capa inferior (*negocio*). En este nivel se pueden distinguir los diferentes clientes utilizados por los usuarios. El cliente Android enfocado a los jugadores y a los dispositivos móviles que cuenten con este sistema operativo, recibiendo la información en formato JSON [20]. Otro cliente es el web, destinado a los Organizadores, accesible mediante la utilización de cualquier navegador del ordenador u otro dispositivo, recibiendo la información del servidor en formato HTML.

- **Capa de negocio:** Capa que proporciona toda la funcionalidad a las diferentes peticiones realizadas por los usuarios, actuando de intermediario entre las capas de *presentación* y de *datos*. Será la encargada en albergar todo el modelo de clases, definiendo el comportamiento de los distintos clientes. Esta capa estará ubicada en el servidor.

- **Capa de datos:** Última capa de la arquitectura, incluye la Base de Datos (BBDD) con toda la información compartida con los usuarios del sistema. Se comunicará la capa de *negocio* con ella, para solicitar el almacenamiento de datos nuevos u obtener la información asociada a un recurso, entre otras posibles acciones. Estará ubicada en la misma máquina que la capa de *negocio*, de esta forma se consigue establecer entre ambas un intercambio rápido de datos sin necesidad de utilizar ningún protocolo de comunicación.

4.3. Servicio Web

Pensada la arquitectura sobre la que se desarrollará el sistema, a continuación, se detalla el diseño de los diferentes módulos que colaborarán entre sí para el correcto funcionamiento del servidor.

4.3.1. Integración con LibreGeoSocial

Para el completo desarrollo de este sistema, el servicio web implementado se ha incluido en el *framework* de la red social móvil LibreGeoSocial, explicado en la sección 2.6. La integración de una nueva aplicación en esta red sócial es muy sencilla como si de un plug-in se tratase, tan solo es necesario copiar en la carpeta "apps/" del árbol de directorio de LGS la carpeta con los diferentes ficheros que definen el funcionamiento y comportamiento de nuestro sistema, es decir, nuestro proyecto de Django. Con esta integración se benefician ambas partes, tanto al *framework* como el servicio integrado, ya que el primero ve ampliadas sus aplicaciones así como sus funciones y el segundo puede reutilizar todas las funcionalidades para el desarrollo de otras nuevas e incluso el uso del modelo de datos para su desarrollo.

4.3.2. Modelo de datos

El diseño de la base de datos utilizado en el Proyecto Fin de Carrera será una extensión del utilizado en el *framework* LibreGeoSocial. Con el análisis y utilización del modelo de la red social conseguimos brindar a nuestro diseño con recursos geolocalizados (equipo, calle) y otras características, gracias a la herencia de elementos del modelo proporcionado por la red social (ver figura 4.4).

A continuación detallamos los principales elementos que describen el diseño de la base de datos utilizados en nuestro sistema, explicando su estructura, relaciones, operaciones, etc.

Equipo: Encargada de almacenar la información de los equipos participantes, utilizados en el cliente Android. Esta clase tendrá una referencia a *Group* de la base de datos de LibreGeoSocial, posibilitando la asignación de un nombre y características de su geolocalización. También tendrá la referencia al juego al que está asociado, ya que un equipo solo podrá participar en uno.

Organizador: Contendrá la información relativa al encargado de administrar los diferentes elementos del servicio web. Poseerá una referencia a la tabla *Person* perteneciente a LibreGeoSocial, en la cual se le darán los privilegios pertinentes para realizar sus funciones. También se tendrá una referencia a todas las partidas que han sido creadas por este.

Patrimonio: Toda la información relativa al valor del patrimonio, dinero y números de propiedades serán almacenados junto a una referencia al *Equipo* que pertenecen. Estos datos serán utilizados en el marcador, mostrado tanto en el seguimiento de la web como en la clasificación de la aplicación Android.

Carta: Contiene la información asociada a cada carta creada por el Organizador para todas las partidas. Todas estarán compuestas por un nombre, descripción, tipo de acción y un campo booleano que indica si ha sido usada. Dependiendo de la acción que realicen poseerán información referente a la cantidad económica que afectará al equipo propietario.

Juego: Contendrá las diferentes partidas creadas por los diferentes Organizadores. Contiene el nombre, la ubicación, fecha de inicio, cantidad de dinero inicial y duración total de la partida. También contará con un campo que indicará el estado de la partida, es decir, si la partida está abierta y puede ser jugada, o por el contrario está cerrada y no puede ser jugada ni modificada por ellos.

Calle: Guarda la información relacionada a las calles creadas en el sistema. Este elemento heredará de la clase *Social_node*, lo que le proporciona al recurso características para su geolocalización. Contendrá la referencia al *Juego* en el que se ha creado y al *Equipo* propietario, en caso de que no tenga propietario este último campo será nulo. También se reúne los datos relacionados con la calle como pueden ser su nombre, precio de compra, si posee una edificación, si ha sido saboteado y por último la versión. Este último campo será usado para el control de la concurrencia.

Edificio: Se almacenará la información de las edificaciones que se podrán realizar, para ello, se hace una referencia a la *Calle* al que está asociada. Además se tendrá información relativa al edificio como puede ser su nombre, bonificación otorgada temporalmente y precio de construcción.

Mensaje: El sistema poseerá un servicio de mensajería, así que es necesario almacenar los diferentes mensajes intercambiados entre los jugadores y el Organizador. Cada

mensaje contendrá una referencia al emisor (*Equipo / Organizador*) y destinatario (*Equipo / Organizador*), también se guardará el texto del mensaje enviado. Está información será mostrada a los usuarios a través de la interfaz de los diferentes clientes del sistema.

Oferta: Además del servicio de mensajería, también se ha diseñado un servicio de ofertas. Contendrá las referencias al propietario, la calle y el ofertante. Además se almacenarán otros datos como pueden ser la cantidad de dinero ofrecida, si ha sido respondida y por último si fue aceptada o rechazada. Esta información será utilizada únicamente por el cliente Android.

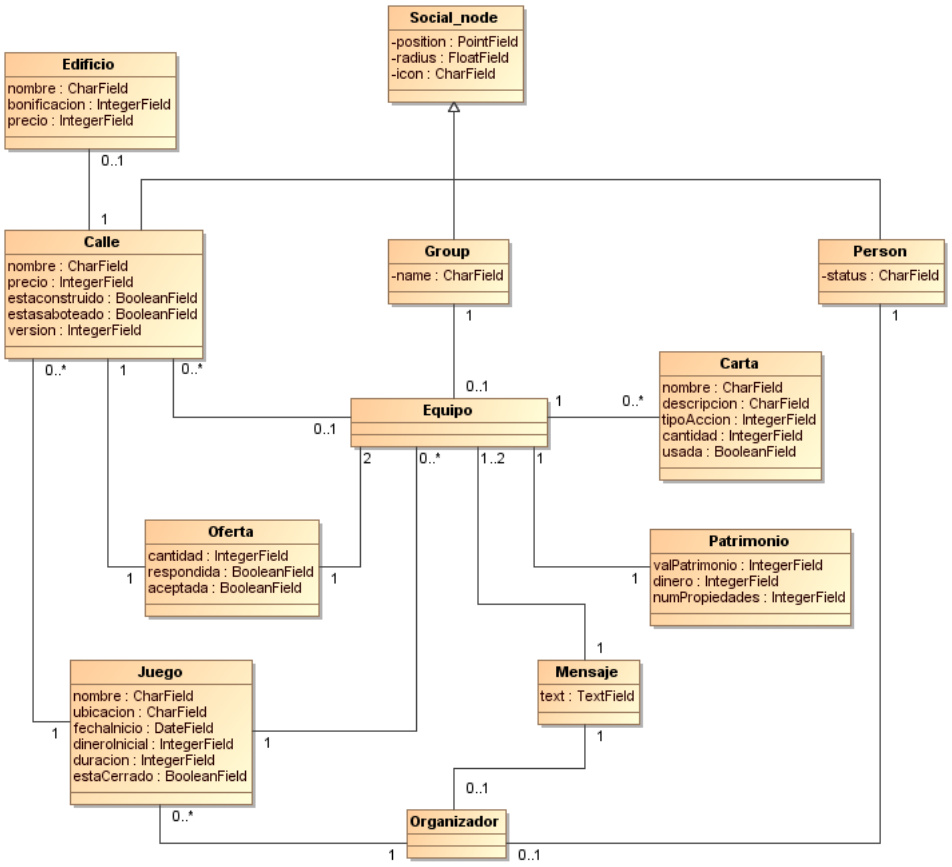


Figura 4.4: Diseño del modelo de datos utilizado en el sistema

4.3.3. Patrón MVC

El *framework* web elegido para el desarrollo del sistema será Django. Como se ha comentado en la sección 2.3, esta herramienta se basa en el patrón de arquitectura

software *Model View Controller (MVC)* [15] pero con diferencias, ya que sus desarrolladores hacen una interpretación diferentes de este patrón, y lo definen como *Model Template View (MTV)*. Se puede observar de forma gráfica el patrón utilizado en la figura 4.5.

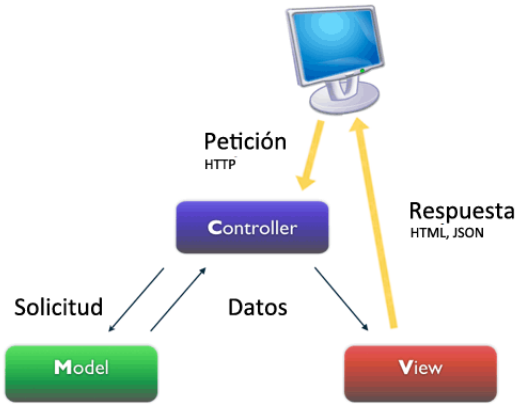


Figura 4.5: Diagrama del diseño del patrón *Model-View-Controller* (MVC)

La utilización de este patrón junto a la arquitectura elegida (tres capas) facilitan la evolución en paralelo de la lógica de negocio y la interfaz de usuario, también permite incrementar su reutilización y flexibilidad en otros sistemas. Destacar que la separación de estos aspectos en una aplicación da mucha flexibilidad al desarrollador. A continuación se presenta una descripción de los componentes que conforman el patrón y las relaciones que tiene con el *framework* Django que hemos elegido para el desarrollo de nuestro sistema.

Model

Es la representación a modo de objetos de las tablas de la base de datos que componen el sistema, siendo la base del funcionamiento. En Django queda definido a través del fichero *models.py* incluido en el servicio web, en el cual, se definirán las diferentes tablas con sus respectivos campos, restricciones, etc., para que posteriormente sean creadas en la base de datos. Este componente interactuará únicamente con el nivel superior (*Controller*).

View

Este componente corresponde a la correcta representación del modelo de datos en la interfaz, que el usuario ha solicitado a través de una URI (Identificador Uniforme

de Recurso) en particular, ya sea mediante una interrelación con la web o escribiendo la URI directamente. En Django este componente se organiza en diferentes ficheros.

Para la creación de las diferentes URIs se ha optado por seguir el estilo arquitectural REST,definido en el fichero *urls.py*, en el cual, se describen las diferentes URLs que conforman el sistema y que el usuario utilizará para cambiar de interfaz y realizar diversas peticiones. En este PFC se recoge en la carpeta "templates/" la representación de los diferentes modelos de datos que el sistema maneja y serán mostrados en la interfaz para la interacción con los usuarios. Al disponer de dos clientes en nuestro sistema, se dispondrán de diferentes ficheros para representar la información, HTML destinados para el cliente web y JSON para el cliente Android.

Controller

El controlador se encarga de manipular el modelo de datos, capturando las diferentes peticiones que realiza el cliente en la interfaz. En el *framework* utilizado se corresponde con la carpeta "Core/" de la aplicación, en esta carpeta se creará una librería por cada tabla utilizada, en estos ficheros se definirán las operaciones de obtención y manipulación de los datos. Cada URI tendrá asociada una función, estas ubicadas en el fichero *views.py*, será el encargado de enlazar las diferentes librerías para la manipulación de los datos y las peticiones realizadas por los clientes. Una vez atendida enviará la respuesta para la generación del documento resultante visualizado en la capa *View* por el cliente.

4.3.4. Estilo de arquitectura REST

Como se comento anteriormente, Django define las diferentes URIs del sistema a las que el usuario puede acceder y realizar las diferentes peticiones, siguiendo el estilo arquitectónico REST (*REpresentational State Transfer*) introducido por Roy Fielding. La utilización de ese estilo resume como los recursos son definidos en un servicio web.

En un servicio REST [33] típico, tenemos una URL identificando únicamente a cada recurso (carta, partida, etc.) que se administra, y que realiza una determinada acción sobre dicho recurso a través de los métodos *GET*, *POST*, *PUT* y *DELETE*, definidos en el estándar HTTP. Cualquier recurso identificado no debe de aportar ninguna información de su implementación. En este proyecto el servidor solo dará soporte a los métodos *GET* y *POST* dando posibilidad a todas las operaciones, esto es debido a que los navegadores habitualmente sólo soportan estos métodos. El método *GET* será usado

únicamente para obtener información de un recurso determinado, en cambio el método *POST* será utilizado para realizar la manipulación de los recursos, incluyendo su eliminación o creación.

La definición de las diferentes URIs utilizadas en este proyecto sigue el principio de REST, cada petición que realice el cliente al servidor debe ser sin estado, es decir, la petición realizada debe de contener toda la información necesaria para su ejecución en el servidor, sin recurrir a información de peticiones anteriores. Se deben de utilizar nombres y no verbos para identificar sus recursos, también cada recurso tendrá asociado un identificador único. Al final de la URI se utilizará un verbo con la acción que el servidor debe de realizar, esto es necesario debido a la utilización únicamente del método *POST* para realizar la manipulación de los recursos. A continuación se muestra un ejemplo del diseño de las URIs utilizadas en este proyecto.

/goldenstreet/recurso[/id]*[/recurso[/id]*]+/acción/
    /goldenstreet/juego/1/equipo/listar/
    /goldenstreet/juego/2/equipo/listar/
    /goldenstreet/juego/1/equipo/2/mostrar/
    /goldenstreet/juego/1/equipo/2/modificar/
    /goldenstreet/juego/1/equipo/2/borrar/
    /goldenstreet/juego/1/equipo/2/posicionar/

La utilización del estilo arquitectónico REST en el diseño de este proyecto, aporta numerosas ventajas al desarrollo del servicio web:

- El proyecto es más fácil de mantener, las URIs utilizadas son más estructuradas y claras para el usuario.

- Se consigue una mayor estabilidad frente a futuros cambios, ya que permite la evolución independiente de los documentos y la creación de nuevos no afecta a los anteriores.

- Al permitir la definición de un sistema por capas aumenta su rendimiento.

- Se consigue una mayor escalabilidad en el servicio web, gracias a que las peticiones realizadas por los clientes son sin estado.

4.4. Cliente Android

En el sistema propuesto hay dos tipos de clientes, el cliente web enfocado a los administradores del juego y el cliente Android destinado a los participantes. Nos centraremos en el diseño del cliente Android, ya que el cliente web será únicamente una serie de páginas HTML, accedidas mediante el navegador web, y está muy ligado con el diseño del servidor. En cambio el diseño del cliente Android posee una mayor complejidad.

4.4.1. Uso de LibreGeoSocial

Como se comentó anteriormente en el apartado del diseño del servidor, el cliente Android será totalmente independiente del módulo LibreGeoSocialApp, copiando el proyecto con la aplicación diseñada en un paquete a modo de plug-in, consiguiendo un fácil mantenimiento. Esta unión se ha realizado principalmente para beneficiarnos de las funcionalidades ya desarrolladas que ofrece para la implementación de otras nuevas o para su utilización.

4.4.2. Modelo de clases

A través del diagrama de clases, que se puede ver en la figura 4.6, se permite describir la estructura mostrando las diferentes clases, atributos y relaciones existentes entre ellos, que intervendrán en la solución. Para una mayor claridad y comprensión se explica a continuación la función de las principales clases mostradas en el modelo, para obtener una mayor información en el código fuente de la aplicación Android, se ha incluido una cabecera en cada clase explicándola detalladamente.

Goldenstreets: Muestra la pantalla de bienvenida a la aplicación. Si el usuario no dispone de conexión a internet y GPS, se le informará de esta necesidad para participar.

JuegoListar: Muestra el listado de los juegos disponibles en la que el usuario puede jugar, almacenando los datos del juego seleccionado para su posterior utilización.

JuegoDatosListar: Muestra el listado de los equipos disponibles con los que el usuario puede participar, almacenando la información del equipo seleccionado y datos relevantes del juego (calle, edificios, etc.).

PartidaInfo: Muestra información sobre el juego y equipos seleccionados, y da la bienvenida a la partida si los datos son aceptados por el usuario.

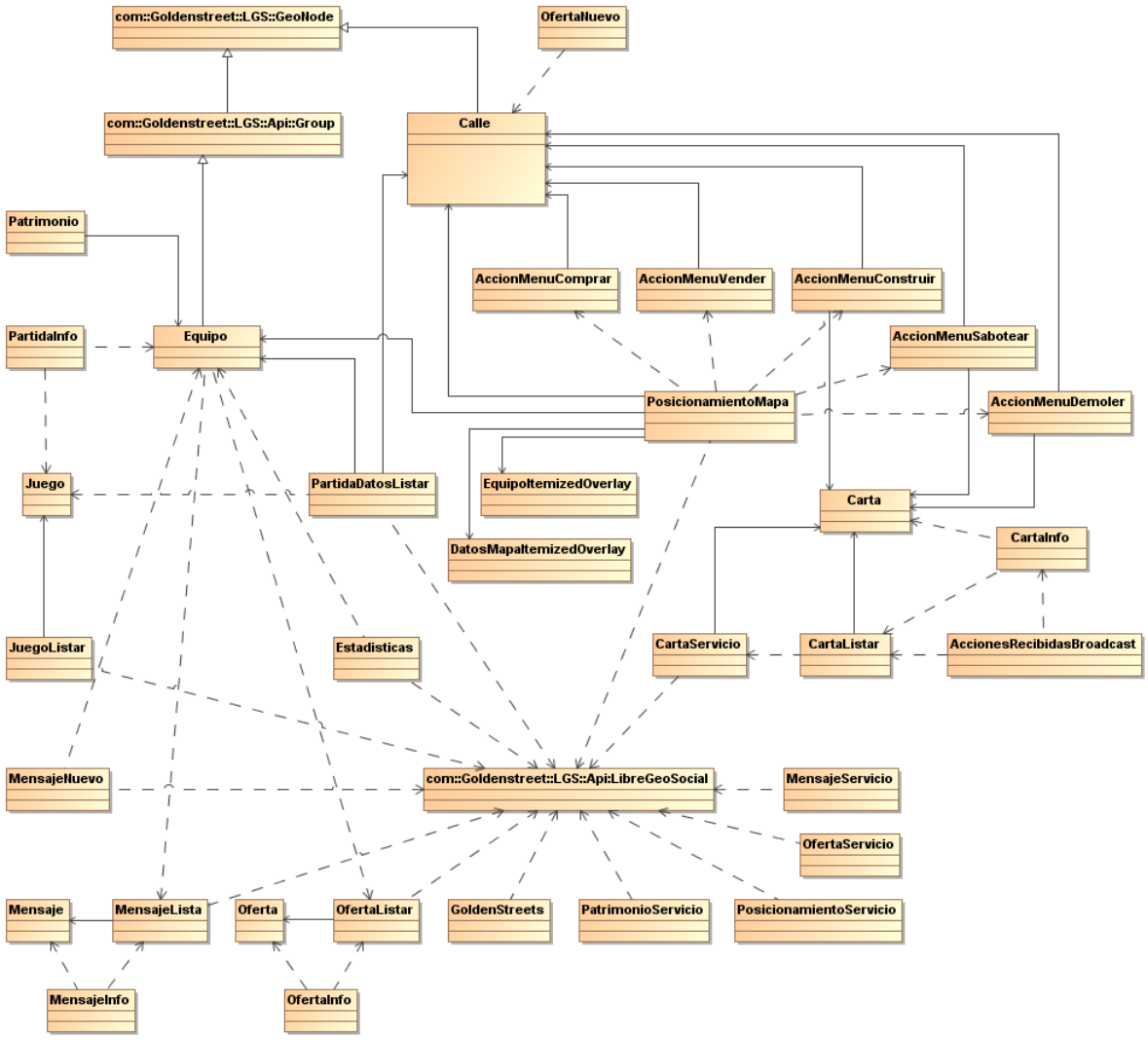


Figura 4.6: Diagrama del modelo de clases de la aplicación Android

Estadisticas: Muestra al equipo el tiempo que lleva jugando, la información sobre su patrimonio y una clasificación de todos los equipos participantes.

CartaInfo, CartaListar: Clases encargadas de mostrar la información detallada y el listado de las cartas que va recibiendo periódicamente el equipo.

PosicionamientoMapa: Muestra un mapa con la posición de todos los recursos geolocalizados que forman parte de la partida (equipos y calles), cada recurso estará representado por diferentes iconos, según su estado.

OfertaMenu, OfertaInfo, OfertaListar, OfertaNuevo: Clases encargadas de mostrar los datos del sistema de ofertas diseñado. Visualiza el menú, la información detallada y el listado de las ofertas recibidas y enviadas por el equipo.

MensajeMenu, MensajeInfo, MensajeListar, MensajeNuevo: Clases destinadas al sistema de mensajería diseñado. Mostrando el menú, la información detallada y el listado de mensajes. También se le permitirá al equipo el envío de nuevos mensajes a otros participantes o al Organizador.

AccionMenuComprar, AccionMenuConstruir, AccionMenuDemoler, AccionMenuSabotear, AccionMenuVender: Clases encargadas de realizar las acciones (comprar, construir, demoler, sabotear y vender) que el equipo realiza en el mapa sobre sus propiedades o las de sus rivales.

PatrimonioServicio, CartaServicio, PosicionamientoServicio, Oferta Servicio, MensajeServicio: Clases destinadas al control de los servicios ejecutados en background, estableciendo comunicación con el servidor para la manipulación u obtención de datos.

AccionesRecibidasBroadcast: Clase auxiliar para tratar la recepción de las acciones enviadas por el servicio de cartas ejecutado en segundo plano.

Juego, Calle, Equipo, Carta, Mensaje, Oferta, Patrimonio: Almacena en la aplicación parte de la información del modelo de datos del servidor.

Capítulo 5

Implementación

Este capítulo se dedicará al estudio de la implementación del servidor y clientes expuestos en el capítulo 4, dedicado al diseño del proyecto. Como recordatorio, debido a las necesidades del sistema se ha utilizado una arquitectura Cliente-Servidor.

El servidor ha sido finalmente implementado a través del *framework* de desarrollo web Django y una base de datos PostgreSQL. La elección de estas herramientas se fundamenta en los principales beneficios que aportan explicados en el capítulo 2 (estado del arte), y debido a la inclusión de nuestra aplicación en el proyecto *LibreGeoSocial* que emplea el mismo *framework* de desarrollo y base de datos. En el caso haber utilizado otra herramienta, habría sido necesario renunciar a la integración de la aplicación desarrollada en el proyecto LibreGeoSocial.

Por otro lado el cliente principal del sistema, será implementado en una aplicación Android soportada por los terminales de dicho sistema operativo. Esta será la encargada de establecer la comunicación con el servidor a través de una conexión a internet, ya sea mediante wi-fi o red móvil. Se utilizarán las librerías proporcionadas por el proyecto LibreGeoSocialApp para realizar las diferentes comunicaciones para la obtención de datos o modificación de los mismos. También se puede considerar como cliente, la interfaz web desarrollada en el lado del servidor para el acceso de los Organizadores. En ella, se podrá realizar diferentes acciones, explicadas en la subapartado 5.1.1, para la administración y monitorización de los diferentes componentes de una partida.

5.1. Servicio Web

En este apartado se detallará la implementación realizada del servidor para la posible creación de partidas y administración de las mismas por parte del Organizador. Para el desarrollo de este, se ha utilizado la base de datos PostgreSQL que proporciona soporte para la geolocalización de los diferentes recursos utilizados (equipos y calles), gracias a la extensión o módulo PostGis [27] se permite el uso de objetos GIS (*Geographic information systems*), almacenando las coordenadas geográficas para su utilización por parte de los clientes, como puede ser en la monitorización del cliente web o en el mapa de la aplicación Android. Para la obtención de un mayor rendimiento y por la recomendación propia de Django, se descartó el uso del servidor que este posee y se decidió utilizar el servidor web HTTP de código abierto Apache, su utilización ha sido posible gracias al uso del módulo mod_wsgi [24], con el cual se consigue enlazar el lenguaje de programación Python empleado en Django y el servidor web Apache.

La posibilidad de tener varios clientes accediendo de forma simultánea a la BBDD ya sea mediante la aplicación móvil o mediante la interfaz web, puede ocasionar problemas de concurrencia [26], por lo que es necesario su control para evitar el incorrecto funcionamiento del sistema. La solución tomada para controlar este problema ha sido la implementación del control de concurrencia optimista en Django, descartando otras opciones como el uso de cerrojos.

La técnica de control de concurrencia optimista, también denominada de validación o certificación, no realiza comprobaciones mientras la operación es llevada a cabo, es decir, retrasan la validación justo antes de realizar la escritura en el campo correspondiente. De esta manera, una petición al servidor con este tipo de control no va a ser retrasada, pero si puede ser rechazada. Para realizarla con éxito es necesario hacer todas las comprobaciones de forma atómica, es decir, procesar simultáneamente la lectura y escritura final de los datos. Para la implementación de esta técnica se ha añadido un campo en las tablas susceptibles a sufrir este problema, como puede ser la tabla *Calle*, también ha sido necesario utilizar operaciones atómicas (realizan la lectura y escritura del dato sin ser interrumpidas) en la base de datos, para ello se han utilizado el objeto *F()* y la operación *Update()* proporcionados por el *framework* Django. En el caso que se produzca un problema de concurrencia entre varios usuarios que realizan la misma petición al servidor, este verificará y realizará la primera petición recibida y rechazará las del resto de los usuarios, enviando la respuesta a sus respectivos clientes. Se ha optado por esta técnica frente a otras, porque se puede implementar de forma

sencilla sin necesidad de utilizar librerías externas, únicamente añadiendo un nuevo campo a las tablas afectadas llamado *version* y realizando las comprobaciones cuando se efectúan las modificaciones que el cliente ha solicitado. Otro motivo e importante para el correcto funcionamiento de esta técnica, es la poca frecuencia de concurrencia sobre el mismo elemento por parte de diferentes equipos en el transcurso de una partida.

Las comunicaciones siempre las iniciará el cliente, ya que se trata de una arquitectura Cliente-Servidor. La información transmitida entre los diferentes extremos dependerá del cliente que inicie la comunicación, en el caso del navegador web siempre se realizará mediante HTML, ya que se trata de una página web. Si es el cliente Android el que comienza la conexión se enviará la respuesta en formato JSON, aunque se podría haber utilizado otro lenguaje como XML, pero se ha utilizar el primero por las ventajas que aporta, comentadas en el capítulo 2. Para la apariencia de las páginas web en formato HTML se ha utilizado la hoja de estilos CSS (*Cascading Style Sheets*) [35], definiendo la forma en la que se visualizan los múltiples elementos utilizados en los diferentes documentos HTML. Debido a la utilización de este mecanismo se puede modificar completamente la apariencia de las páginas web sin necesidad de modificar el código HTML. Mencionar que la hoja de estilos ha sido validada satisfactoriamente y desarrollada siguiendo las especificaciones técnicas y directrices de la comunidad internacional W3C (*World Wide Web Consortium*) [34].

5.1.1. Interfaz web: Acciones del juego

Al acceder a la aplicación web, ya sea mediante el navegador web de un ordenador o un móvil, se requerirá el nombre de usuario y la contraseña para la autenticación (ver figura 5.1). En el caso de que el usuario se autentique correctamente pero no sea el Organizador de la aplicación *GoldenStreets*, no podrá realizar ninguna acción (crear, modificar y eliminar), pero si podrá visualizar los diferentes apartados del sitio web, por ejemplo, las partidas creadas o equipos participantes. En el caso de que los datos introducidos en el servidor para la validación del Organizador sean erróneos, se redirecciona al usuario a una página de error informándole que el nombre y/o contraseña no son válidos. También será utilizada la página de error para informar al usuario de cualquier posible fallo detectado durante la navegación o administración de los diferentes recursos del sitio web.

Una vez se ha verificado que los datos de entrada corresponden con el Organizador

Figura 5.1: Pantalla con el formulario de autenticación del servicio web

de la aplicación *GoldenStreets*, es redirigido a la pantalla de bienvenida, donde tendrá a su disposición todas las funciones para la creación y administración completa del sitio web.

Cabe destacar como norma general que cada vez que se visualiza un formulario en el sitio web, este es solicitado al servidor mediante una petición *GET*, una vez correctamente cumplimentado por el Organizador este será enviado a través de una petición *POST* a la misma URI, también se realizará la acción "Borrar" con el mismo tipo de petición, cuando el Organizador lo solicite desde el sitio web.

Administrar cartas

Cuando el Organizador se ha autenticado, podrá administrar (crear, modificar y eliminar) las cartas que se utilizarán en los juegos. Una carta será un elemento independiente y común para todos los juegos creados en el servidor, es decir, eliminar una carta o todas no imposibilita la partida, se juega sin cartas o con las que estén creadas. Si el Organizador desea administrar este elemento, en la esquina superior izquierda se sitúa un menú horizontal con las diferentes opciones que el usuario puede administrar, y entre ellas la opción "Cartas". Una vez seleccionada esta opción se le mostrará una tabla con el listado de las cartas disponibles, como se observa en la figura 5.2, en las que se mostrará un breve resumen de su información, compuesto por el identificador, el nombre, la acción y las opciones disponibles para cada carta.

Una de las acciones que podrá efectuar el Organizador, será crear una nueva carta pulsando para ello el botón "Crear Carta" situado debajo de la tabla con el listado.

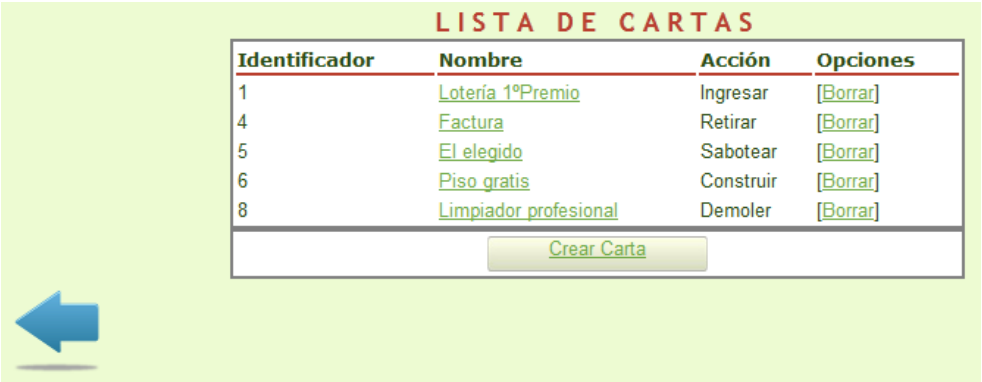


Figura 5.2: Pantalla con el listado de las cartas creadas

Una vez que el usuario seleccione esta opción, se le redirecciona a una nueva página, en ella, el usuario podrá introducir los datos (nombre, tipo, cantidad y descripción) para la nueva carta que se desea crear a través del cumplimentación y envío de un formulario HTML (ver figura 5.3) al servidor. Los tipos de cartas que el Organizador puede crear son los siguientes:

Ingresar: Tipo de carta con la cual se procederá a incrementar el valor del patrimonio del equipo beneficiado, ingresando la cantidad de dinero que se indicó en el formulario de su creación.

Retirar: Carta perjudicial para el equipo poseedor, al cual se le retirará la cantidad de dinero que se ingreso en el formulario de creación, reduciendo su valor de patrimonio y sus posibilidades de victoria.

Sabotear: Carta destinada a promover el intercambio de propiedades entre los diferentes equipos en el transcurso de una partida, la utilización de esta ocasionará en el equipo afectado gastos periódicos en relación a la propiedad sobre la que se ha aplicado y el valor del patrimonio se verá reducido.

Construir: Beneficiosa para el equipo poseedor del tipo de esta carta, con la cual se podrá construir la edificación en una calle en posesión del equipo, sin necesidad de gastarse nada.

Demoler: Carta con doble funcionalidad según el equipo lo requiera. Útil para lastrar a tus rivales destruyéndole la edificación, reduciendo su valor de patrimonio y los ingresos que aporta la calle afectada. También es recomendable utilizarla cuando

al equipo poseedor le hayan saboteado una calle, evitando así los gastos que ocasionan periódicamente esa propiedad.

Cabe mencionar que cuando se deja en este formulario o en cualquier otro del sitio web un campo en blanco o se introduce un valor incorrecto, por ejemplo, en el campo "Cantidad" no se introduce un valor numérico, no es posible la creación de una carta nueva y se alerta al usuario. El aviso es mostrado en el propio formulario,, indicando la celda y el motivo del error como se observa en la figura 5.3. Esto es posible gracias a la librería JavaScript de código abierto LiveValidation que utiliza tecnología AJAX. Esta librería valida en el lado del cliente el formulario sin necesidad de refrescar la página, evitando así enviar la información incompleta o errónea al servidor, aunque por cada petición este también verificará que la información es correcta para evitar posibles errores en el sistema. Los motivos de la utilización de esta librería son, cómo se menciona anteriormente, no enviar información incompleta al servidor para su comprobación y también algo bastante importante, poder ofrecer al usuario una interfaz amigable informándole de los errores que se han producido al intentar enviar un formulario. Cuando es validada toda la información del formulario para la creación de la nueva carta, se envía una petición *POST* al servidor con los datos. Este volverá a verificar que los datos recibidos son correctos y se realizarán las modificaciones pertinentes en la base de datos, ya sea la creación de un nuevo, la modificación o borrado de un elemento, si todo está correctamente. En caso de producirse un error se lo comunicará al cliente web, el cual mostrará dicho error al usuario.

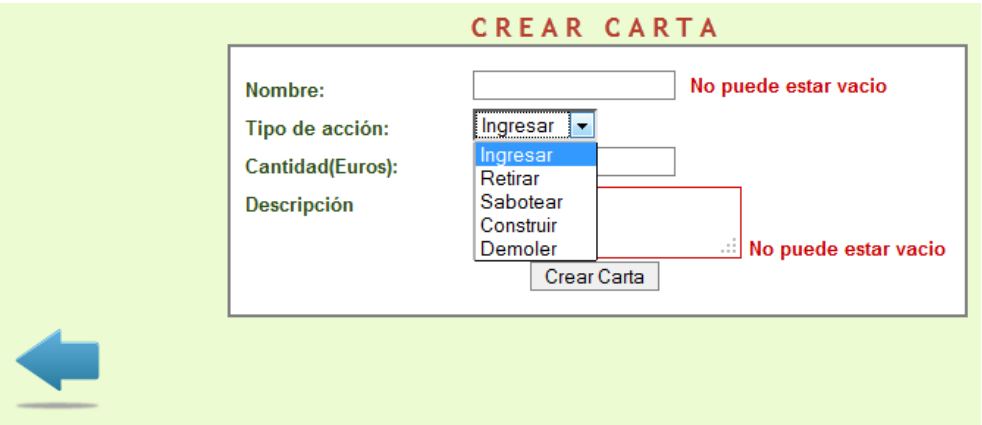


Figura 5.3: Pantalla con la validación y el formulario de creación de una carta

Al crear la nueva carta o al seleccionar una carta del listado en la interfaz web se redirigirá a la página con la información detallada de la carta (véase la figura 5.4), en

la cual se podrá ver toda su información. En esta interfaz también se le da al usuario una nueva opción, la de modificar la información referente a la carta visionada. En el caso que el usuario decida cambiar algún valor deberá pulsar el botón destinado a ello "Modificar carta", con el cual se abrirá una nueva interfaz con el mismo formulario de creación (nombre, tipo, cantidad y descripción), pero en este caso los campos contendrán los valores actuales de la carta. El proceso de modificación se realiza de forma similar al de creación, primero se valida en la interfaz web y posteriormente en el servidor Django que realizará la acción solicitada si las verificaciones fueros correctas.

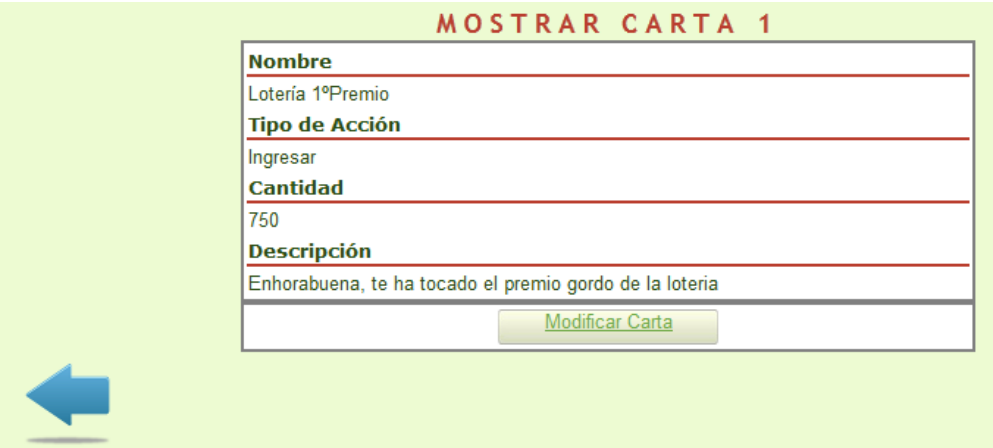


Figura 5.4: Pantalla con la información detallada de la carta

La última opción disponible para el usuario referente a las cartas es su borrado, para ello el Organizador deberá desplazarse a la página con el listado de todas las cartas de la aplicación *GoldenStreets*, en su columna "Opciones" se dispondrá a disposición del Organizador la acción de "Borrar", con la que conseguimos eliminar completamente de la base de datos la carta seleccionada.

Administrar juegos

Otra opción disponible en el menú horizontal del sitio web es "Juegos", con la que el Organizador podrá administrar todos los juegos y todos sus elementos asociados. La primera página que se visualizará al seleccionar esta opción, es la tabla con el listado de todas las partidas creadas, se visualizará el identificador, nombre y estado, en esta tabla también se pondrá a disposición del Organizador dos posibles acciones. A través de la columna "Opciones" se podrá cambiar el estado de la partida, es decir, se podrá cerrar o abrir según crea conveniente el usuario, esta acción ha sido implementada para evitar

que un grupo de jugadores entren en alguna partida en la que otro grupo de jugadores ya ha jugado. Otra de las opciones disponibles en el listado de juegos es la de "Borrar", con la que se podrá eliminar la partida asociada a la fila en la que se seleccione esta acción. Cuando se pretende eliminar de la base de datos, no solo se eliminan sus datos sino que también todo lo relacionado con ella, es decir, se eliminan las calles, edificios, equipos, etc.

También en esta página se le dará la opción al usuario de crear una nuevo con todos sus elementos, para ello, deberá de seleccionar el botón en la parte inferior destinado a ello "Crear Juego", una vez seleccionado, será redirigido a una nueva página (ver figura 5.5). En ella, el usuario tendrá que introducir los datos (nombre, ubicación, fecha inicio, dinero inicial, duración) del juego que desea crear a través del cumplimiento y envío del formulario HTML al servidor Django. Cuando se proceda a su envío como con otro cualquier formulario se validará primero en el lado del cliente y posteriormente en el servidor, donde se realizará la acción solicitada si se verifica todo correctamente.

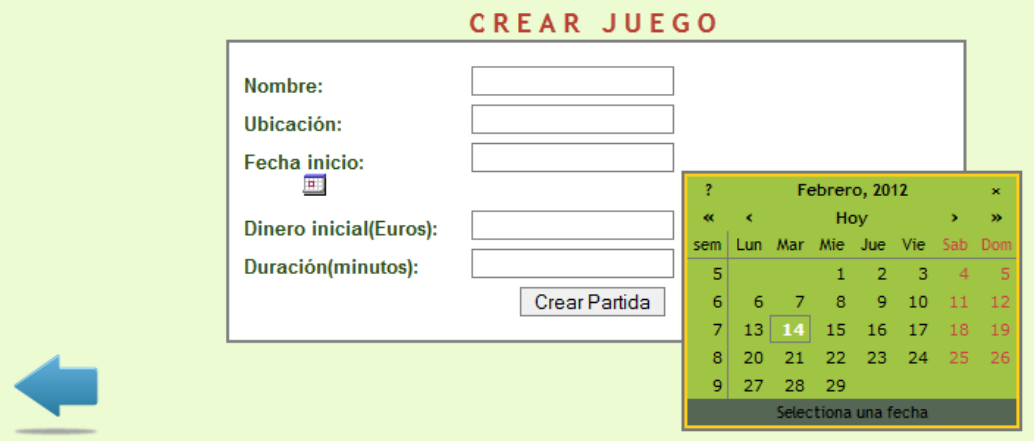


Figura 5.5: Pantalla con el formulario de creación de un juego y el calendario

Por último al crear un nuevo juego o al seleccionarlo del listado se redirige al usuario a una nueva página con los detalles y la administración del elemento elegido, denominada "Mostrar juego", como se observa en la figura 5.6 se podrá obtener información de todos sus datos asociados, por ejemplo, nombre, ubicación, estado, etc. En esta también se pondrá a disposición del usuario diferentes acciones para la administración de todos sus recursos, algunas de ellas ya explicadas en este subapartado o en subapartados anteriores, como puede ser, "Borrar juego", "Cambiar estado" y "Listar cartas", otras de las acciones explicadas y detalladas, debido a su complejidad, en subapartados posteriores son, "Listar equipos" y "Listar calles".

Figura 5.6: Pantalla con la información detallada del juego

Por último otra opción ofrecida en la parte inferior del listado es "Modificar Juego", con el cual se podrá cambiar o corregir cualquier valor del juego a través de la modificación de un formulario, similar al de la creación de un juego, pero el valor de los campos estará rellenado con los datos actuales del juego. Mencionar que tanto en el formulario para la creación de un juego como en la modificación del mismo se ha utilizado un Script con el que se facilita al usuario introducir la fecha, evitando así posibles errores en su formato.

Administrar equipos

Cada juego tendrá su grupo de equipos independiente al resto de las partidas, para poder administrar todos los equipos asociados y sus elementos es necesario acceder a su listado pulsando el botón "Listar equipos", botón situado en la parte inferior de la tabla con la información relativa al juego. Una vez accedida a la página con al información, el usuario visualizará una tabla con el identificador, nombre y opciones disponibles. Para obtener más información de un determinado equipo es necesario pulsar sobre su nombre, redirigiendo al usuario a una nueva página. Relacionado con las opciones mostradas sobre este recurso, solo se podrá realizar su borrado, ya que otras opciones, como "Cambiar estado", no tienen sentido. Otra opción disponible, situada en la parte inferior, es la creación de nuevos equipos para la partida.

Inicialmente una partida no tendrá en su listado ningún equipo creado, así que el primer paso que el Organizador deberá realizar, es pulsar el botón "Crear equipo" visualizado en la parte inferior de la página. Una vez se haya pulsado el botón, se le redirecciona a una nueva página con el formulario de creación de equipos. En este formulario se permitirá únicamente introducir el nombre del equipo que queremos asociar a esta partida, que será visualizado en el cliente móvil y le identificará. En este caso se ha utilizado el script de validación para evitar que el usuario introduzca palabras superiores a 8 carácteres, para su posterior envío al servidor. La creación de un nuevo equipo implica la creación de un grupo en LibreGeoSocial, ya que va a ser un recurso geolocalizado. Otro detalle a tener en cuenta en la creación, es que si el servidor detecta que ya existe el nombre del grupo que deseamos asociar a la partida en la base de datos, este no podrá ser creado y como cualquier otro fallo del sitio web se redireccionará al usuario a la página de error para ser informado.

Realizada la creación con éxito o seleccionando del listado un equipo, se redirige al Organizador a una nueva pantalla "Mostrar equipo" (ver figura 5.7), en la que se visualizará su nombre pero también se dará la acción de modificarlo. Si el usuario decide utilizar la acción mencionada, como en los casos anteriores y posteriores, se redirigirá a un formulario similar al de la creación del equipo, pero en el campo del nombre su valor actual. Otra de las opciones disponibles en la pantalla con la información del equipo, es la posibilidad de visualizar en una nueva página el patrimonio que posee actualmente, en esta página solo se mostrará la información (dinero, valor patrimonio, número de propiedades) sin que el usuario pueda interactuar con ella, ya que el cliente Android será el encargado de modificar estos valores perteneciente al equipo, en el transcurso de una partida.

Figura 5.7: Pantalla con la información detallada del equipo y las calles en su posesión

Si el Organizador decide eliminar un equipo perteneciente a una partida, deberá dirigirse a la página con el listado de ellos, en la cual podrá seleccionar en la fila la opción "Borrar" de la columna "Opciones". La eliminación del equipo implica su borrado en la base de datos como grupo perteneciente a LibreGeoSocial, y también la eliminación de todos sus recursos asociados, como puede ser el patrimonio.

Administrar calles

Un elemento indispensable para poder participar en un juego son las calles, ya que sin ellas un equipo no podrá comenzar la partida en el cliente Android. Al estar este recurso asociado a un juego, será necesario acceder en el sitio web a la página "Mostrar juego" en el que se desee administrar, y seleccionar la opción "Listar calles" situada en la parte inferior, junto a las demás.

La administración de todas las calles asociadas a un juego comienza con su listado en la página del sitio web "Lista de calles", en el cual se mostrará su identificador, nombre y posibles opciones a efectuar (en este caso solo borrar). Se podrá ver los detalles de una con tan solo pulsar sobre su nombre, también será posible borrarlas seleccionado la opción habilitada para ello en su fila y por último el Organizador podrá crear esta pulsando para ello el botón "Crear calle" localizado en la parte inferior de la tabla.

Si se decide crear una calle se accederá a una nueva página, en ella, como en los anteriores casos en los que se creaban los recursos, se mostrará el formulario con los campos nombre, ubicación (compuesta por latitud y longitud) y precio. Además en esta página se pondrá a disposición del usuario un mapa, gracias a la utilización de la API de Google Maps [9], con el cual el usuario podrá interactuar para determinar la posición exacta. Al pulsar sobre el lugar en el que desea posicionar la calle se colocará un icono, además en el formulario se rellenarán los campos destinados a la ubicación con los valores de la latitud y longitud extraídos del mapa. En el caso que no se desee utilizar el mapa porque ya se conozcan las coordenadas del lugar, las puede introducir manualmente en el formulario. Una vez rellenado todos los campos se validará y se procederá a su envío al servidor para crearla. Señalar finalmente que el recurso calle deberá de ser geolocalizado, similar al equipo, ya que se necesitarán almacenar en el servidor sus coordenadas geográficas. Para conseguir este propósito se reutiliza la clase *Social_node* la librería de LibreGeoSocial.

Cuando se ha creado correctamente o seleccionado en el listado la calle, se visualizará en la página una tabla con toda su información (nombre, latitud, longitud y

precio). En esta pantalla se pone a disposición del usuario la posibilidad de modificarla o de administrar el listado de sus edificios (explicado en el siguiente subapartado). Si se detecta algún error en los valores, se puede pulsa el botón "Modificar calle", con el cual, se le redirecciona a un formulario similar al de la creación, con la salvedad que los campos ya estarán cumplimentados, como se aprecia en la figura 5.8. También se visualizará en la parte superior un mapa, en este se localizará marcada mediante un icono la ubicación actual de la calle.

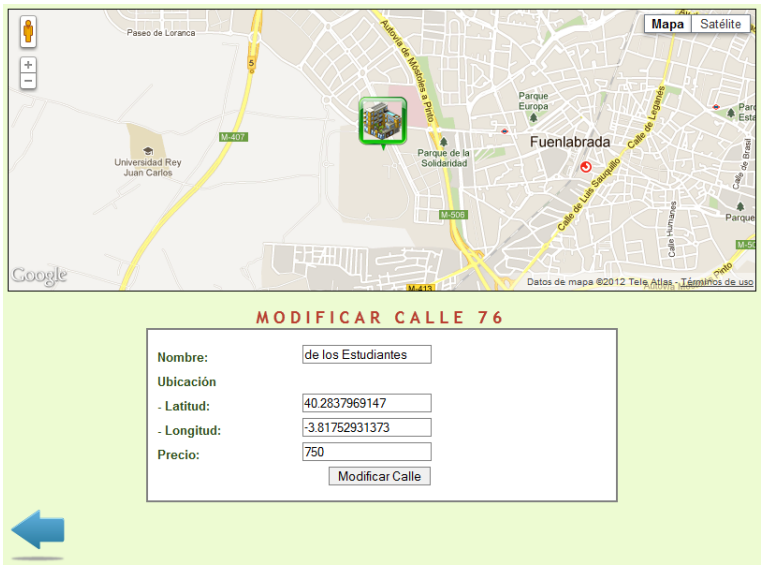


Figura 5.8: Pantalla con el formulario y mapa para la modificación de la calle

Por último, como en los demás recursos del sitio web, se podrá eliminar desde la página "Lista de calles", que contiene el listado con todas las calles perteneciente al juego. Al seleccionar en la columna "Opciones" la acción correspondiente a la fila que deseamos borrar, se le solicitará al servidor que elimine todos los datos de la calle elegida y sus recursos asociados.

Administrar edificios

Para que la partida sea completamente válida, es necesario asociar a cada calle creada un edificio, en caso contrario la partida no se podrá jugar mediante el cliente Android, informando del problema. Los edificios proporcionarán una bonificación extra a los participantes en el transcurso de la partida. Este recurso se podrá administrar accediendo a través de la página "Mostrar calle", pulsando en la opción "Listar edificios" situado en la parte inferior.

Una vez accedida a la primera página “Lista de edificios” se podrá administrar la tabla con el listado del edificio asociado a la calle. Inicialmente no habrá ningún elemento creado, para ello se pone a disposición del Organizador el botón “Crear edificio” localizado en la parte inferior de la tabla, con el cual realizar la acción. La opción mencionada solo será visible si no hay ningún edificio creado, en caso contrario, no podrá acceder al formulario de creación, ya que solo se puede asociar a una calle un único edificio. Si el usuario decida acceder directamente mediante la URL, para crear un segundo edificio, será informado del error mediante la página que se observa en la figura 5.9, como cualquier otro error detectado. Otras acciones disponibles para el usuario en la página del listado serán la opción de eliminarlo y un enlace a la información del elemento creado (ambas opciones como en cualquier otro recurso).

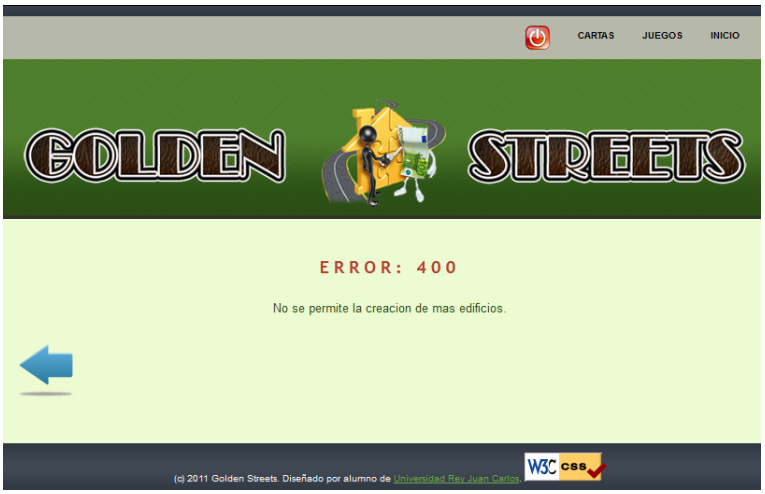


Figura 5.9: Pantalla con el error ocasionado al intentar crear un segundo edificio

Cuando se decida crear un nuevo edificio, se pondrá a su disposición una página con el formulario para su creación, este estará compuesto por el nombre, bonificación y precio. Como todos los demás formulario se validará la correcta cumplimentación de los campos en el lado del cliente mediante el JavaSript LiveValidation y se procederá a su envío al servidor para verificar de nuevo los valores recibidos y realizar la acción que se ha solicitado.

Creado satisfactoriamente o seleccionado en el listado de edificios, se visualizará la información de la propiedad situado en esa calle. Los datos mostrados estarán compuestos por el nombre, bonificación, precio y ubicación. También está disponible la posibilidad de “Modificar edificio” si el Organizador lo considera oportuno. Si decide realizar esta última acción se procederá a la apertura de una nueva página, en ella, se

visualizará el formulario de modificación, similar al de la creación pero esta vez con los campos rellenos con los datos actuales del edificio, como en todos los casos.

Finalmente, se podrá eliminar el edificio asociado a la calle si se considera oportuno desde la página "Lista de edificios". Como en todos los listados explicados, se procede a su borrado mediante la acción "Borrar", con la cual se eliminará completamente la edificación. Realizar esta acción implica la necesidad de volver a crearlo, como dijimos al principio de este subapartado, todas las calles pertenecientes a un juego deberán tener vinculado un edificio, ya que en caso contrario no podrán participar a través del cliente Android.

5.1.2. Monitorización del juego

Finalmente creado completamente un juego de *GoldenStreets* y asociando a este todos los recursos necesarios (equipos, calles, edificios, etc.) para su correcta celebración. El usuario podrá verificar que el juego transcurre sin ningún problema mediante el sitio web. En este se pone a su disposición una página para realizar un exhaustivo seguimiento de las acciones que se llevan a cabo entre los diferentes equipos participantes. Accediendo mediante el menú "Seguimiento", opción que será habilitada en la barra del menú cuando se esté visualizando la información detallada o algún elemento asociado a un juego. Esta página tendrá una tabla representando la clasificación de los equipos, en la cual se muestra la información más relevante de cada uno de ellos. También se visualiza un Google Maps con los diferentes iconos, representando la ubicación geográfica de los recursos. Por último se localiza en la parte inferior de la página un listado con todos los mensajes intercambiados entre los jugadores y el propio Organizador. A continuación se detalla la implementación de cada uno de estos apartados.

Cabe mencionar, que para evitar que el usuario tenga que actualizar constantemente la página para la correcta visualización de la clasificación o la posición de los equipos en el mapa, se ha utilizado un JavaScript que permite refrescar la página solicitando la nueva información al servidor (clasificación, mapa y mensajes). De esta manera, se consigue que el Organizador pueda prestar toda su atención al desarrollo de la partida.

En la parte superior de la página se mostrará la clasificación, denominada "Patrimonios", de todos los equipos participantes con su nombre, valor de patrimonio (campo por el que esta ordenada), dinero actual y número de propiedades (ver figura 5.10). Los datos son solicitados al servidor y actualizados en la página mediante el Script mencio-

nado anteriormente. Con esta tabla se pretende mantener informando al Organizador en todo momento de los datos más relevantes asociados a cada equipo en el transcurso del juego y del ganador final.

Figura 5.10: Pantalla de la monitorización con la clasificación de la partida

A continuación de la tabla con la clasificación, se pone a disposición un Google Maps (ver figura 5.11). Para incrustar este mapa en el sitio web, solo ha sido necesario crear un nuevo Script con el que darle formato y dimensiones al mapa que se visualizará en la página. El motivo de la utilización de este mapa, es para proporcionar al usuario información detallada de los diferentes recursos en el transcurso de un juego, estos elementos estarán posicionados en su ubicación actual, identificados mediante diferentes iconos. Si el Organizador desea obtener información detallada de algún elemento, solo es necesario pulsar sobre este. Pulsando sobre una propiedad se visualizará el nombre, equipo propietario, nombre de la edificación, si está construido, si esta saboteado, precio de la construcción y bonificación que aporta a su propietario. En el caso de seleccionar un equipo se visualizará su nombre, el valor de su patrimonio y el número de propiedades que posee. Finalmente cabe destacar que en la interfaz del mapa se pone a disposición del usuario los controles para modificar el nivel de zoom y desplazamiento

sobrer este.

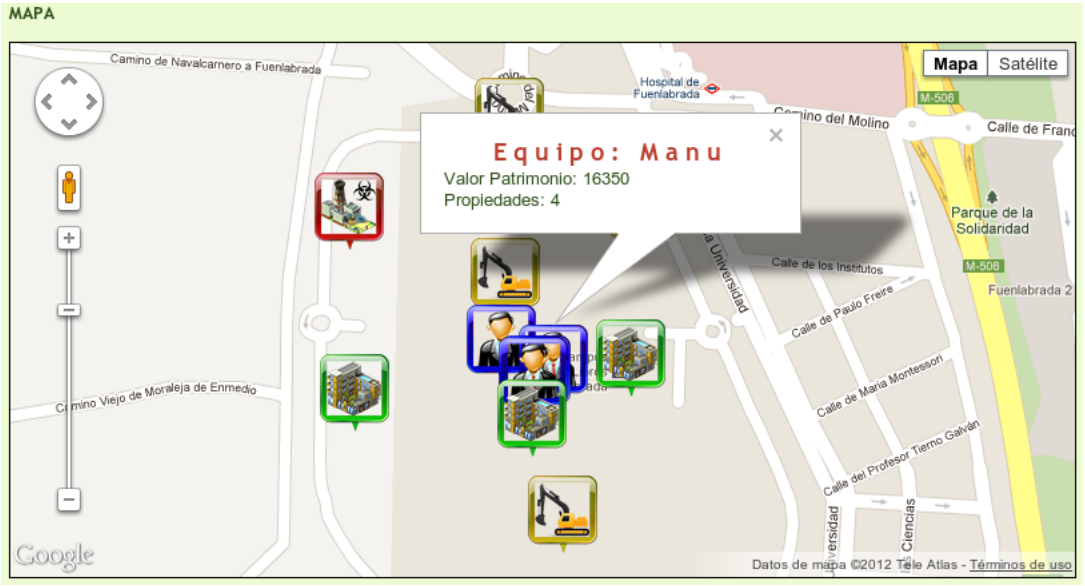


Figura 5.11: Pantalla de la monitorización con el mapa de la partida

Por último lugar destacar, el listado con todos los mensajes intercambiados en el transcurso de la partida (ver figura 5.12). La creación de los mensajes por parte del Organizador serán explicados en el siguiente subapartado debido a su complejidad.

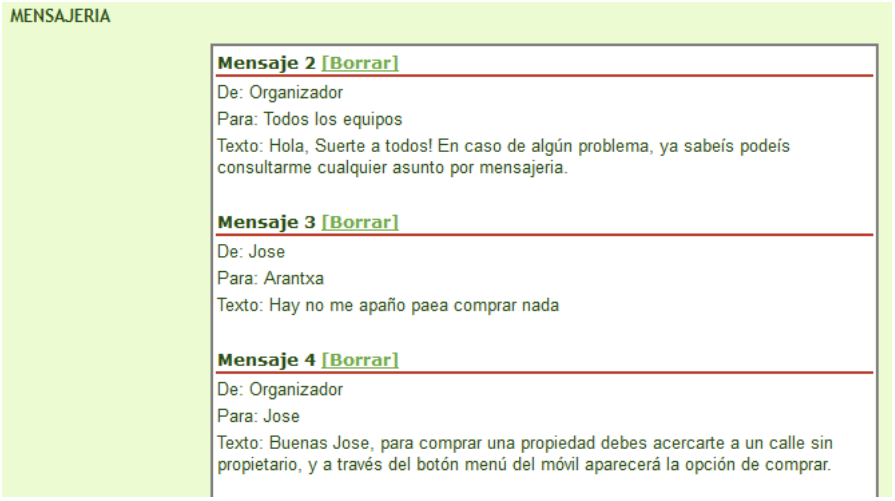


Figura 5.12: Pantalla de la monitorización con el listado de mensajes de la partida

5.1.3. Mensajería

De manera complementaria, se ha implementado en el sitio web un sistema de mensajería con el que poder controlar los inconvenientes surgidos en el transcurso de la partida. De forma similar, también se ha implementado este sistema en el cliente Android detallado su implementación en el apartado 5.2 de la aplicación Android.

El Organizador puede crear en cualquier momento un mensaje a través del sitio web mediante la opción "Mensaje" del menú. En la figura 5.13 se visualiza el formulario de creación para un mensaje, en el cual se podrá especificar si se desea enviar a todos los participantes del juego o únicamente a un equipo. En este último caso mediante un menú desplegable se podrá especificar el destinatario. A continuación, se dispone del campo en el que se debe de escribir el texto del mensaje. Cuando el usuario decida enviar el mensaje, como en los anteriores formularios, se validará mediante el Script si se ha cumplimentado correctamente, informando al usuario en caso contrario. Validado el formulario, se procederá su envío al servidor para la creación y almacenamiento en la base de datos.

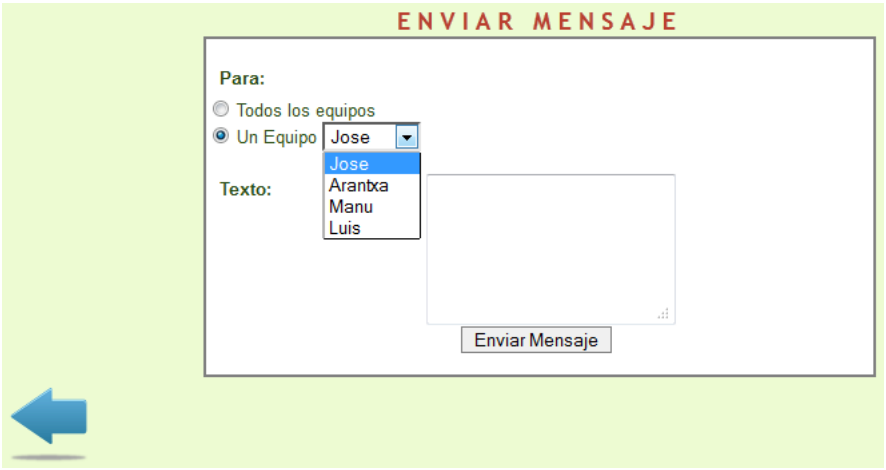


Figura 5.13: Pantalla con el formulario de creación de un nuevo mensaje

Enviado el mensaje correctamente, el Organizador será redirigido a la página del seguimiento. Al final de esta se podrá visualizar una tabla con el listado completo de todos los mensajes intercambiados entre jugador-jugador o jugador-organizador mostrando el emisor, destinatario y cuerpo del mensaje. Cuando se considere oportuno eliminar un mensaje, ya sea por que algún jugador se equivocó al seleccionar el destinatario o por cualquier otro motivo, estará disponible al lado del número de mensaje la opción "Borrar", enviando la solicitud al servidor para que realice dicha acción.

5.2. Cliente Android

En este apartado se examinará la implementación realizada de la aplicación desarrollada en Android para la participación en el juego *GoldenStreets*. Los participantes de este juego geolocalizado, a través de la aplicación serán capaces de ver la localización de las calles, realizar diferentes acciones sobre ellas, enviar ofertas a otros equipos si está interesado en algunas de sus propiedades, contactar con otros participantes de la partida, etc.

Una vez accedida a la aplicación se mostrará al usuario la pantalla de presentación, en el caso de que no se hayan activado los servicios de GPS ni el de la conexión de datos en el terminal móvil antes de acceder a la aplicación (ambos necesarios para un correcto funcionamiento), esta comprobará su estado y mostrará un diálogo alertando al usuario (se utilizará la clase *AlertDialog* de la API de Android [2] para su implementación, apareciendo delante de la *Activity* en curso y requiriendo la acción del usuario para poder continuar con la ejecución), para la activación de los servicios necesarios (ver figura 5.14).

Figura 5.14: Pantallas de presentación de la aplicación y notificaciones de error

Cuando el usuario pulse sobre el botón "Jugar", se pasará a una nueva *Activity* por lo tanto a una nueva pantalla, en la que se deberá seleccionar la parida en la que se desea participar. La lista de juegos que será mostrada se solicitará al servicio web mediante una petición, recibiendo la información que será visualizada en la pantalla, como se puede observar en la figura 5.15. Esta será implementada mediante la clase *ListActivity*

de Android, permitiendo al desarrollador manejar diferentes eventos relacionados con la lista creada, como en este caso la selección de un juego, que a través de sobrescribir (termino que en Java se anota como *@Override*) el método *onListItemClick*, se consigue saber con que elemento se interacciona. En la GUI se mostrará al usuario un listado con todos los juegos disponibles con la información más relevante (nombre, ubicación, dinero inicial, etc.) de cada uno de ellos.

Una vez que el usuario decida la partida de la que desea formar parte, la siguiente pantalla le mostrará una lista de los equipos con los que puede participar en el juego seleccionado anteriormente. Como en la pantalla anterior, la lista se mostrará en la GUI (ver figura 5.15) a través de la implementación de la clase *ListActivity*. Puede darse el caso que en la lista de juegos o equipos no haya ningún elemento para mostrar por la pantalla, o que el Organizador no haya incluido todos los elementos indispensables (equipos y calles con sus edificios) para una partida, por lo que se informará al usuario mediante una *AlertDialog*.

Figura 5.15: Captura con el listado de juegos y equipos disponibles

La obtención de esta lista se realiza de la misma forma que con el listado de los juegos, mediante una petición al servidor web, como en este caso y en todos los casos posteriores la información recibida del servidor llega a la aplicación en formato JSON para su manejo en Android (en algunos casos la información recibida de un JSON ya sea de la lista de juegos, equipos, etc., no se cambiará durante su transcurso, como el nombre del equipo, por lo que será utilizada para la creación de diferentes objetos Java, como pueden ser, *List<Object>*, *String*, etc. De este modo se evita realizar nue-

vas peticiones al servidor, reduciendo la comunicación de datos entre los componentes del sistema). Además de solicitar los equipos también se obtendrá información sobre los elementos indispensables para el juego (citados anteriormente), esta información será comprobada en la aplicación, ya que la ausencia de estos datos en el juego elegido lo haría inviable, y se le informaría al usuario mediante un *AlertDialog*, volviendo al inicio de la aplicación una vez aceptado.

Cabe mencionar que durante toda la implementación de la aplicación se han incluido barras de progreso en determinadas acciones que se realizan y ocasionan el bloqueo de la *Activity*, por ejemplo, petición de datos al servidor, etc. De esta forma se mantiene informado al usuario de las acciones que se están realizando, y se le ayuda a distinguir en el caso que se produjese el mal funcionamiento de la aplicación. Las barras de progreso se han implementado mediante la clase *ProgressDialog* de la API de Android. Además se han incluido notificaciones temporales que aparecerán sobre la *Activity*, sin necesitar la interacción con el usuario y permitiendo al usuario realizar otras acciones con la GUI, consiguiendo informar al usuario de los errores producidos como pueden ser, debido a la perdida de la conexión de datos del móvil (ya sea mediante wi-fi o red móvil) o a la incorrecta recepción de la información enviada por el servidor en formato JSON. Estas notificaciones se han efectuado con la utilización de la clase *Toast* de la API de Android y no con *AlertDialog* para ofrecer al usuario una experiencia más rápida y no provocar tantas interrupciones que dificulten su manejo (ver figura 5.16).

Figura 5.16: Barra de progreso y notificaciones de error durante el juego

Cuando el usuario ya ha elegido el juego y seleccionado el equipo con el que desea

participar, se pasará a una nueva pantalla (acción que en todos los casos será realizado por la clase *Intent* de la API de Android) que servirá al usuario principalmente de información, como se observa en la figura 5.17 se mostrarán recursos seleccionados con anterioridad para evitar equivocaciones durante la partida, también se visualizará un breve texto de bienvenida (común en todos los juegos) y el botón "Jugar" que nos llevará al comienzo de la partida.

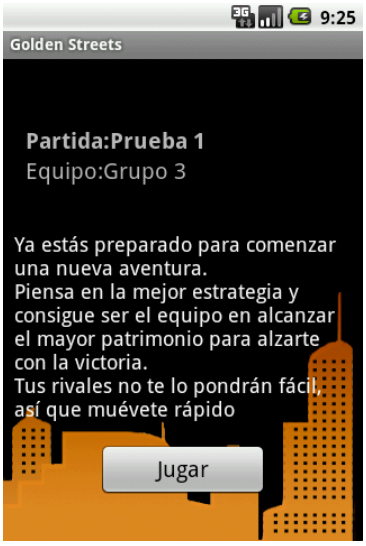


Figura 5.17: Pantalla de bienvenida a la partida

Si el usuario verifica que todo está correcto y no desea cambiar ninguna selección anterior, estará preparado para comenzar. Cuando el usuario pulse el botón destinado a ello se pasará a la nueva pantalla, pero antes se iniciarán en segundo plano de la aplicación un *Thread* de Java que controlará el tiempo que el usuario lleva jugando y cinco nuevos servicios (desarrollada mediante la herencia de la clase *Service* de Android): de posicionamiento, de patrimonio, de cartas, de ofertas y de mensajería. Pero la utilidad e implementación de estos servicios se detallan íntegramente en el subapartado .

La nueva pantalla que se ha abierto se puede considerar la más importante, ya que es donde se va a concentrar todo el desarrollo del juego, gracias a que incluye en la parte superior unas pestañas permitiendo cambiar rápidamente de una *Activity* a otra con tan solo pulsar sobre una de ellas, consiguiendo así una GUI de fácil aprendizaje y manejo para el usuario. Estas pestañas han sido añadidas mediante la instanciación del objeto *TabHost* de Android, que permite crear una lista de pestañas, en la que a cada elemento se le asocia una pantalla, es decir, se crea un *Intent* hacia la nueva *Activity*.

De ahora en adelante en las principales pantallas se incluirá información relevante

sobre la utilización y la finalidad de estas, ayudando al usuario a sacar el mayor partido a la aplicación o resolver alguna duda que le surja en el transcurso de la misma. La opción de ayuda será accesible a través del botón físico "Menú" del termina móvil, con el cual emergerá una ventana (implementada mediante un *AlertDialog*) con la información solicitada que requerirá la interacción del usuario (véase la figura 5.18). Dependiendo de la pantalla en la que se encuentre el usuario también serán mostradas otras opciones en el menú, determinantes para el desarrollo de la partida que serán explicadas en subapartados posteriores, como puede ser la opción "Comprar" de la pantalla del mapa.

En la GUI se podrá observar un total de cinco pestañas diferentes que incluyen iconos y texto para facilitar su identificación, se detallan a continuación:

Estadísticas

Primera pestaña, mostrada por defecto al usuario cuando comienza un juego, y en la que se visualizará la nueva clase *Estadisticas* de la aplicación (implementada mediante la herencia de la clase *Activity*).

Esta pantalla jugará un papel muy importante en el desarrollo de la partida, debido a que en ella se representará el tiempo que un equipo lleva jugando (el tiempo se actualizará en intervalos de un minuto por medio de un *Thread* creado al comienzo de la partida) y el tiempo límite del juego que estableció el Organizador cuando la creó. Como se observa en la figura 5.18 se mostrará al equipo el patrimonio que posee (compuesto por su nombre, dinero del que dispone para realizar diferentes acciones sobre sus posesiones o las de sus rivales, su valor y el número de propiedades que ostenta actualmente), la posición que ocupa en ese momento y la clasificación de la partida (ordenada según el valor total del patrimonio), la cual dará información de los demás participantes. Esta ayudará al jugador a elegir la mejor estrategia en determinados momentos de la partida.

Siempre que el usuario pulse sobre esta pestaña, la clase *Estadisticas* realizará una petición al servidor para actualizar los datos que serán mostrados. En la conexión HTTP realizada se usará el método GET, ya que solo se requiere la información asociada al recurso solicitado. Como ya se comento anteriormente la respuesta será recibida en formato JSON.

Figura 5.18: Capturas de la pestaña Estadísticas del equipo y menú de información

Cartas

Siguiente pestaña de la lista que tendrá asociada una nueva pantalla, es la clase *CartaListar* implementada mediante la herencia de *ListActivity* de Android. En la GUI se presentará al jugador todas las cartas que se irán recibiendo periódicamente en el transcurso de la partida (ver figura 5.19), en la que cada una será identificada con un pequeño icono (clase *ImageView* de Android) situado en la parte izquierda que corresponderá con la acción que se puede realizar sobre las propiedades que posee el usuario o el rival. También se mostrará una breve descripción de la misma y una casilla (en la que se ha empleado un *CheckBox* de la API) con la que se pretende informar, de forma rápida, al usuario de las cartas que ya han sido utilizadas por él o por la propia aplicación.

Cuando el equipo seleccione una de las cartas listadas pulsando sobre ella en la pantalla táctil, se presentará una nueva *Activity* que corresponderá con la visualización del elemento, como se observa en la figura 5.19. En ella siempre se podrá ver el nombre, la acción y su propia descripción, en algunos casos dependiendo del tipo se incluirá nueva información, como puede ser, la cantidad de dinero (en los casos que se trate de acciones que afecten al dinero del equipo) o una nota al pie de la pantalla explicando las instrucciones para su utilización.

La *Activity* que permite la visualización de una carta no dependerá de la pantalla que posee las pestañas, es decir, la GUI únicamente visualizará la información detallada

Figura 5.19: Capturas del listado y vista detallada de una carta

de la carta que se eligió. En el caso que el equipo decida ver otra carta o volver a la pantalla principal deberán usar el botón físico del móvil denominado "Atrás" (botón identificado con el dibujo de una flecha). Así se consigue volver a la pestaña cartas y continuar con el juego.

Mapa

Pestaña más importante en la que se centrará gran parte del desarrollo de esta aplicación. En ella se incluirá un Google Maps en la que se representarán los elementos que conforman una partida para la interacción por parte del usuario con estos. La implementación del mapa y las respectivas acciones que pueden llevarse a cabo serán detalladas en el subapartado 5.2.1.

Ofertas

Al pulsar sobre la pestaña Ofertas, como se puede ver en la figura 5.20 se mostrará al equipo las opciones con las que podrá interaccionar:

- Ofertas Recibidas
- Ofertas Enviadas

Cuando el usuario quiere consultar las ofertas que ha recibido o ha enviado, pulsará el correspondiente botón sobre la pantalla táctil. Se pasará a una nueva pantalla,

que se ha desarrollado con la clase *ListActivity*. En ella, se mostrará un listado de las ofertas enviadas o recibidas según la opción elegida por el usuario, en la que cada elemento de la lista tendrá un icono que indicará el estado actual de la oferta, y una breve descripción de la misma (ver figura 5.20). La obtención de esta lista se realiza mediante una petición *GET* al servidor web.

En caso que el usuario pulse sobre una oferta recibida, se presentará una nueva *Activity* con información detallada de la misma, en ella se mostrará el ofertante que la realizó, el equipo a que va dirigida, la propiedad afectada, la cantidad ofrecida y el estado de la oferta. También en la GUI se mostrará las acciones que el usuario puede realizar si todavía está pendiente de una respuesta, es decir, podrá aceptarla o rechazarla según el interés que le despierte. Si decide rechazar la oferta por que no es de su agrado, se realizará una petición con el método *POST* al servidor para modificar el estado de la oferta en la base de datos. En el caso que decida aceptar la oferta se realizará de forma similar, enviando una petición con el mismo método, en la que el servidor antes de modificar el estado de la oferta comprobará si el equipo que envió la oferta puede hacerse cargo del coste. Se realiza esta comprobación porque desde que se envió la oferta hasta que es aceptada, el dinero del ofertante pudo verse afectado por lances del juego, por ejemplo, la acción de una carta (véase la figura 5.20).

Figura 5.20: Pantallas principal de ofertas, listado y vista detallada de una oferta recibida

Si el equipo decide visualizar una oferta enviada pulsando sobre ella, se pasará a la misma *Activity* que el caso anterior, mostrando en la GUI información detallada de la

misma, pero sobre la que se podrá realizar una acción diferente como se observa en la figura 5.21. En el caso que la oferta este todavía pendiente de una respuesta por parte del propietario, el ofertante podrá retirarla si considera que es excesiva o se equivoco al introducir la cantidad pulsando el botón. Para esto se realiza una petición HTTP con el método *POST* al servidor web para eliminar completamente la oferta de la base de datos.

Al realizar alguna acción sobre cualquier oferta enviada/recibida sin respuesta, se pueden producir problemas de concurrencia, ya que puede ocurrir que varios clientes Android accedan al mismo recurso simultáneamente (la implementación del control de concurrencia se detalla en el apartado 5.1), a continuación se expone un ejemplo para explicar este problema. Un equipo A realiza una oferta sobre la propiedad de un equipo B, pero el equipo A introduce una cantidad mayor de la que quería ofrecer. El otro recibe la oferta y decide aceptarla pulsando sobre la acción correspondiente, en ese mismo momento el equipo A retira la oferta realizada. El resultado que se obtiene es impredecible, pudiendo ser aceptada y borrada a la misma vez, este es el motivo por el que es necesario detectar e invalidar la acción que trate posteriormente el servidor.

Recordar que todas las respuestas enviadas del servidor al cliente por las acciones que se han realizado estarán en formato JSON. Esta será procesada en el lado del cliente para mostrarle una notificación con la información (implementada mediante la clase *Toast* de la API de Android), ya sea por problemas de concurrencia, con el servidor o su correcta realización.

Figura 5.21: Pantallas con el listado y vista detallada de una oferta enviada

Mensajes

La última pestaña, Mensajes, a la que el usuario podrá acceder estará destinada a la comunicación entre los equipos y el Organizador de la partida para que estén en permanente contacto mediante el intercambio de mensajes. Se ha decidido incluir un sistema propio de mensajería interno en la aplicación, para evitar el uso de los servicios incluidos en un teléfono móvil como pueden ser las llamadas, el envió de mensajes, etc. Se ha considerado clave debido a las diferentes funcionalidades que puede cumplir dentro de la aplicación, como puede ser para la negociación entre varios equipos por la compra de una propiedad o el envío de mensajes al Organizador para informar de algún problema en el transcurso de la partida.

Se pueden diferenciar principalmente dos partes en el sistema de mensajes de la aplicación Android, un servicio encargado de informar al equipo de la recepción de estos en el transcurso de una partida, y la GUI con la que el usuario interaccionará con el sistema expuesto. Los servicios desarrollados en la aplicación, incluido el de la mensajería están explicados detalladamente en el subapartado 5.2.

La *Activity* que se visualizará estará formada por dos botones como se puede observar en la figura 5.22, permitiendo al equipo listar los mensajes (enviados y recibidos) y enviar un mensaje nuevo. Cuando el usuario decida ver la lista de mensajes se pasará a una nueva pantalla (implementada por la clase *ListActivity*), en la que se mostrarán todos los mensajes que ha recibido y enviado en el transcurso de la partida. La decisión de unificar los mensajes en una misma *Activity* se llevó a cabo para que al usuario le sea más fácil entender el contexto de un nuevo mensaje con la ayuda de la visualización de todos los mensajes ya sean recibidos o enviados.

De cada elemento se mostrará una breve descripción, y su fondo cambiará de color según el tipo de mensaje, ayudando al equipo a identificarle rápidamente según el emisor.

- El color azul del fondo determina que el mensaje se ha enviado por el Organizador de la partida.
- Los mensajes que el propio equipo envía a los demás serán de tonalidad roja. Se consideran mensajes salientes.
- El color verde identifica a los mensajes que se han recibido de otros equipos. Se consideran mensajes entrantes.

La lista de mensajes se obtendrá del servidor como ya se ha explicado en subapartados anteriores, cada vez que se decida listarlos. Si decide visualizar un mensaje pulsando sobre el elemento en la GUI, se abrirá una nueva *Activity* que corresponderá con la información detallada. En esta, se mostrará el emisor, receptor y texto del mensaje, dando la posibilidad de responder si el usuario lo considera necesario pulsando el botón destinado a ello. En el caso que decida responder al mensaje se abrirá una nueva pantalla en la que redactar un nuevo mensaje (ver figura 5.22).

Figura 5.22: Pantallas del menú principal, el listado y detalles de un mensaje

Cuando el usuario pulse el botón para el envío de un nuevo mensaje o decida responder a un mensaje recibido, la pantalla mostrada será la misma. En la GUI se incluirá el emisor (en todos los casos será el nombre del equipo), y se tendrá que seleccionar al destinatario mediante la utilización de un *Spinner* de la API de Android (permite al usuario escoger un elemento de un grupo, es similar a una lista desplegable), redactar el cuerpo del mensaje y finalmente su envío. Si se decide mandar el mensaje y el cuerpo esta vacío, no se permitirá su envío mostrando una notificación (serán implementadas mediante un *Toast*), también se hará uso de estas notificaciones si el envío se realizo correctamente o en caso contrario se produjo algún error para informar al usuario (ver figura 5.23).

Si el usuario está redactando o visualizando un mensaje y quiere volver a la pantalla principal tendrá que hacer uso del botón físico "Atrás" para volver y continuar con la partida.

Figura 5.23: Pantallas para el envío de un mensaje nuevo

Finalización de la partida

Una vez terminada la explicación de las pestañas de la GUI, se pasará finalmente a explicar cómo se produce la finalización de la partida. Cuando se alcance el tiempo límite establecido por el Organizador se presentará la pantalla "Estadísticas", en la que se mostrará inicialmente una ventana alertando al usuario que vuelva al lugar de origen que se acordó antes de iniciar la partida. Una vez que el usuario cierre el *AlertDialog* para que pueda salir de la partida será necesario que pulse el botón físico "Atrás" del terminal. Cuando se inicie esta *Activity* se finalizarán todos los servicios creados, preparando la aplicación para el comienzo de un nuevo juego (ver figura 5.24).

Si el usuario quiere abandonar la partida antes de que finalice el tiempo, tan solo tiene que pulsar el botón "Atrás" en cualquier pantalla que pertenezca a alguna pestaña, por lo que se mostrará un *AlertDialog* para verificar que desea abandonar, en caso de que se haya equivocado o cambiado de opinión podrá cerrar la ventana y continuar jugando.

5.2.1. Creación y administración de Google Maps

La utilización del servicio de Google Maps [9] ofrece al desarrollador la posibilidad de añadir mapas con múltiples funcionalidades, gracias a una librería externa que se encuentra en el paquete *com.google.android.maps*. Debido a las necesidades de la aplicación, el manejo de un mapa es fundamental porque las principales funciones del

Figura 5.24: Pantallas de finalización de la partida

juego se realizan sobre este, como puede ser el posicionamiento de los equipos y las diferentes acciones que un usuario puede realizar con las calles. La principal clase de esta librería es *MapView*, una vista que permite representar el mapa con las diferentes capas asociadas (*Overlays*) y que solo puede ser construida sobre un *MapActivity*. Esta vista será la encargada de manejar los desplazamientos y zoom que se realicen sobre el mapa.

Para poder visualizar un mapa en Android es necesario registrarse en el servicio de Google Maps para obtener el *Maps API Key* [10] asociado al certificado con el que se ha firmado la aplicación. Android proporciona para todos los desarrolladores un certificado de prueba (fichero *debug.keystore*). Por último se debe referenciar en el fichero *manifest.xml* las librerías utilizadas.

Para representar las diferentes calles y equipos se ha creado la clase *DatosMapaItemizedOverlay* que extiende de *ItemizedOverlay<OverylayItem>*, consiguiendo añadir en el mapa una nueva capa con la información detallada de los elementos del juego. El icono que identifica a los equipos rivales es el mismo para todos, en el caso de las calles, su representación visual dependerá de las acciones que se llevarán a cabo en el transcurso de una partida por parte de los usuarios (ver figura 5.25), dichas acciones se explicarán con detalle en el subapartado 5.2.2.

La visualización del equipo del usuario, se ha realizado con la creación de la clase *EquipoItemizedOverlay*, debido a que su representación en le mapa es diferente al del resto de los equipos, ha sido necesario redefinir el método *draw()*. A diferencia de estos,

como se observa en la figura 5.25 al usuario se le ha incluido una circunferencia de color verde representado el área de acción con las diferentes calles, es decir, si dentro de su área se localiza una calle, el usuario podrá realizar las acciones pertinentes dependiendo del estado de esta. Tanto los iconos de los equipos y propiedades rivales, estarán en blanco y negro.

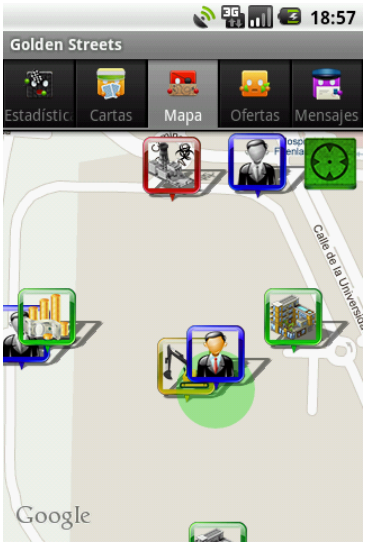


Figura 5.25: Captura de los iconos de los equipo y calles en la pestaña del mapa

Cuando el usuario se encuentre en el mapa y necesite conocer algún detalle ya sea de una calle o de otro equipo, bastará con pulsar sobre el elemento elegido para visualizar un *Toast* con su información. Si se pulsa sobre un equipo se mostrará únicamente su nombre, en este caso no se ha incluido más información para no ser redundantes con la misma, ya que el patrimonio que este posee se puede visualizar con mayor detalle en la clasificación de la pestaña "Estadísticas". En el caso de que el jugador pulse sobre una calle para conocer más datos sobre ella, se mostrarán dos bloques de información, uno sobre la calle y otro sobre su propiedad. En el primero de ellos se incluirá el nombre, el precio y propietario de dicho elemento. El último bloque contendrá el nombre de la edificación, su estado de construcción y si está saboteada.

Para controlar la pulsación sobre cualquier elemento del mapa ha sido necesario sobrescribir el método *onTap(int index)* de las clases que se han creado, ya que este es el encargado de manejar dicha acción. En estas clases se puede tener una lista con las diferentes capas que hay que pintar en el mapa, para saber cuál de estas se ha pulsado se utiliza el parámetro *index*, mostrando al usuario un *Toast* con la información obtenida del elemento (ver figura 5.26).

Figura 5.26: Capturas de las notificaciones con la información de un equipo y una calle

Otro aspecto a destacar en la pantalla del mapa es el menú proporcionado al usuario con la pulsación de su correspondiente botón físico en el terminal ya que a aparte de la información de ayuda incluida en cada *Activity*, se ha incluido las acciones permitidas al usuario sobre una calle. En este menú no se mostrará todas las opciones que se han programado en la aplicación, sino que dependerá de la distancia que se encuentre de las calles y de su estado. En el caso de que varias de las calles se encuentren dentro del área del equipo solo aparecerán las acciones de la calle más cercana a la ubicación del equipo.

5.2.2. Acciones del juego

A continuación se citan las acciones que se pueden realizar en el transcurso de una partida analizándolas y explicando su implementación en los siguientes subapartados:

- Comprar una calle
- Vender una calle
- Edificar una calle
- Sabotear una calle
- Demoler una edificación
- Realizar una oferta

Comprar una calle

Esta acción solo estará disponible al usuario en el menú cuando se aproxime lo suficiente a una calle que no tenga ningún propietario, como se observa en la figura 5.27. Con ella conseguirá incrementar su patrimonio con una nueva propiedad si la acción se realiza con éxito. El equipo puede conocer de forma detallada la información asociada a la calle entre la que se encuentra su precio, con tan solo pulsar sobre su icono. Si el usuario finalmente decide realizar la acción se le mostrará una notificación implementada mediante la clase *AlertDialog* de Android(ver figura 5.27). En ella, el usuario podrá ver el nombre de la calle que va a comprar junto con su valor económico, dando la opción de continuar con la compra o de cancelarla si al final cambia de opinión.

Figura 5.27: Capturas de la opción Comprar en el menú del mapa y su notificación

Cuando el usuario decida realizar la compra de una calle, el cliente Android enviará una petición *POST* al servidor web con información del equipo comprador, en el cuál se realizarán diferentes comprobaciones antes de asignarle la calle al equipo, para un correcto funcionamiento. La primera de estas dependerá del dinero que posea en ese momento el comprador, ya que no se permite que un equipo realice ninguna acción cuando este se encuentra en números negativos. La otra estará relacionada con la posibilidad de que dos o más equipos quieran comprar simultáneamente la misma calle ocasionando problemas de concurrencia, el control de la misma se realizará en el servidor.

Finalmente la calle será adquirida por el equipo que tenga el dinero suficiente para afrontar los gastos y en caso de que varios equipos realicen la acción a la vez sobre

la misma propiedad, esta será asignada a la primera petición que atienda el servidor negando la compra al resto de ellas. El servidor enviará la respuesta en formato JSON a cada cliente Android que realizó la acción, esta será tratada por la aplicación y generará una notificación *Toast* informando al usuario si ha logrado adquirir la calle o por el contrario se ha producido algún problema, ya sea por un error de conexión u otro equipo la adquirió antes. Además el icono que la representa en la GUI del mapa cambiará para indicar que la calle está lista para su edificación (ver figura 5.28).

Figura 5.28: Notificaciones con los posibles resultados de la compra de la calle

La compra de una calle aportará diferentes beneficios al equipo poseedor, ya que se verá incrementado su número de propiedades y lo que es más importante, su valor de patrimonio (factor que determinará el ganador de la partida) en relación al precio de la calle adquirida. Además, por cada calle sin edificar que posea el equipo periódicamente le será ingresado el dinero correspondiente a la mitad de la bonificación que esta posea, esta tarea será realizada en segundo plano por el servicio *PatrimonioServicio* que se detallará su implementación en el subapartado 5.2.

Vender una calle

Una vez realizada la primera prueba, detallada en en el apartado 6.1, los voluntarios que participaron en esta, sugirieron añadir una nueva funcionalidad denominada "Vender". Esto se debe a que si alguien tiene la mala suerte de cara y se endeuda rápidamente, el sistema está programado para que no pueda gastarse más dinero, es decir, no le permite realizar ninguna acción. La única manera de ganar algo de efectivo era

negociando con los diferentes participantes, pero estas se pueden prolongar más de lo deseado, haciendo perder un tiempo muy valioso a ambas partes. Por lo que finalmente se analizó, diseñó e implementó dicha función.

Siempre que un usuario se acerque a una propiedad en su poder, esta opción estará disponible, pudiendo venderla por una cantidad inferior a la adquirida, penalizando siempre su valor. En el caso que se disponga de alguna edificación, su precio de venta se incrementará pero siempre saldrá perdiendo económicamente el usuario que decida utilizar esta acción, a intentar negociar con los demás participantes un precio superior por la propiedad. Al decidir realizar dicha función, se enviará una petición *POST* con la información de la calle implicada. El servidor realizará las modificaciones necesarias y le envirará la respuesta al cliente Android, el cual informará al usuario mediante un *Toast*. En la figura 5.29 se aprecia el proceso que hay que seguir.

Figura 5.29: Capturas del proceso de la venta de una calle

Edificar una calle

En el transcurso de una partida el equipo podrá contar con diferentes calles en su posesión de las que sacará un mayor rendimiento en los ingresos con su edificación. Esta acción aportará al usuario el importe integro de la bonificación que la calle tiene asociado y se incrementará su valor de patrimonio en relación al gasto realizado en la mejora del estado de la calle. Para que el usuario conozca el coste que conlleva la construcción del edificio, podrá pulsar sobre su calle para visualizar en un *Toast* esta

información, evitando el desplazamiento del usuario en el caso que no esté interesado o que no tenga dinero suficiente para afrontar la mejora.

Para poder visualizar sobre una calle de su propiedad la acción "Construir" entre las opciones del menú del mapa, será necesario que el equipo se acerque lo suficiente a su objetivo, es decir, que la calle este dentro del radio de acción del usuario y que esta sea la más cercana a él. Una vez que se decida pulsar sobre esta opción le será mostrado un *AlertDialog* como paso intermedio, previniendo así una equivocación del usuario a la hora de pulsar sobre la opción deseada en la pantalla táctil. Esta ventana contendrá información sobre el nombre y precio de la calle, dándole la opción de seguir con la edificación o de cancelarla porque ya no desea continuar. En determinadas ocasiones se mostrarán dos botones para poder continuar con esta acción, el primero y el que siempre aparece estará encargado de edificar la calle pagando el coste que tiene asociado, el otro solo se mostrará al usuario si posee en su colección como mínimo una carta que le permita la construcción de un edificio y que esta no haya sido usada previamente en la partida.

Cuando el usuario finalmente escoja alguna de las opciones para continuar con la edificación, la aplicación Android establecerá la conexión con el servidor para el envió de información del equipo y calle implicados a través de una petición *POST*. El servidor realizará diferentes comprobaciones antes de llevar a cabo la acción, para evitar posibles problemas como la incoherencia de datos o concurrencia. Lo primero que se verificará es que la calle pertenece al equipo que pretende edificar en esta, otra contrastará que el estado de la propiedad sea el de construcción.

Por último y solo cuando no se realiza la acción mediante una carta, se comprobará el dinero que el equipo tiene en ese momento y pueda costear su gasto. Una vez finalizado el proceso en el servidor, se establecerá conexión con el cliente para tratar la respuesta generada. El usuario será informado mediante la visualización de un *Toast* en la GUI del mapa, si finalmente consiguió edificar la calle o por el contrario no lo logro. Por último si se realizó con éxito toda la acción se modificará el icono de la calle representando el nuevo estado de la misma (ver figura 5.30).

Figura 5.30: Capturas del proceso de la edificación de una calle

Sabotear una calle

Otra de las acciones disponibles en el menú del mapa durante el transcurso del juego será "Sabotear". Con la utilización de esta acción se pretende fomentar el intercambio de calles entre los equipos participantes, teniendo en cuenta que poseer una propiedad afectada en el transcurso de la partida ocasionará gastos reduciendo su dinero y valor del patrimonio en relación al coste. Para poder visualizar esta acción en el menú es necesario acercarse a la calle de otro equipo participante y que esta tenga construido su edificio. Si el usuario decide sabotear una calle se le mostrará un *AlertDialog* con su nombre, siempre que posea una carta para esta acción. Esta ventana es utilizada por seguridad y evitar un posible error del equipo, que permite continuar con la acción o cancelarla. En caso de que el saboteador no disponga de una carta en su colección sin usar para esta acción, el equipo será notificado mediante un *Toast*.

Cuando el usuario ha decidido finalmente continuar con el sabotaje de una propiedad, el cliente Android establecerá la conexión con el servidor mediante una petición con el método *POST*. Como en los casos anteriores también habrá que controlar la concurrencia, una vez realizado todo el proceso se envirará la respuesta a la aplicación Android para que informe mediante un *Toast* si la realización de la acción se desarrollo satisfactoriamente o por el contrario se produjo algún problema. Si se logró correctamente, el icono mostrado en la pantalla del mapa se cambiará para representar el nuevo estado de la calle (ver figura 5.31).

Figura 5.31: Capturas del proceso de sabotaje de una calle

Demoler una edificación

A parte del sabotaje de una calle el equipo también contará con la acción "Demoler" para perjudicar a su rival. Como en todas las acciones del juego será necesario aproximarse a la calle en la que se desea aplicar, además como en el caso de sabotaje, esta acción solo se podrá realizar por parte del equipo si posee en su baraja una carta que le permita realizarla y que no haya utilizado previamente en el transcurso de la partida. Si el equipo selecciona la opción en el menú y no tiene la carta correspondiente será notificado con un *Toast*, en caso contrario, como se puede observar en la figura 5.32 se mostrará una ventana con el nombre de la calle que será objetivo de la acción, interaccionando con ella el usuario podrá seleccionar si desea continuar con la demolición.

En el momento que el equipo decida llevar a cabo la acción seleccionada se enviará una petición *POST* al servidor web, en el cual una vez procesada la solicitud se envía la respuesta al cliente. La información recibida en formato JSON será manejada por la aplicación Android notificando (a través de un *Toast*) al equipo si se realizó correctamente la demolición en la propiedad seleccionada o por el contrario se produjo algún error en su proceso. Cuando se realiza con éxito la acción se verá afectado el patrimonio del equipo en proporción al valor que tenga el edificio destruido.

Destacar que esta acción no solo servirá para importunar a un equipo, sino que también será beneficiosa para el propio usuario, ya que esta acción es la única carta

Figura 5.32: Capturas del proceso de la demolición de una edificación

capaz de eliminar el estado de sabotaje de una calle de su posesión. Aunque el uso de esta afecte a su valor del patrimonio, evita que en el transcurso de una partida el equipo tenga pérdidas de una calle saboteada.

Realizar una oferta

La última opción implementada que aparecerá en el menú de la GUI del mapa será "Oferta". Igual que el resto de las acciones implementadas se requerirá que la calle objetivo este dentro del radio de acción y sea la más próxima al equipo. Las ofertas serán la única manera de que un equipo pueda hacerse con la calle de otro en el transcurso de una partida. También será otra de las formas con la cual un equipo se pueda deshacer de la calle que le está causando pérdidas o necesita un ingreso económico para sanear sus cuentas.

El equipo podrá elegir la propiedad que más se ajusta a sus necesidades comparando la información relacionada a cada una de estas pulsando sobre su icono. Una vez que el equipo haya elegido la calle, cuando pulse sobre la acción se mostrará una nueva *Activity*. Como se observa en la figura 5.33 en esta nueva pantalla se presentarán todos los detalles asociados a una calle, el dinero del equipo, una casilla para introducir el precio que se desea ofertar y el botón con el cual se realiza la oferta. La información que se visualizará está compuesta por el nombre y precio original de la calle, nombre del propietario y por último el nombre y estado de la edificación. En función del dinero disponible que tenga el usuario podrá introducir el precio de la oferta que crea

conveniente en la casilla destinada a ello (limitada solo a valores numéricos). Cuando se seleccione el botón destinado a realizar la acción se comprobará que el equipo no haya dejado la casilla vacía y que el valor ofertado no supere sus fondos, mostrando en ambos casos una notificación mediante un *Toast* para informar al usuario. Cabe mencionar como se muestra en la pantalla que para que una oferta pueda ser aceptada, el equipo debe poseer la cantidad ofrecida en el momento que el propietario de la calle acepte la oferta.

Figura 5.33: Pantallas del menú y realización de una oferta

Una vez que los datos hayan sido verificados en la parte del cliente, estos serán enviados al servidor para efectuar diferentes comprobaciones y si todo se realiza correctamente finalmente se creará la oferta. El resultado del proceso se enviará al cliente Android, el cual mediante un *Toast* notificará la respuesta al usuario. Cuando se realiza con éxito una oferta o se responde la oferta de un equipo, estos serán informados mediante una nueva notificación en la barra de estado del móvil.

Por último mencionar que si un usuario finalmente decide no realizar la oferta sobre la calle elegida, para poder volver a la GUI principal y continuar con el juego, solo tendrá que pulsar el botón físico "Atrás" del terminal móvil.

5.2.3. Servicios

En el transcurso de una partida es necesario realizar diferentes tareas de forma transparente al usuario para el correcto funcionamiento de la aplicación, aunque en

algunos casos sea posteriormente informado. Para la implementación de estas tareas se ha utilizado la clase *Service* de Android, que permite ejecutarlas en segundo plano sin la necesidad de la interacción del usuario. Pero un *Service* corre por defecto en el hilo principal pudiendo bloquear la aplicación si hace un uso intensivo. Por este motivo, como norma general se ha utilizado un *timer* (clase de Android) que nos proporciona un nuevo hilo para cada servicio y nos permite determinar su periodo de ejecución en la partida, preservando el consumo de batería. A continuación se explica detalladamente la funcionalidad e implementación de los servicios que se han creado.

CartaServicio

Este servicio tendrá el cometido de obtener y repartir periódicamente las diferentes cartas al equipo en el transcurso de una partida. Estas serán creadas previamente por el Organizador del juego en el servidor web. La aplicación Android se encargará de enviar una petición al servidor mediante el método *GET* para requerir las cartas de la partida, en el caso que hubiese algún problema al realizar la conexión el servicio lo reintentará las siguientes ejecuciones hasta que reciba la respuesta del servidor, este problema ocasionará la perdida de esa carta. Cuando el servicio reciba la respuesta del servidor en formato JSON, este la procesará y en el caso que haya recibido las cartas se almacenarán internamente. Cuando se reparte una carta, el equipo será alertado de su acción mediante un *AlertDialog* posibilitando visualizarla de forma detallada o continuar jugando.

Un *Service* solo puede realizar notificaciones a través de un *Toast* o a través de la barra de estado de la GUI del terminal, en este caso se ha decidido mostrar un *AlertDialog* necesitando registrar en cada *Activity* que es visible, el uso de un *BroadcastReceiver* de Android encargado de notificar al usuario la recepción de la carta. El registro se realizará cuando el usuario este visualizando esa pantalla (método *onResume()*) y en el caso que cambie a otra (método *onPause()*) se elimina el registro que se realizo y se añade en la que se visualizará a continuación. El servicio enviará un mensaje a todas las pantallas, ya que se desconoce cuál de ellas está en primer plano y será atendido por la que tenga ese momento el registro del *BroadcastReceiver* mostrando la notificación.

MensajeServicio

Otro de los servicios implementados se encargará de comprobar periódicamente si el equipo ha recibido un nuevo mensaje ya sea de otro equipo o del Organizador. El

cliente Android realizará la solicitud al servidor mediante una petición *GET*, el cuál le enviará la respuesta con el identificador del último mensaje destinado al equipo. La aplicación comprobará si ya ha sido visionado ese mensaje y en caso que no lo haya sido, como se observa en la figura 5.34 se le añadirá una nueva notificación en la barra de estado del terminal, interaccionado con esta se pasará al listado de los mensajes, también será alertado mediante un sonido.

OfertaServicio

Este servicio estará ligado a las ofertas del juego informando al equipo de diferentes sucesos. Se notificará al usuario si ha recibido nuevas ofertas sobre alguna de sus propiedades y también si se ha respondido alguna de las ofertas realizadas a la propiedad de otro equipo. Para que el cliente pueda comprobar cualquiera de estos dos eventos establecerá conexión con el servidor mediante una petición *GET*, respondiendo con el número total de ofertas respondidas y el identificador de la última oferta recibida. Una vez procesada la respuesta se añadirán las correspondientes notificaciones en la barra de estado (ver figura 5.34).

PatrimonioServicio

Este servicio se encargará de realizar periódicamente el ingreso o el retiro del dinero que sus propiedades van ocasionando en el transcurso de la partida, lo que incrementará el valor de su patrimonio. Para ello la aplicación Android realizará la solicitud al servidor en el cuál se realizarán las modificaciones necesarias en el patrimonio del equipo afectado.

PosicionamientoServicio

El último servicio implementado está relacionado con la ubicación del equipo, el cuál obtendrá su localización mediante el GPS disponible en el terminal móvil para su utilización en el mapa y que los demás equipos conozcan en todo momento su posición. Este *Service* implementa la *interface LocationListener*, con la cuál se conseguirá saber cuándo se ha actualizado la posición del equipo mediante el método *onLocationChanged()* y así evitar establecer una conexión con el servidor. En este servicio la actualización de las coordenadas del usuario dependen de dos factores, el primero con la distancia recorrida y el segundo con el tiempo transcurrido. Cuando se obtenga la nueva posición se almacenará de forma interna en la aplicación Android para sus uso y

también se enviará mediante una petición *POST* al servidor para su actualización en la base de datos y uso por parte del resto de los equipos.

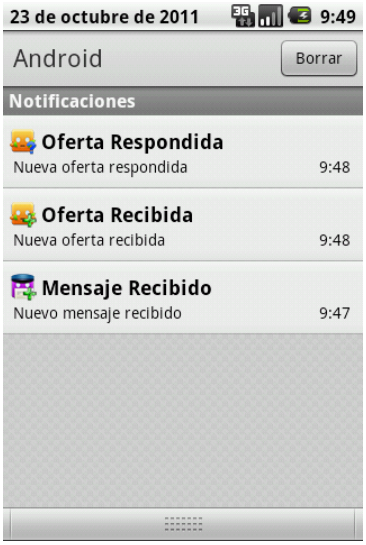


Figura 5.34: Captura de las notificaciones en la barra de estado

Capítulo 6

Pruebas y resultado

Durante el desarrollo del proyecto se realizaron numerosas pruebas a los diferentes elementos que componen el sistema (cliente Android, servicio web), siendo verificadas las funciones de la aplicación de una en una cuando eran implementadas, lo que intentaba buscar una mayor estabilidad, evitar riesgos y detectar rápidamente los problemas, si surgían.

Las pruebas, que podemos considerar de verificación, sobre el servicio web se realizaron inicialmente sobre el servidor de Django, para descubrir errores durante el desarrollo. Una vez solventados se puso en funcionamiento el servidor Apache, el cual daría soporte en las pruebas realizadas con los diferentes voluntarios, generando en este una mayor carga de tráfico. La aplicación Android, como comentábamos antes, se desarrollo probando las diferentes funcionalidades una a una. Se realizaban las pruebas en el emulador de dispositivo que proporciona el Android SDK. En caso de detectar algún problema, en el emulador o terminal, se utilizarán las herramientas de depuración para realizar una traza y así identificar el error y solventarlo.

6.1. Primera prueba y resultados

Esta primera prueba estará enfocada a probar el funcionamiento básico de la aplicación desarrollada en terminales físicos, para ello un número reducido de voluntarios jugarán una partida, preparada a través del servicio web, en un escenario real, siendo este el Campus de Fuenlabrada de la Universidad Rey Juan Carlos (URJC). Con esto se pretende fijar como objetivos comprobar la correcta implementación y fiabilidad del sistema generado. Además de recopilar en una encuesta la opinión de ellos sobre dife-

rentes aspectos, como pueden ser la interfaz de usuario, manejo o funcionalidad, entre otros. Con el fin de mejorar el sistema.

Para cumplir con los objetivos marcados anteriormente, se quedó con el grupo de voluntarios el día 23 de Noviembre de 2011 en la ubicación acordada. El grupo estaba formado por dos estudiantes de ingeniería informática y un recién licenciado en la misma especialidad, no fue posible buscar más voluntarios debido a la proximidad de esta fecha con los exámenes del primer cuatrimestre. A continuación se les expuso mediante una presentación las características de la aplicación, así como sus funcionalidades, para solventar las posibles dudas surgidas. Finalmente se les repartió los terminales (HTC Magic) con la aplicación instalada y empezaron a jugar una partida de 45 minutos. A la finalización de esta y al regreso de los participantes, se les pidió que cumplimentasen una encuesta realizada en el Moodle de la URJC [25].

A continuación se analizan y detallan los resultados obtenidos de la primera encuesta realizada a los voluntarios, siendo esta recogida íntegramente en el anexo A.1. Para una mejor comprensión y estudio de las respuestas, se ha estructurado en diferentes bloques, de esta forma también se facilita la realización de la misma a los usuarios agrupando las preguntas.

El primer bloque está enfocado a la explicación realizada en la presentación y la ayuda mostrada en la aplicación. De los resultados se puede extraer que es recomendable una breve explicación de la misma, para su mayor comprensión. Esto puede ser debido a su carácter geolocalizado, un tipo de juego no muy expandido entre los participantes. Además de tratarse de un juego con tintes sociales siempre existe un componente de rivalidad, lo que lleva a los participantes a aprovechar al máximo el tiempo, viéndose reflejado en el transcurso de la partida, ya que solo uno de los voluntarios dedicó algo de su tiempo en consultar la ayuda. Como conclusión, podemos afirmar que con una breve explicación y la ayuda ofrecida en la aplicación, los jugadores han tenido pocos problemas para comprender la dinámica del juego.

A continuación se realizan las preguntas enfocadas a la Interfaz gráfica del usuario (GUI) para verificar su manejo, calidad y utilidad. En base de las respuestas dadas se puede afirmar que los voluntarios han percibido una interfaz sencilla y buena, como se pretendía. Aunque como todo es mejorable, ya que destacan algunas características para mejorar principalmente la pestaña de Estadísticas y del Mapa, en esta última un voluntario propone: "Añadir algún elemento para mejorar la orientación del usuario".

Otro bloque propuesto, está enfocado a las funcionalidades y estabilidad de la aplicación. Teniendo como objetivo adecuar el juego a las necesidades de los usuarios y

descubrir los errores producidos, se pude generalizar que las funcionalidades implementadas han sido acogidas satisfactoriamente por ellos, y la partida transcurrió sin ningún contratiempo para los participantes, salvo en uno de ellos. A este jugador se le cerró inesperadamente la aplicación mientras caminaba, analizando el informe del error obtenido cuando devolvió el terminal, se pudo detectar que mientras se desplazaba pulso una determinada acción sin darse cuenta sobre una propiedad, pero en ese momento se actualizaron los recursos provocando el conflicto, ya que en su nueva área de acción no había ninguna calle.

El siguiente bloque pretende recopilar la valoración general que han tenido los voluntarios, preguntándoles tanto la estética general (GUI), rendimiento, fiabilidad y su valoración media. Como conclusión a esta serie, se puede extraer el alto grado de aceptación por parte de los participantes para tratarse de una primera prueba.

Al final de la misma, se ofrece la libertad a los usuarios de destacar los puntos fuertes y débiles de la aplicación expuesta. Dándoles la posibilidad de realizar cualquier sugerencia, comentario o aclaración ya sea de la aplicación o de la encuesta. Las acciones implementadas y el carácter geolocalizado del juego han gustado bastante, pero el principal inconveniente que han encontrado es la alta frecuencia de las cartas que se reparten periódicamente en el juego. También han propuesto nuevas funcionalidades para la aplicación que han sido estudiadas y analizadas.

6.2. Segunda prueba y resultados

Realizada la primera prueba y analizados sus resultados, se comprobaron los errores producidos y se valoró la opinión dejada por los usuarios, lo que implicó la modificación de la aplicación. Se corrigió el error surgido a un voluntario en la anterior prueba, ocasionándole el cierre inesperado de la aplicación. Además viendo el consumo de batería se optó por mejorar la actualización del posicionamiento del usuario, así como atendiendo a las peticiones de los voluntarios se implementó una nueva función, la cual permite vender una propiedad al sistema de forma instantánea por una cantidad menor a la adquirida. Por último, se cambio el diseño de algunos iconos del mapa, para identificar rápidamente las propiedades que posee el equipo, además del incremento del área de acción y reducción en el reparto de las cartas.

En esta segunda prueba se pretende verificar nuevamente el correcto funcionamiento de la aplicación modificada. En este caso, estará enfocada a un mayor número de

personas, incluso habrá voluntarios que participen en grupo a través de un mismo terminal, ya que también se pretende descubrir la mejor forma de participar en el juego La prueba se desarrollará en el Campus de Fuenlabrada de la URJC, como la anterior. Fijándose como objetivos comprobar la correcta implementación, número de participantes por grupo y fiabilidad del sistema, así como recopilar la valoración realizada por los voluntarios para la mejora de la aplicación.

Para cumplir con los objetivos marcados anteriormente, se quedó con el grupo de voluntarios el día 19 de Enero de 2012 en la ubicación acordada. Está vez el grupo estará formado por estudiantes de la Escuela Técnica Superior de Ingeniería de Telecomunicación, a los que se les expuso a través de una breve presentación las características más importantes de la aplicación. Una vez finalizada la explicación, se solventaron las dudas surgidas y se repartieron los dispositivos móviles, en este caso se dispuso de dos modelos con versiones diferentes de Android. Al final de la partida, con duración de 45 minutos, se les solicitó que completasen una encuesta para su posterior análisis.

Seguidamente se analizan los resultados obtenidos de la segunda encuesta realizada, recogida en el anexo A.2. Como en la anterior para facilitar el análisis y cumplimentación por parte de los voluntarios, se ha estructurado en diferentes bloques agrupando las preguntas. Esta vez el número de preguntas realizadas es menor y generales, debido a la aceptación de la aplicación en la primera encuesta.

En el primer bloque se realizan las preguntas relacionadas con la presentación de la aplicación y su utilidad. Los resultados extraídos, son similares a la primera, obteniendo como punto común que es recomendable una breve explicación de la misma. Como se puede apreciar en preguntas finales, los voluntarios le dan un carácter competitivo y participar en este tipo de juegos sin conocerlo previamente le quita este importante aliciente.

El siguiente bloque de preguntas se enfoca a las funcionalidades y estabilidad que ofrece la aplicación, con el objetivo de verificar la aceptación por parte de los usuarios de las modificaciones realizadas. Durante el transcurso de la partida ningún participante presentó ninguna anomalía o cierre inesperado, lo que no exime que haya algún error pero se presenta una aplicación más madura. Las diferentes funcionalidades, como el aviso de notificaciones o la corrección que se realizó en el área de acción, han sido bien acogidas recibiendo en general una alta valoración.

Seguidamente se sitúa una serie de preguntas destinadas a obtener la evaluación general de la aplicación, es decir, se intenta recopilar información sobre la interfaz gráfica, consumo de batería, diseño, fiabilidad y valoración global. Analizadas las respuestas, se

puede extraer el alto grado de aceptación, ya que todos los campos han sido valorados satisfactoriamente, mejorando los resultados de la primera prueba.

Un objetivo fijado inicialmente, es conocer los participantes que deberían de formar un grupo, para esto se pregunto a los voluntarios su opinión al respecto. En la prueba realizada solo un equipo estaba formado por dos miembros, pero de los cinco participantes cuatro consideraron que es recomendable la participación en pareja. Además cabe destacar que todos los participantes probablemente repetirían la experiencia, lo que es una agradable noticia y se puede afirmar la buena aceptación realizada por los voluntarios.

Finalmente se pide a los diferentes voluntarios que destaquen las características que más le ha gustado y la que menos. A la mayor parte le agrado la competitividad que se genera al participar en el juego, así como una interfaz gráfica sencilla y un participante destacó: "Buen uso de geolocalización". En contrapartida, la mayor parte expone la dificultad que tiene hacerse con otras calles, ya que sus propietarios no aceptan las ofertas realizadas. Por último, dos de los participantes proponen diferentes mejoras que les gustaría que se implementasen en el juego. Una es adquirir cartas aunque se penalice con un alto coste de dinero, pero al implementarlo se perdería un importante factor, el azar, de esta forma cualquiera podría acceder a las cartas más importantes. Otra idea es reducir el tiempo de visualización de los *Toast* que muestran la información en el mapa.

Como conclusión a esta encuesta y a la anterior, se puede destacar el alto grado de satisfacción con la que se quedaron los participantes al finalizar las pruebas, y este hecho se ve reflejado en las valoraciones recogidas en las encuestas examinadas A.

Capítulo 7

Conclusiones

En este apartado se extraen las principales conclusiones obtenidas en la realización del Proyecto Fin de Carrera, así como el grado de éxito alcanzado en base a los resultados obtenidos y los objetivos planteados en su desarrollo. También se expone el posible enfoque que podría tener sobre posteriores trabajos y estimaciones de las ideas con las que continuar el futuro desarrollo del sistema. Finalmente se presenta la planificación seguida en el presente proyecto.

7.1. Conclusiones del proyecto

En la consecución del proyecto, se puede afirmar que se ha conseguido satisfacer de manera exitosa los objetivos marcados inicialmente en el capítulo 3. Basándose en los resultados obtenidos en los diferentes experimentos realizados, el sistema desarrollado ha logrado mejorar sustancialmente la experiencia vivida por los usuarios participes en un juego a través de un dispositivo móvil, introduciéndoles en un mundo virtual basado en el real. Además se simplifica y facilita sustancialmente la labor de administración de las diferentes partidas disponibles, reduciendo la cantidad de recursos necesarios.

Tras la finalización de manera definitiva del presente Proyecto Fin de Carrera, es el momento adecuado de hacer balance repasando la trayectoria seguida para la obtención de las conclusiones, tanto en el aspecto técnico como en el ámbito personal.

- En el desarrollo del sistema propuesto se ha conseguido ampliar el conocimiento de mi etapa de estudiante en la Universidad Rey Juan Carlos, con las tecnologías y herramientas necesarias para la correcta finalización del proyecto, la mayoría

de las cuales eran desconocidas. Al llevar a cabo este proyecto se han adquirido nociones sobre la programación para terminales móviles, teniendo en cuenta sus limitaciones tanto de memoria como de procesamiento, así como las diferentes formas de conectividad. Se ha aprendido un nuevo lenguaje de programación, Python, debido a la propuesta del tutor en utilizar el *framework* Django, el cual también se desconocía y ha sido necesario su aprendizaje. Además se han asimilado diferentes librerías (LibreGeoSocial, entre otros) y lenguajes para transmitir información en el sistema (JSON). Para esto, ha sido necesario la búsqueda y lectura de libros, manuales y ejemplos relacionados con los aspectos citados.

- La creación de un sistema con un funcionamiento estable y eficaz, desarrollando un sitio web con el que se facilita la administración a los Organizadores de los diferentes recursos asociados a una partida, mediante una interfaz web fácil y sencilla. Además de permitirles realizar en tiempo real el seguimiento de las partidas en curso para guiar en caso de que lo necesiten a los participantes a través del servicio de mensajería interno.

  También se ha implementado correctamente una aplicación para los terminales Android, cumpliendo con otro de los objetivos marcados inicialmente, con la cual los usuarios pueden interactuar con las partidas disponibles. Poniendo a su disposición una interfaz intuitiva para la realización de las diferentes acciones, lo que facilita su rápido aprendizaje e inmersión en el juego.

- Cumpliendo con uno de los objetivos marcados, se han realizado diferentes pruebas a lo largo de todo el desarrollo del sistema para aumentar su fiabilidad y estabilidad. Para esto ha sido fundamental las dos últimas pruebas realizadas con voluntarios independientes al proyecto, así como la importancia de las opiniones capturadas en las encuestas. Mencionar que una vez analizados los resultados obtenidos, se puede concluir con una muy positiva valoración en todos los aspectos y un alto grado de aceptación por parte de los usuarios de la aplicación. Quedando verificada su madurez y funcionalidad en diferentes escenarios, aunque algunos de los aspectos se podrían mejorar sensiblemente.

Al margen de los objetivos marcados inicialmente y las conclusiones obtenidas, en la consecución de este proyecto la tarea más complicada, que ha resultado ser bastante laboriosa, es el control de concurrencia. Ya que ha sido necesario la actualización del *framework* Django para la utilización de las nuevas funciones y facilitan esta labor. Además fue necesario implementar el mecanismo con el cual detectarla, tanto en el

servidor como en el cliente, aceptando solo una de las peticiones realizadas simultáneamente y descartando el resto de estas, manteniendo siempre informado al usuario.

Finalmente, calificar de un gran éxito todo el trabajo realizado durante los diferentes años de estudio que me han conseguido instruir profesionalmente como en lo personal. Todo esto me llena de satisfacción para terminar este ciclo de mi vida y comenzar uno nuevo que me deparará nuevos retos.

7.2. Futuras líneas de trabajo

El proyecto realizado presenta diferentes posibilidades, pudiendo actuar como una posible fuente de ideas para proyectos similares. El carácter geolocalizado de este sistema está en pleno auge, lo que permite la mejora de este. A continuación se presentan algunas de las ideas.

- Facilitar aún más y ofrecer un mayor soporte a los jugadores mediante una aplicación Android específica para los Organizadores. Esta posibilitaría realizar todas las funciones que están a su disposición en el actual servicio web pero sin estar inmovilizado en un ordenador, pudiendo estar en continuo movimiento. Proporcionaría una mayor atención a los participantes, favoreciendo el desplazamiento del Organizador a sus posiciones para resolver las dudas o guiarles en caso que lo necesiten. Actualmente el sitio web es accesible mediante cualquier navegador disponible en el móvil, pero la agilidad y rendimiento que presenta respecto a la posible implementación de una aplicación es bastante inferior.

- Incluir en la aplicación Android un completo sistema de autentificación, basado en los miembros pertenecientes a la red social móvil LibreGeoSocial. Esto sería bastante útil para que los participantes utilizase un nombre de usuario único en cada evento, y así facilitar la relación de los miembros para la organización y quedada en futuras partidas. Además, con esta medida se facilita la creación de un *ranking* con los principales valores que los usuarios han obtenido en las diferentes partidas jugadas, como pueden ser los números de victorias, derrotas o propiedades en su poder, etc. De esta forma se incentiva más la competitividad entre ellos, ya que al conocer la información que proporciona la red social móvil, posibilita la comunicación con los mejores participantes para retarles a una nueva partida.

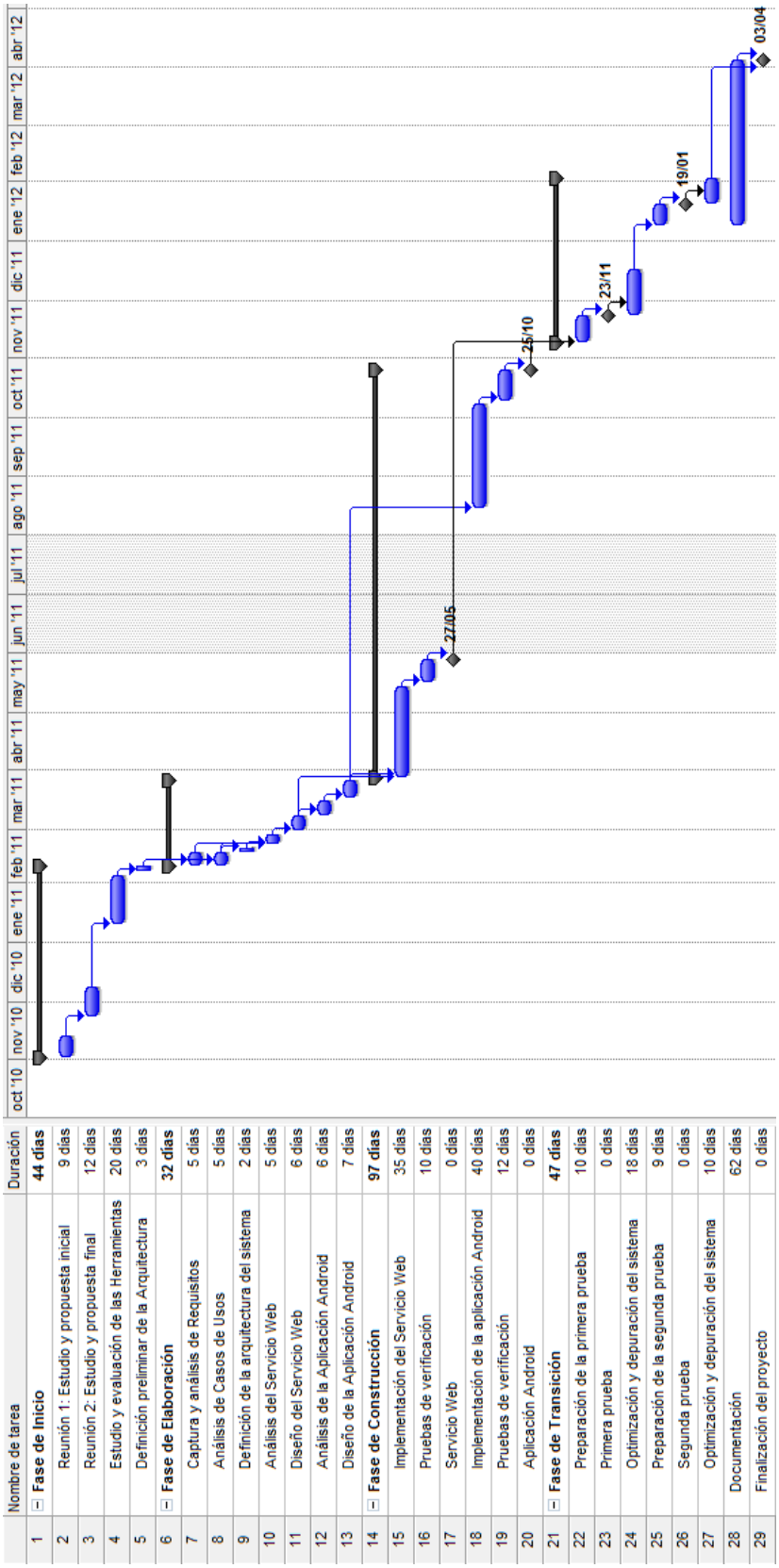
- Continuar mejorando y ampliando el sistema desarrollado ofreciendo un mayor abanico de posibilidades, con las cuales mejorar la experiencia de los diferentes participantes. Una posible propuesta es la introducción de diferentes eventos producidos, como puede ser un coste económico, por el paso de un participante junto a una propiedad rival. Complicando bastante más la lógica se podría ampliar el escenario a una ciudad entera, diseñando un juego en tiempo real sin limitación de tiempo. Creando un mundo virtual permanente, compitiendo con los demás usuarios, accesible únicamente desde el terminal móvil. Esto implica una continua evolución de las partidas cuando el jugador no esté conectado, ofreciéndole diferentes posibilidades en cada ocasión. Esto implicaría profundos cambios en el actual sistema.

- Crear un modo para un único jugador para no depender constantemente de los demás usuarios. Esto implicaría la creación de bots, es decir, un sistema que realice las diferentes funciones imitando el comportamiento de cualquier jugador. De esta forma se le ofrece la posibilidad de practicar y probar previamente sus estrategias para las partidas con jugadores reales.

- Difusión de la aplicación creada a través de Google Play Store [11] (conocido anteriormente como Android Market), para promocionar el generó geolocalizado, así como la red social móvil LibreGeoSocial. De esta forma mostrar a los demás usuarios las numerosas posibilidades que ofrece este tipo de juego.

7.3. Planificación

En este apartado se muestra la figura 7.1, con el *Diagrama de Gantt* con la planificación que se ha realizado para la consecución del proyecto. En este diagrama se especifican las fases seguidas de la metodología (Proceso Unificado) con las principales tareas llevadas a cabo en cada una de estas, así como la duración de las mismas. También se establecen los principales hitos conseguido en el transcurso del presente proyecto.



	Nombre de tarea	Duración
1	**Fase de Inicio**	**44 días**
2	Reunión 1: Estudio y propuesta inicial	9 días
3	Reunión 2: Estudio y propuesta final	12 días
4	Estudio y evaluación de las Herramientas	20 días
5	Definición preliminar de la Arquitectura	3 días
6	**Fase de Elaboración**	**32 días**
7	Captura y análisis de Requisitos	5 días
8	Análisis de Casos de Usos	5 días
9	Definición de la arquitectura del sistema	2 días
10	Análisis del Servicio Web	5 días
11	Diseño del Servicio Web	6 días
12	Análisis de la Aplicación Android	6 días
13	Diseño de la Aplicación Android	7 días
14	**Fase de Construcción**	**97 días**
15	Implementación del Servicio Web	35 días
16	Pruebas de verificación	10 días
17	Servicio Web	0 días
18	Implementación de la aplicación Android	40 días
19	Pruebas de verificación	12 días
20	Aplicación Android	0 días
21	**Fase de Transición**	**47 días**
22	Preparación de la primera prueba	10 días
23	Primera prueba	0 días
24	Optimización y depuración del sistema	18 días
25	Preparación de la segunda prueba	9 días
26	Segunda prueba	0 días
27	Optimización y depuración del sistema	10 días
28	Documentación	62 días
29	Finalización del proyecto	0 días

Figura 7.1: Diagrama de Gantt con la planificación del presente proyecto

Bibliografía

[1] Android. Sitio web. http://www.android.com/. Último acceso: Abril 2012.

[2] Android Developers. Api de Android. http://developer.android.com/reference/packages.html. Último acceso: Abril 2012.

[3] Android Developers. Documentación para desarrolladores de Android. http://developer.android.com/guide/basics/what-is-android.html. Último acceso: Abril 2012.

[4] Apache Software Foundation. Sitio web de Apache HTTP Server Project. http://httpd.apache.org/. Último acceso: Abril 2012.

[5] BigShot. Sitio web. http://bigshot.mobi/. Último acceso: Abril 2012.

[6] Django. Sitio web. https://www.djangoproject.com/. Último acceso: Abril 2012.

[7] DjangoSites. Deployment Statistics. http://www.djangosites.org/stats/. Último acceso: Abril 2012.

[8] Eclipse. Sitio web. http://www.eclipse.org/. Último acceso: Abril 2012.

[9] Google Developers. Documentación de Google Maps. https://developers.google.com/maps/?hl=es. Último acceso: Abril 2012.

[10] Google Developers. Obtención de la Api Key para de Google Maps. https://developers.google.com/maps/signup?hl=es. Último acceso: Abril 2012.

[11] Google Inc. Sitio web de Google Play Store. https://play.google.com/store. Último acceso: Abril 2012.

[12] GSyC. Sitio web. http://gsyc.escet.urjc.es/. Último acceso: Abril 2012.

[13] Guido van Rossum. Personal History part1 CWI. http://python-history.blogspot.com.es/2009/01/personal-history-part-1-cwi.html. Último acceso: Abril 2012.

[14] Hasbro. Sitio web de Monopoly. http://libresoft.es/. Último acceso: Abril 2012.

[15] Ian Sommerville. *Ingenieria Del Software*. Pearson Educación, 2005.

[16] Ivar Jacobson, Grady Booch, and James Rumbaugh. *El proceso unificado de desarrollo de software*. Addison Wesley, 2004.

[17] Jacob Keplan-Moss and Adrian Holovaty. *The Definitive Guide to Django: Web Development Done Right*. Apress, 2009.

[18] Jorge Fernández González. Sitio web de LibreSoft Gymkhana. http://gymkhana.libresoft.es/indice.html. GSyC/LibreSoft. URJC. Último acceso: Abril 2012.

[19] Joshua Drake and John Worsley. *Practical PostgeSQL*. O'Reilly Media, 2011.

[20] JSON. Documentación. http://www.json.org/. Último acceso: Abril 2012.

[21] LibreGeoSocial. Documentación. http://libregeosocial.morfeo-project.org/wiki/index.php/Docs_LibreGeoSocial-OMF. Último acceso: Abril 2012.

[22] LibreGeoSocial. Sitio web. http://www.libregeosocial.org/. Último acceso: Abril 2012.

[23] LibreSoft. Sitio web. http://libresoft.es/. Último acceso: Abril 2012.

[24] mod_wsgi. Sitio web. http://code.google.com/p/modwsgi/. Último acceso: Abril 2012.

[25] Moodle URJC. Sitio web. http://docencia.etsit.urjc.es/moodle/. Último acceso: Abril 2012.

[26] Peter Rob and Carlos Coronel. *Sistemas de bases de datos:Diseño, implementación y administración*. Cengage Learning Editores, 2003.

[27] PostGis. Sitio web. http://postgis.refractions.net/documentation/. Último acceso: Abril 2012.

[28] PostgreSQL. Sitio web. http://www.postgresql.org.es/. Último acceso: Abril 2012.

[29] Python. Sitio web. http://www.python.org/. Último acceso: Abril 2012.

[30] Python. The Python Wiki. http://wiki.python.org/moin/. Último acceso: Abril 2012.

[31] Sayed Hashimi, Satya Komatineni, and Dave MacLean. *Pro Android 2*. Appres, 2010.

[32] Sergio Liján Mora. *Programación en Internet: Clientes Web*. Club Universitario, 2001.

[33] World Wide Web Consortium (W3C). Documentación de REST. http://www.w3.org/2001/sw/wiki/REST. Último acceso: Abril 2012.

[34] World Wide Web Consortium (W3C). Sitio web. http://www.w3.org/. Último acceso: Abril 2012.

[35] World Wide Web Consortium (W3C). Sitio web de CSS. http://www.w3.org/Style/CSS/. Último acceso: Abril 2012.

[36] XCubeLabs. Infografía de la historia de Android. http://www.xcubelabs.com/the-android-story.php. Último acceso: Abril 2012.

Apéndice A

Encuestas de las pruebas

A continuación se recogen las diversas opiniones realizadas por los voluntarios que participaron en las diferentes pruebas realizadas para la depuración y mejora del sistema implementado.

A.1. Primera encuesta: Resultados

Respuestas enviadas: 3
Preguntas: 36

1.- Valore la explicación que se ha realizado al comienzo sobre el funcionamiento del terminal y del juego.

- Mala: 0
- Regular: 0
- Buena: 2 (66.67 %)
- Muy buena: 1 (33.33 %)

2.- ¿Le ha resultado necesaria la explicación para comprender el funcionamiento del juego y del terminal?

- Inneccesarias: 0
- Algo necesarias: 1 (33.33 %)
- Necesarias: 2 (66.67 %)

3.- ¿Ha consultado el menú de información que explica detalladamente las funcionalidades de la aplicación?

- No: 2 (66.67 %)
- Sí: 1 (33.33 %)

4.- En caso de haber respondido afirmativamente, valore la información mostrada sobre las funcionalidades de la aplicación.

- Mala: 0
- Regular: 0
- Buena: 0
- Muy buena: 0

5.- ¿Qué le parece el manejo de la interfaz gráfica de usuario (GUI)?

- Sencilla: 2 (66.67 %)
- Normal: 1 (33.33 %)
- Complicada: 0
- Muy complicada: 0

6.- ¿Le parece adecuado el uso que se hace de las barras de progreso para evitar la sensación de un posible error?

- Innecesarias: 0
- Algo necesarias: 2 (66.67 %)
- Necesarias: 1 (33.33 %)

7.- Valore la GUI de la pestaña estadísticas.

- Mala: 0
- Regular: 1 (33.33 %)
- Buena: 0
- Muy buena: 2 (66.67 %)

8.- En caso de haber respondido negativamente (Mala/Regular), ¿qué cambiaría en la pestaña estadísticas?

-
-
- Compactar más la información para ver mejor la información de todos los equipos participantes.

9.- Valore la GUI de la pestaña cartas (incluida la pantalla con el listado, la visualización detallada de una carta y las notificaciones mostradas).

- Mala: 0
- Regular: 0
- Buena: 3 (100.00 %)
- Muy buena: 0

10.- En caso de haber respondido negativamente (Mala/Regular), ¿qué cambiaría en la pestaña cartas?

-
-
-

11.- Valore la GUI de la pestaña mapa (incluidas las notificaciones y forma de realizar las acciones).

- Mala: 0
- Regular: 1 (33.33 %)
- Buena: 1 (33.33 %)
- Muy buena: 1 (33.33 %)

12.- En caso de haber respondido negativamente (Mala/Regular), ¿qué cambiaría en la pestaña mapa?

-
-
- Añadir algún elemento para mejorar la orientación del usuario

13.- Valore la GUI de la pestaña ofertas (incluidas las pantallas con el listado, la visualización detallada de una oferta y las notificaciones mostradas).

- Mala: 0
- Regular: 0
- Buena: 2 (66.67 %)
- Muy buena: 1 (33.33 %)

14.- En caso de haber respondido negativamente (Mala/Regular), ¿qué cambiaría en la pestaña ofertas?

-
-
-

15.- Valore la GUI de la pestaña mensajes (incluidas las pantallas con el listado, la visualización detallada, envío de un nuevo mensaje y notificaciones mostradas).

- Mala: 0
- Regular: 0
- Buena: 2 (66.67 %)
- Muy buena: 1 (33.33 %)

16.- En caso de haber respondido negativamente (Mala/Regular), ¿qué cambiaría en la pestaña mensajes?

-
-
-

17.- ¿Qué le parece el área de acción para realizar las diferentes acciones sobre las diferentes propiedades?

- Pequeña: 1 (33.33 %)
- Normal: 2 (66.67 %)
- Grande: 0

18.- ¿Qué le parece el aviso acústico y mediante el led de la notificación de un nuevo mensaje u oferta en la barra de estado de la interfaz del móvil?

- Malo: 0
- Regular: 0
- Bueno: 3 (100.00 %)
- Muy bueno: 0

19.- Valore la frecuencia con la que se van repartiendo las cartas.

- Baja: 0
- Normal: 0
- Alta: 3 (100.00 %)

20.- Valore la precisión de su localización en la pestaña mapa.

- Mala: 0
- Regular: 0
- Buena: 2 (66.67 %)
- Muy buena: 1 (33.33 %)

21.- Valore la frecuencia con la que se van suministrando las bonificaciones de las calles en su posesión.

- Baja: 0
- Normal: 3 (100.00 %)
- Alta: 0

22.- ¿Se produjo algún error o cierre inesperado durante la ejecución de la aplicación?

- No: 2 (66.67 %)
- Sí: 1 (33.33 %)

23.- En caso de haber respondido afirmativamente, ¿qué acción estaba realizando en ese momento?

- No estaba realizando ninguna acción en particular, estaba mirando el mapa mientras caminaba.
-
-

24.- Valore el consumo de la batería durante la partida.

- Bajo: 0
- Normal: 1 (33.33 %)
- Alto: 0
- No me he fijado: 2 (66.67 %)

25.- Valore la estética general de la aplicación (textos, colores, iconos, notificaciones, etc.).

- Mala: 0
- Regular: 0
- Buena: 2 (66.67 %)
- Muy buena: 1 (33.33 %)

26.- Valore el rendimiento general de la aplicación

- Malo: 0
- Regular: 0
- Bueno: 3 (100.00 %)
- Muy bueno: 0

27.- Valore la fiabilidad general de la aplicación.

- Mala: 0
- Regular: 0
- Buena: 3 (100.00 %)
- Muy buena: 0

28.- Valoración media de la aplicación.

- Mala: 0
- Regular: 0
- Buena: 3 (100.00 %)
- Muy buena: 0

29.- ¿Ha participado en algún juego parecido a este?

- No: 1 (33.33 %)
- Sí: 2 (66.67 %)

30.- En caso de haber respondido afirmativamente, detalle y explique brevemente el juego.

- Si, en un juego con "geolocalización" del mismo estilo con ejercitos y zonas a conquistar.
- Trataba de un juego de conquista de territorios Geolocalizado
-

31.- Indique la característica que más le ha gustado del juego.

- La distintas posibilidades que hay en relación a las calles y propiedades, se puede comprar, puedes demoler, puedes sabotear al rival, hacer ofertas...
- La mecánica del juego
- Ir a tus territorios y edificar en ellos.
  También demoler los sabotajes de los rivales.

32.- Indique la caracerística que menos le ha gustado del juego.

- Demasiada frecuencia de Cartas y el posible excesivo castigo al inicio con las cartas de perdida de dinero.
- El campo de acción y los iconos de las calles
- Al principio costaba ganar dinero para poder comprar más terrenos y poder edificar.
  Las cartas aparecían con mucha frecuencia.

33.- ¿Añadiría alguna acción que se realizase sobre los diferentes equipos?

- No: 0
- Sí: 3 (100.00 %)

34.- En caso de haber respondido afirmativamente, detalle y explique la propuesta.

- La posibilidad de vender la propiedad por un precio fijo pero sin vendérselo a ningún oponente, simplemente para recuperar el dinero invertido aunque menos de lo que costo...
- Que los jugadores puedan ser encerrados en la cárcel por malversación de fondos
- Poder vender una calle sin que un equipo haga previamente una oferta.

35.- ¿Qué eliminarías, cambarías o añadirías al juego?

- Las notificaciones sobre cartas nuevas quizás molesten, debería informarte sin tener que pararte a aceptar la notificación.
- Aumentaría el campo de acción, añadiría a los iconos el propietario al que pertenece
- Mejorar la orientación en la pestaña mapa.
  Identificar mejor los distintos iconos de los terrenos.

36.- Finalmente, escriba cualquier comentario o sugerencia no recogida en la encuesta, o matizaciones a alguna de las respuestas que has realizado.

-
- Ha sido una experiencia muy entretenida y me gustaría participar en otras ocasiones.
-

A.2. Segunda encuesta: Resultados

Respuestas enviadas: 5
Preguntas: 20

1. Valore la explicación que se ha realizado al comienzo sobre el funcionamiento del terminal y del juego.
- Mala: 0
- Regular: 1 (20.00 %)
- Buena: 3 (60.00 %)
- Muy buena: 1 (20.00 %)

2. ¿Le ha resultado necesaria la explicación para comprender el funcionamiento del juego y del terminal?
- Innecesarias: 0
- Algo necesario: 0
- Necesaria: 5 (100.00 %)
- Muy necesarias: 0

3. ¿Qué le parece el área de acción para realizar las diferentes acciones sobre las diferentes propiedades?
- Pequeña: 0
- Correcta: 5 (100.00 %)
- Grande: 0

4. ¿Qué le parece las notificaciones de la recepción de un nuevo mensaje y oferta en la barra de estado de la interfaz móvil?
- Mala: 0
- Regular: 0
- Buena: 2 (40.00 %)
- Muy buena: 3 (60.00 %)

5. Valore la frecuencia con la que se van suministrando las bonificaciones de las calles en su posesión.
- Baja, pocas bonificaciones: 1 (20.00 %)
- Correcta: 0
- Alta, muchas bonificaciones: 4 (80.00 %)

6. Valore la frecuencia con la que se van repartiendo las cartas.
- Baja, pocas cartas: 0
- Correcta: 1 (20.00 %)
- Alta, muchas cartas: 4 (80.00 %)

7. ¿Se produjo algún error o cierre inesperado durante la ejecución de la aplicación?
- No: 5 (100.00 %)
- Sí: 0

8. En caso de haber respondido afirmativamente, detalle la acción que estaba realizando en ese momento.
-
-
-
-
-

9. Valore la precisión de su localización en la pestaña mapa.
- Mala: 0
- Regular: 3 (60.00 %)
- Buena: 2 (40.00 %)
- Muy buena: 0

10. Valore de forma general, el manejo de la interfaz gráfica de usuario (GUI).

- Sencilla: 5 (100.00 %)
- Normal: 0
- Complicada: 0
- Muy complicada: 0

11. Valore de forma general, el consumo de batería durante una partida.

- Bajo: 1 (20.00 %)
- Normal: 1 (20.00 %)
- Alto: 0
- No me he fijado: 3 (60.00 %)

12. Valore de forma general, el diseño de la aplicación (textos, colores, iconos, notificaciones, etc.).

- Mala: 0
- Regular: 1 (20.00 %)
- Buena: 2 (40.00 %)
- Muy buena: 2 (40.00 %)

13. Valore de forma general, el rendimiento de la aplicación (ralentizaciones, velocidad de ejecución en las acciones, etc.).

- Malo: 0
- Regular: 0
- Bueno: 4 (80.00 %)
- Muy bueno: 1 (20.00 %)

14. Valore de forma general, la fiabilidad de la aplicación (errores, cierres inesperados, bugs, etc.).

- Mala: 0
- Regular: 0
- Buena: 3 (60.00 %)
- Muy buena: 2 (40.00 %)

15. Valore de forma general, la aplicación a nivel de usuario.

- Mala: 0
- Regular: 0
- Buena: 4 (80.00 %)
- Muy buena: 1 (20.00 %)

16. Tras la experiencia de haber participado en un grupo junto a otros compañeros en un mismo dispositivo, ¿de cuantos integrantes considera que deberían de estar formado los grupos?

- 1: 1 (20.00 %)
- 2: 4 (80.00 %)
- 3: 0
- 4: 0
- Más de 4: 0

17. ¿Repetiría la experiencia de volver a participar en este juego?

- No: 0
- Probablemente no: 0
- Probablemente sí: 4 (80.00 %)
- Seguro que sí: 1 (20.00 %)

18. Indique las características que más le han gustado de juego.

- Interfaz gráfica, sencillez de utilización.
- Bastante movimiento
  Intuitivo (reglas fáciles)
  Libertad de acción
  Buen uso de geolocalización
- la competitividad que se genera
- La competitividad que se genera por la capacidad de fastidiar directamente a un rival. Permite una partida bastante fluída, ya que no te da tiempo a esperar a que cambien las condiciones de la partida.
- La competitividad entre los equipos.

19. Indique las características que menos le han gustado del juego.

- Reparto de cartas.
- Balanceado regular
  Mensajes del mapa demasiado persistentes, sin posibilidad de ocultar tras verlos
- una vez construidos los edificios no se podia ofrecer nada por ellos, no por el programa, sino por los equipos que no aceptaban las ofertas
- A veces resultaba confuso porque no sabías a la hora de realizar acciones en un edificio si había cambiado recientemente su estatus
- Cuando ofertabas por una construcción los otros equipos no aceptaban la oferta.

20. Finalmente, escriba cualquier comentario, sugerencia no recogida en la encuesta o matizaciones de alguna de las respuestas que has realizado.

- Quizás se debería dar la opción de comprar cartas como demolición o sabotaje, aunque penalizándolo con un coste alto de dinero.
- Las pantallas de información de equipos y territorios tardan demasiado en ocultar, llegándose al caso de cambiar de pantalla y seguir mostrando mensajes durante un tiempo. Recomendaría o reducir el tiempo (prefiero darle dos veces a esperar) o introducir algún sistema que obligue a ocultarlo (un botón o algo).
-
-
-