ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Diseño, Implementación y Despliegue de un Servicio de Telecomunicación para M-Learning en Móviles Android: Gymkhanas de Nueva Generación

Autor: Jorge Fernández González
Tutor: Gregorio Robles Martínez

Curso académico: 2009/2010

Proyecto Fin de Carrera
Diseño, implementación y despliegue de un servicio de telecomunicación para m-learning en móviles Android: gymkhanas de nueva generación

Autor
Jorge Fernández González

Tutor
Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de , siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de

A mis padres.

El corazón tiene razones
que la razón no entiende.

Blaise Pascal

Agradecimientos

Muchas son las horas de trabajo que quedan atrás cada vez que una persona alcanza cada meta que se ha impuesto, cayendo a menudo todo ese esfuerzo en el olvido, especialmente al resumirse el tiempo dedicado en un tomo de unas cuantas páginas. Pero hay algo que nunca podrá caer en el olvido: todas y cada una de las personas que han colaborado poniendo su granito de arena, con cada pequeño empujón, para que lograse alcanzar el objetivo marcado.

Durante la realización de este Proyecto Fin de Carrera he tenido la suerte de estar rodeado de un nutrido grupo de personas de las que me siento orgulloso y a las cuales quiero mostrar ahora mi agradecimiento. Comenzando por Gregorio Robles Martínez, mi tutor, por ofrecerme su idea como proyecto y por guiarme durante la realización del mismo. También Pedro de las Heras Quirós, por cedernos los teléfonos cada vez que los necesitábamos y sin lo cual, el proyecto no habría sido lo mismo. A los compañeros que he tenido durante mi efímero paso por *GSYC/LibreSoft*, especialmente a José Antonio Santos, por su ayuda con el servidor de *LibreGeoSocial*, a Raúl Román, también buen compañero de clase, y a Roberto Calvo, por su amabilidad y su ayuda siempre que lo necesitaba. E igualmente, a todos los voluntarios que participaron en los experimentos de manera desinteresada, dedicándonos su tiempo y ofreciéndonos una información muy valiosa que nos permitió perfeccionar continuamente el proyecto.

Pero este Proyecto Fin de Carrera no es más que la culminación a tantos años de estudio, por lo que quiero hacer extensivo mi agradecimiento a los extraordinarios compañeros que he tenido durante este Segundo Ciclo y que han hecho la experiencia mucho más corta, llevadera y enriquecedora. Entre todos ellos debo destacar a Ángel, por su complicidad, y a Marta, por su generosidad y por su genuina forma de ser. Y muy especialmente a Marga, mi excelente compañera de clase y de prácticas; por tantas horas de charlas con sus buenos y sus malos momentos; por *el comienzo de una hermosa amistad*. Pero también debo recordar a José Luis, Manu, Rico y Humberto, junto a los cuales inicié el camino universitario por nuestra querida Telemática.

Por supuesto, también a mis padres, por toda una vida de sacrificio para que pudiera llegar hasta aquí y por su confianza plena en mí. A mis hermanos. A todos aquellos que siempre me apoyaron y creyeron en mis posibilidades.

Por último, también a ti, querido lector, tanto si buscas algo en concreto entre estas páginas y esperando que encuentres algo que te resulte de utilidad e interés, como si tu nombre debía aparecer entre estas líneas pero has sido víctima de un desafortunado descuido.

Resumen

En una sociedad cada vez más tecnológica, debido a la convergencia de las redes de telefonía móvil y las conexiones a redes de datos, se está produciendo una gran revolución dentro del ámbito de las telecomunicaciones. La irrupción en el panorama social y comercial con gran éxito y acogida de las tecnologías propias de la llamada *web 2.0*, unida a la aparición en el mercado de sofisticados teléfonos móviles denominados *smartphones*, que integran dispositivos de muy diversa índole (*GPS*, múltiples sensores, cámara fotográfica, interfaces de red, etc.), gestionados por los sistemas operativos más avanzados, hacen que las aplicaciones y servicios telemáticos cobren mayor relevancia e interés que nunca. Estos avances tecnológicos suponen el mejor caldo de cultivo para explotar las propiedades más ventajosas de los juegos tradicionales, hasta el punto de ofrecer una nueva perspectiva mucho más potente y atractiva, como pueda ser mediante la orientación de los contenidos hacia el aprendizaje a través de los métodos y técnicas del *m-learning* (*mobile-learning*), o por ejemplo, redescubriendo nuevas formas alternativas de hacer turismo. Es decir, conseguir una nueva dimensión en la que el usuario, la persona, se enriquezca por medio de la combinación de tecnología y mundo tradicional con sus importantes componentes y valores sociales e interpersonales.

Dentro del contexto establecido, en este Proyecto Fin de Carrera se ha procedido al diseño e implementación de un servicio de telecomunicaciones que además de facilitar la organización y control de *gymkhanas*, revoluciona la experiencia vivida por los participantes, introduciendo elementos impensables para una *gymkhana* tradicional como pueda ser la comunicación inmediata entre participantes y organizadores por medio de un servicio de mensajería interno, o pruebas basadas en el uso de *GPS*.

Partiendo de un diseño del sistema basado en una arquitectura cliente-servidor, la estructura final es de un servidor *web* interaccionando con una base de datos y comunicándose por medio del protocolo *HTTP* con dos tipos distintos de clientes, pudiéndose codificar los datos intercambiados en diversos formatos: *HTML*, *JSON*, *XML*, etc. El primer cliente se refiere a la interfaz *web* a través de la cual el organizador creará la *gymkhana* y monitorizará el transcurso de la misma (ubicación de equipos, respuestas, mensajes, etc.). El segundo cliente es el relativo a los participantes de la *gymkhana*, constituido por una aplicación *Android* para cualquier teléfono móvil con dicho sistema operativo de *Google* y cuya comunicación con el servidor se realizará, normalmente, vía *UMTS*. Además, en la última fase del proyecto se tiene el valor añadido de haber desplegado y probado el correcto funcionamiento del servicio en un escenario real con teléfonos *HTC Magic* de *Vodafone*, obteniendo en base a las opiniones de los asistentes a los experimentos un *feedback* que permite mejorar continuamente la calidad del servicio ofrecido.

Acrónimos

API	***A****pplication* ***P****rogramming* ***I****nterface*
AJAX	***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML*
CSS	***C****ascading* ***S****tyle* ***S****heets*
DOM	***D****ocument* ***O****bject* ***M****odel*
GSM	***G****lobal* ***S****ystem for* ***M****obile Communications*, originalmente ***G****roupe* ***S****pécial* ***M****obile*
GPS	***G****lobal* ***P****ositioning* ***S****ystem*
HTML	***H****yper****T****ext* ***M****arkup* ***L****anguage*
HTTP	***H****yper****T****ext* ***T****ransfer* ***P****rotocol*
IDE	***I****ntegrated* ***D****evelopment* ***E****nvironment*
IMAP	***I****nternet* ***M****essage* ***A****ccess* ***P****rotocol*
IP	***I****nternet* ***P****rotocol*
JDK	***J****ava* ***D****evelopment* ***K****it*
JSON	***J****ava****S****cript* ***O****bject* ***N****otation*
JVM	***J****ava* ***V****irtual* ***M****achine*
MVC	***M****odel-****V****iew-****C****ontroller*
MVT	***M****odel-****V****iew-****T****emplate*
MTA	***M****ail* ***T****ransport* ***A****gent* o ***M****essage* ***T****ransport* ***A****gent*
REST	***RE****presentational* ***S****tate* ***T****ransfer*
RFC	***R****equest* ***F****or* ***C****omments*
SAX	***S****imple* ***A****PI for* ***X****ML*
SDK	***S****oftware* ***D****evelopment* ***K****it*
SMS	***S****hort* ***M****essage* ***S****ervice*
SMTP	***S****imple* ***M****ail* ***T****ransfer* ***P****rotocol*
SQL	***S****tructured* ***Q****uery* ***L****anguage*
TCP	***T****ransmission* ***C****ontrol* ***P****rotocol*
UML	***U****nified* ***M****odeling* ***L****anguage*
UMTS	***U****niversal* ***M****obile* ***T****elecommunications* ***S****ystem*
W3C	***W****orld* ***W****ide* ***W****eb* ***C****onsortium*
Wi-Fi	***Wi****reless* ***Fi****delity*
WWW	***W****orld* ***W****ide* ***W****eb
XHTML	*EX****T****ensible* ***H****yper****T****ext* ***M****arkup* ***L****anguage*
XML	*EX****T****ensible* ***M****arkup* ***L****anguage*
XMPP	*EX****T****ensible* ***M****essaging and* ***P****resence* ***P****rotocol*
XSL	*EX****T****ensible* ***S****tylesheet* ***L****anguage*
XSLT	***XSL*** ***T****ransformations*

Índice General

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

When something can be read without effort,
great effort has gone into its writing.

Enrique Jardiel Poncela

1.1. Motivación

Las *gymkanas* o *gymkhanas* son en la actualidad uno de los juegos más populares de entre aquellos que se realizan al aire libre con grandes grupos de personas. Su objetivo final es superar el mayor número posible de los retos propuestos, fomentándose ante todo el trabajo en equipo y la cooperación, pero nunca la competitividad.

Al mismo tiempo, la proliferación y perfeccionamiento de las nuevas tecnologías, como pueden ser los servicios de telecomunicaciones basados en telefonía móvil, los servicios *web*, los sistemas operativos, las conexiones de red y los terminales de comunicaciones más avanzados, abren un nuevo horizonte de cara a la optimización y mejora de estos juegos tradicionales, tanto a nivel organizativo como en lo referido a la experiencia que vivirá el participante.

Así, teniendo en cuenta la cada vez mayor importancia y relevancia de los contenidos ofrecidos en la sociedad de las tecnologías de la información y de las comunicaciones, ahora que la convergencia de las redes de datos y de telefonía es ya una realidad, se puede optar por la combinación de estos juegos tradicionales con las tecnologías más modernas. De hecho, en este caso de las *gymkhanas*, consiguiendo el enfoque adecuado en cada evento organizado, se puede lograr que aquello que en principio supone una alternativa de ocio, también se convierta, por ejemplo, en un nuevo canal de aprendizaje (*mobile learning* o *m-learning*) de la manera más amena posible.

De hecho, ese aprendizaje ya queda garantizado en el ámbito de las nuevas tecnologías, pues estas *gymkhanas* contarían con cierto carácter divulgativo dado que los participantes entrarían en contacto con tecnologías y dispositivos muy avanzados, que actualmente cuentan todavía con poca penetración entre el gran público, y dando a conocer al usuario todas las posibilidades que ofrecen esas tecnologías punteras del mercado que a día de hoy no quedan al alcance de todos los sectores sociales debido a su coste económico aparejado. O bien, se podrían abrir nuevas posibilidades y alternativas a la manera habitual de realizar rutas turísticas, de modo que un turista cualquiera, con tan sólo disponer de un teléfono móvil con conexión de datos, pudiera hacer uso del sistema, proponiéndole durante la *gymkhana* un recorrido por los puntos fundamentales de la ciudad, evitando seguramente aglomeraciones por los grupos, o generando cierto sentimiento de independencia y autonomía en el turista pese a encontrarse en un lugar desconocido. Y por supuesto, haciendo también gala del ya citado *m-learning* (por ejemplo, ante una visita a la ciudad de Toledo, además de realizar una ruta turística por los principales monumentos, se pueden dar a conocer aspectos sobre las tres culturas, sobre la Historia de España, etc.).

Del mismo modo, se podrían complementar y aumentar las posibilidades de la cada vez más exitosa actividad denominada como *geocaching*, en la que una persona esconde normalmente en algún paraje natural unos determinados artículos de escaso valor, invitando a otros miembros de la comunidad o foro a que lo encuentren a partir de sus coordenadas geográficas, y teniendo como fin último el de compartir una jornada rodeado de familiares y/o amigos disfrutando de la naturaleza. Igualmente, en la actualidad se suele acusar a las videoconsolas y a los ordenadores personales del alto índice de obesidad infantil, debido principalmente al sedentarismo que llegan a provocar. Precisamente ésta podría ser una buena manera de combatir ese factor, ofreciendo a los niños actividades mucho más atractivas por medio de las nuevas tecnologías con las que tan familiarizados se encuentran, y recurriendo realmente a un juego que no sólo reduciría el sedentarismo al poder introducir en su transcurso pruebas deportivas (al igual que otras con fines didácticos), sino que también fomentará sus relaciones sociales mejorando sus destrezas interpersonales como puedan ser la cooperación, el trabajo en equipo, el diálogo, los acuerdos y el consenso, etc.

Como primera aplicación de este proyecto en un futuro próximo, la *Universidad Rey Juan Carlos* desde hace varios años organiza una serie de recepciones y actividades destinadas a futuros alumnos con tal de darles a conocer cada campus de la universidad, diversas pruebas de introducción tecnológica, etc. Precisamente la elaboración de una *gymkhana* se ha considerado la manera idónea de hacer estas actividades mucho más atractivas y entretenidas, así como que esos futuros estudiantes se sientan más motivados de cara a la realización de los recorridos, así como integrados en la universidad que

visitan. Al mismo tiempo, se estaría generando en esos futuros estudiantes una buena imagen por parte de la *Escuela Técnica Superior de Ingeniería de Telecomunicación* y de la *Universidad Rey Juan Carlos* en general, pudiendo conllevar esto también una mayor atracción y captación de nuevos alumnos (particularmente en Ingeniería de Telecomunicación) provenientes de institutos y que visitasen la universidad, dando una muestra de entre las muchas posibles, de lo que cualquiera de ellos estaría capacitado para desarrollar como Proyecto Fin de Carrera una vez superados sus estudios.

Además, las ventajas que ofrecen las nuevas tecnologías de cara a la organización y realización de *gymkhanas*, frente a las formas más tradicionales de llevarlas a cabo, son inagotables. Se presentan a continuación algunos ejemplos:

- **Reducción de los recursos materiales.**

  Para la organización de la *gymkhana* serán necesarios menos recursos materiales. Ya no serán necesarias las clásicas tarjetas y papeles con los retos a superar, banderines y mapas para indicar el lugar al que los equipos deben llegar para realizar una determinada prueba, etc., lo que conllevará cierto ahorro económico en la organización del juego.

- **Reducción del tiempo de organización.**

  Esto primero también llevará aparejada una reducción considerable del tiempo necesario para la organización de la *gymkhana*, pues tan sólo se necesitará que una vez pensadas y elaboradas las pruebas, sean introducidas en la base de datos del servidor del sistema por medio de una sencilla interfaz *web*, sin necesidad de colocar los enunciados de las pruebas en determinados lugares, pistas, banderines, etc. O por ejemplo, en una *gymkhana* a través de una ciudad desconocida por parte de los participantes, no será necesaria la entrega de mapas puesto que ya dispondrán de ello en el terminal móvil.

- **Reducción de los recursos humanos.**

  También se necesitará un menor número de personas que a modo de jueces controlen las acciones de los equipos, validen las respuestas de cada uno para dar paso a la siguiente etapa, etc., puesto que todo ello se podrá realizar desde un punto central en el que la organización ejecutará un seguimiento del evento en curso.

- **Aumento del control sobre los equipos.**

  Se podrá tener un mayor control de los equipos en todo momento, gracias a que se dispondrá de la ubicación geográfica de cada uno, una comunicación inmediata para la realización de cualquier comentario o indicación, las respuestas que ofrezcan, etc. Lo que a su vez desemboca

en un fácil seguimiento de los participantes y con un menor coste, ofreciendo por ejemplo pistas específicas para la resolución de una prueba a un equipo que consume mucho más tiempo del estimado para superarla, o si por equívoco se alejan demasiado de las localizaciones preparadas para el transcurso de la *gymkhana*.

- **Supresión de desplazamientos innecesarios.**

  Se evitarán desplazamientos innecesarios a los organizadores y a los equipos, puesto que podrán dar su respuesta a un reto desde cualquier lugar y en cualquier momento, sin necesidad de buscar a un juez de la *gymkhana*. Todo ello se traducirá en un mejor aprovechamiento del tiempo, puesto que los equipos estarán intentando superar pruebas continuamente sin espacios de tiempo vacíos.

- **Comunicación permanente, inmediata y a distancia.**

  En consonancia con la reducción del número de personas necesarias para organizar la *gymkhana*, y con la reducción de desplazamientos improductivos dentro de la misma, se tiene también que ante cualquier duda o problema que un equipo pueda encontrar en una determinada prueba, podrá ponerse en contacto con los organizadores en todo momento, consiguiendo una comunicación mucho más rápida ante posibles dudas, consultas, etc.

- **Modificación dinámica de la *gymkhana* en curso.**

  Se podrá disponer de una creación/supresión de pruebas de manera dinámica durante el propio transcurso de la *gymkhana*. Útil por ejemplo si se observa que para la superación de una prueba, los equipos consumen demasiado tiempo, por lo que quizás fuera necesario insertar una prueba que facilite su resolución a modo de puente, o incluso, simplemente eliminarla.

- **Ventajas específicas sobre *gymkhanas* tradicionales de una determinada temática.**

  Igualmente se presentan múltiples ventajas respecto a tipos específicos de *gymkhanas*. Por ejemplo, en una *gymkhana* fotográfica, se dispondrá de inmediatez en el envío de las fotografías con que cada grupo responde a cada pregunta, y una validación de las respuestas en tiempo real, mientras que de una manera más tradicional, sería necesario esperar a la llegada de los equipos al punto de reunión, revelado de fotos o en el mejor de los casos, copia de las fotos desde la memoria de la cámara al ordenador de la organización, posterior corrección, etc. O en una *gymkhana* en la que los grupos vayan recorriendo una serie de emplazamientos donde se les haga una determinada demostración

o deban realizar un juego, se pueden minimizar los tiempos de espera para el resto de equipos, haciendo que se dirijan a otras ubicaciones mientras el equipo del primer emplazamiento termina.

- **Se presentan infinitas posibilidades aún por explorar por los creativos con experiencia en la elaboración de *gymkhanas*.**

Así pues, teniendo a nuestra disposición una serie de recursos como son la nueva generación de terminales móviles (por ejemplo, *HTC Dream* (*G1*), *HTC Magic* (G2) o *Nexus One Phone*), que cuentan con la innovadora plataforma *Android* de *Google* y con unas capacidades aún por explotar, unido a las ya conocidas bondades de los servicios *web* y de la *web 2.0*, se configurará la herramienta perfecta para la celebración de este tipo de eventos de una manera mucho más innovadora, divertida y provechosa, mejorando con creces y ofreciendo ventajas respecto a las *gymkhanas* tradicionales.

1.2. Objetivos

El presente Proyecto Fin de Carrera queda enmarcado dentro de un proyecto mucho más ambicioso que desde el grupo de trabajo *LibreSoft* de *GSYC* (*Grupo de Sistemas y Comunicaciones*) se está desarrollando bajo el nombre de *LibreGeoSocial*. *LibreGeoSocial* explota toda la funcionalidad de las redes sociales llevándolas hacia la categoría de móviles o dinámicas, gracias a la geolocalización de todos los nodos pertenecientes a la red. Concretamente, esta red social dinámica se apoya en el uso de modernos terminales de telefonía móvil con el sistema operativo *Android* lanzado al mercado por *Google* como principal valedor, así como el *hardware* más avanzado (conexiones de datos ya sea vía *UMTS*, *Wi-Fi* o *Bluetooth*, pantalla táctil, *GPS*, acelerómetros, sensor de campo magnético, cámara digital, etc.). Pero también se sustenta en los servicios *web* ofrecidos a través de un *framework* de desarrollo *Django*, y a lo que se debe añadir el desarrollo de un interesante módulo de realidad aumentada a modo de interfaz de la red social.

De este modo, este proyecto pasará a formar parte de las múltiples funcionalidades que *LibreGeoSocial* ya ofrece en la actualidad, teniendo como objetivo principal, y presente en todo momento:

- **Facilitar la organización de *gymkhanas* y mejorar la experiencia vivida por los participantes en ellas.**

  Disponer de un servicio de telecomunicaciones plenamente funcional, que ofrezca una mejora sustancial respecto a las *gymkhanas* realizadas

con los medios más tradicionales, tanto en el aspecto organizativo de *gymkhanas* de diversa índole, temática o finalidad, como en la experiencia vivida por los participantes en una de estas *gymkhanas*.

Pero para la consecución de este objetivo final y global, es necesario marcar una serie de subobjetivos mucho más locales, menos ambiciosos. Esto permitirá una descomposición del proyecto en etapas más pequeñas y fácilmente tratables, siendo los principales subobjetivos establecidos los indicados a continuación:

1. **Disponer de una base de datos para el almacenamiento estructurado de toda la información que se manejará en el sistema.**

   Disponer de una base de datos que permita el almacenamiento de todo tipo de información relativa a cada *gymkhana*, como pueden ser desde los propios enunciados de las pruebas, hasta las respuestas que los equipos participantes ofrecen a cada una de ellas.

2. **Disponer de una aplicación *web* para la creación de *gymkhanas* y posterior monitorización.**

   Dotar al sistema de una sencilla interfaz *web* mediante la cual los organizadores de una *gymkhana* puedan realizar dos operaciones fundamentales: la creación de la propia *gymkhana* con sus pruebas o retos, soluciones, etc. Y la monitorización o control del transcurso del evento (geolocalización de los equipos, respuestas recibidas, consultas y dudas de los equipos, etc.).

3. **Disponer de una aplicación *Android* a través de la cual participar en la *gymkhana*.**

   Desarrollar una aplicación para terminales de telefonía móvil con sistema operativo *Android*, a través de la cual puedan participar en la *gymkhana* obteniendo los enunciados de los retos a superar, enviando sus respuestas, teniendo comunicación permanente con la organización de la *gymkhana*, etc.

4. **Comprobar el correcto funcionamiento del sistema en un escenario real y atender todo tipo de comentarios de los usuarios finales: los participantes en las *gymkhanas*.**

   Desplegar el sistema en un escenario real con tal de realizar una serie de experimentos con personas ajenas al Proyecto Fin de Carrera. Con esto, además de comprobar el correcto funcionamiento del servicio fuera de un entorno controlado o un laboratorio, se podrá recavar información y opiniones sobre la percepción que los usuarios tienen del sistema, con tal de mejorar aquellos aspectos que puedan resultar más débiles y con mayor margen de mejora tras la fase de desarrollo.

Por tanto, además de estudiar y trabajar a fondo sobre las tecnologías aquí involucradas y que cuentan con un estado de plena ebullición tanto en la industria de las telecomunicaciones como en múltiples grupos de investigación alrededor de todo el mundo dada la revolución que suponen estos aspectos en el sector, y debido a toda la potencia que llevan aparejadas, el principal objetivo será el diseño y desarrollo de una plataforma *software* que a modo de aplicación cliente-servidor, permita en primer lugar a los responsables de la organización de la *gymkhana* a través de un servicio *web*, la creación de nuevos eventos, además de la configuración de cada una de las pruebas integrantes del concurso, inscripción de grupos participantes, etc.

Una vez puesta en marcha la *gymkhana* y entregando a cada equipo participante uno de estos terminales móviles con la correspondiente aplicación cliente para la participación en ella, los diferentes grupos podrán obtener en dicho terminal la información de cada nuevo reto a superar, así como posibles pistas, todo ello ofrecido siempre desde el servidor *web* sobre el que actuará la organización. Y del mismo modo, desde este servidor *web* y cómodamente, la organización podrá mantener un completo control sobre la evolución de la *gymkhana*, cosa que no siempre se está en condiciones de conseguir en las *gymkhanas* realizadas de las maneras más tradicionales. Para ello, a dicha organización se le permitirá monitorizar en todo momento la ubicación de cada equipo, comprobar la corrección o no de la respuesta dada a una determinada prueba, o incluso, el número de pruebas superadas hasta el momento por cada uno de los equipos.

Y por supuesto, se trata de una aplicación perfectamente utilizable en cualquier otro tipo de *gymkhanas*, como las tan populares y de moda *gymkhanas* fotográficas promovidas por los *Patronatos de Turismo* de múltiples ciudades a través de sus cascos históricos y monumentos. A su vez, esto permitirá conocer a fondo, las tecnologías aquí involucradas para continuar en un futuro trabajando en nuevos desarrollos innovadores, interesantes y útiles destinados a una sociedad cada vez más apegada a las nuevas tecnologías.

1.3. Estructura de la Memoria

A modo de panorámica sobre la estructura de la memoria, a continuación se presenta una breve indicación del contenido de cada uno de los capítulos en los que ésta se divide:

- ***Capítulo 1: Introducción.***

  En este capítulo se presenta la motivación para la realización del presente Proyecto Fin de Carrera, así como los objetivos marcados. Además,

se realiza un breve comentario sobre la estructura de la memoria y se comentan someramente otros proyectos con similar finalidad.

- ***Capítulo 2: Estado del Arte.***

  En este capítulo se presentan los aspectos más relevantes de las tecnologías con más presencia en el Proyecto Fin de Carrera, recomendándose la complementación de los aspectos técnicos con la lectura del glosario de términos presentado al final de la presente memoria.

- ***Capítulo 3: Diseño del Sistema.***

  En este capítulo se presenta el diseño del sistema con el que se podrá ofrecer el servicio establecido. Comenzando con un análisis general del sistema, se estudiarán los aspectos fundamentales de cada uno de los componentes del mismo.

- ***Capítulo 4: Implementación.***

  En este capítulo se presenta la implementación final llevada a cabo del servicio de telecomunicaciones que se marcó como objetivo, tanto del servidor con su base de datos, como de la interfaz *web* y la aplicación *Android*.

- ***Capítulo 5: Despliegue y Resultados.***

  En este capítulo se presentan las tareas ejecutadas una vez implementado el servicio, conducentes al estudio de las reacciones de usuarios con tal de poder probar el correcto funcionamiento del sistema en un escenario real y mejorando los aspectos que pudieran resultar más débiles.

- ***Capítulo 6: Conclusiones y Futuras Líneas de Trabajo.***

  En esta capítulo y a modo de cierre de la memoria, se presenta una serie de conclusiones sobre el proyecto de ingeniería realizado, así como las posibles líneas de trabajo a seguir con tal de avanzar, evolucionar, completar y mejorar la primera versión del sistema aquí presentado.

- ***Bibliografía.***

  Se presentan las principales referencias bibliográficas, definiciones de estándares, artículos e informes técnicos consultados durante la realización del Proyecto Fin de Carrera.

- ***Apéndices.***

  Se presentan un total de tres apéndices en que se recogen los resultados obtenidos en las encuestas realizadas a los participantes en los experimentos de la fase de pruebas del Proyecto Fin de Carrera, así como el presupuesto del proyecto y la planificación del mismo.

- ***Glosario.***

  Se presenta un breve glosario de términos involucrados en mayor o menor medida en la realización del presente Proyecto Fin de Carrera. Mediante las breves definiciones ofrecidas para cada término recogido, se pretende que el lector menos ducho en la materia pueda consultarlo a modo de guía rápida cuando lo necesite con tal de que consiga realizar una lectura inteligible de la memoria.

1.4. Otras Implementaciones de *Gymkhanas* Móviles

Durante la realización del presente Proyecto Fin de Carrera se tuvo noticia de una empresa, *Animatu*[1], creada en 2002, dedicada al desarrollo y distribución de videojuegos y contenidos para teléfonos móviles, así como servicios de *marketing* móvil.

Prestando especial atención a su desarrollo de juegos educativos, la mayoría de ellos son juegos *Java* integrados en campañas de *e-learning*. Entre los trabajos más recientes en este ámbito, se encuentra la realización de varios proyectos de *gymkhana* móvil[2]: en Bilbao, en el *Centro de Arte de Burgos*, en Lejona/Leioa y en Cáceres. En todos los casos presentando fines turísticos, y estando siempre apoyado por el *Ministerio de Cultura del Gobierno de España*.

Sin embargo, pese a las breves informaciones disponibles en línea, así como algunos vídeos demostrativos de las *gymkhanas* celebradas por esta empresa en colaboración con diferentes organismos, sólo se pueden obtener ciertas conjeturas sobre el funcionamiento del sistema propuesto. Entre estas ideas, se tiene en primer lugar que la participación en la *gymkhana* llevará aparejado cierto coste a los participantes que se corresponderá con la propia descarga de la aplicación para ser instalada en el teléfono móvil, no siendo éste el caso de la aplicación *Android* aquí desarrollada.

Por otro lado, actualmente muchos teléfonos disponibles en el mercado soportan la aplicación de *Animatu*, y perteneciendo a fabricantes muy variados: *Huawei*, *Motorola*, *Nokia*, *Samsung*, *Sharp* y *Sony Ericsson*. En el caso de *Android* no son tantos los teléfonos disponibles en el mercado hasta el momento, pero se debe tener presente que *Android* es una tecnología en pleno periodo de crecimiento, que viniendo a su vez avalada por una marca como *Google*, supone una garantía de calidad y de éxito, como bien demuestra el hecho de que cada vez más fabricantes de terminales de telefonía móvil comien-

[1] Sitio *web* de la empresa: *http://www.animatu.com*
[2] Sitio *web* del proyecto: *http://www.gymkanamovil.com*

zan a incluirlo en sus teléfonos o cuando menos, a plantearse un cambio de sistema operativo: *HTC*, *Motorola*, *Samsung*, *LG*, *Sony Ericsson*, *Huawei*, etc. De esta manera, el desarrollo de este sistema para *Android*, hace que se pueda explotar todo el *hardware* y el *software* que incluyen estos sofisticados *smartphones*, presentando posibilidades inimaginables para un teléfono de gama media y de cierta antigüedad, como por ejemplo el uso de *GPS* o de mapas digitales a consultar en la propia pantalla del móvil.

Además, según parece en el caso del sistema de *Animatu*, en cada *gymkhana* es necesario descargar una nueva aplicación que incluye las nuevas pruebas. Sin embargo, en el caso del sistema aquí propuesto, el usuario una vez haya descargado e instalado la aplicación en su teléfono, podrá elegir cualquier *gymkhana* disponible en cada momento, descargando los enunciados de los retos bajo demanda y nunca prefijados en la propia aplicación (y muy posiblemente siendo en el propio teléfono donde se determine la validez o no de una respuesta dada en el caso de *Animatu*), consiguiendo además un mayor grado de seguimiento y control de los equipos por medio de la interfaz *web* desarrollada para los organizadores del evento.

Capítulo 2

Estado del Arte

To understand a science it is necessary to know its history.

Auguste Comte

En este capítulo se presentan las nociones elementales sobre los aspectos tecnológicos con mayor peso dentro del Proyecto Fin de Carrera. Igualmente, ante cualquier desconocimiento por parte del lector en alguna tecnología o término que pueda aparecer a lo largo de la memoria, se recomienda la complementación de este capítulo con el glosario de términos que aparece al final de la misma.

2.1. Aplicaciones Cliente-Servidor

Dentro de los sistemas distribuidos, el modelo estándar más extendido para la ejecución de aplicaciones en una red en todos sus ámbitos territoriales (ya sea *WAN*, *MAN*, *LAN* o *PAN*) es el modelo cliente-servidor [Márquez-García:1996].

Partiremos del hecho de que un servidor será un proceso ejecutándose en una determinada máquina de la red. Su cometido se basará en gestionar diversos servicios así como el acceso a los recursos capaces de ofrecer dichos servicios. Para hacer posible esta gestión, el servidor estará en continua espera de una petición de servicio. En el momento en que esa petición se produzca, el servidor comenzará a ejecutar las acciones necesarias para atenderla, volviendo al estado de espera una vez se haya terminado de ofrecer el servicio solicitado.

Dependiendo de la forma que posea el servidor de prestar el servicio, se podrá estar ante dos tipos distintos de servidores:

- Los servidores interactivos, que serán aquellos que tras recoger una petición de servicio, la atienden.

- Los servidores concurrentes, aquellos otros que por cada petición de servicio recogida, crearán un proceso hijo encargado de atender la petición asignada cuya finalización se producirá una vez que el servicio se haya ofrecido. Este patrón a su vez tiene múltiples variantes, como puede ser el uso de un *pool* o repositorio de hilos disponibles en todo momento y destinados únicamente a la atención de las peticiones de servicio entrantes.

En cuanto a las ventajas e inconvenientes de cada uno de estos tipos de servidores, se puede decir que para el primero de ellos existe una menor complejidad en la implementación, aunque bien es cierto que si se cuenta con un servidor lento y una alta demanda de servicio, se producirán retardos de importante magnitud en las comunicaciones. Este tipo de servidores están indicados por tanto cuando la respuesta ante una petición de servicio es sencilla, por lo que el tiempo de procesado es mínimo.

Por su parte, los servidores concurrentes, presentan los inconvenientes de una mayor complejidad, además de que tan sólo se podrán implementar sobre sistemas multiproceso, como es el caso de *Linux* y de *UNIX* en general. Pero la ventaja que ofrecen, y que por otra parte hace que sean los servidores más extendidos por la red, es una alta velocidad de recogida de peticiones en el servidor por parte de su proceso padre, al ser éstas atendidas por cada uno de los procesos hijo creados. Por lo tanto, los servidores concurrentes son especialmente recomendables en aquellas aplicaciones para las cuales los tiempos de servicio sean variables.

En la figura 2.1 se encuentra un resumen gráfico de la interacción entre los dos entes de una aplicación cliente-servidor, teniendo en cuenta que se ha considerado un servidor concurrente, a partir del cual fácilmente se puede entender cuáles serían las acciones de uno interactivo.

Así pues, se pueden resumir de manera genérica las acciones que llevará a cabo un programa servidor y las que realizará un programa cliente. Analizando en primer lugar el servidor:

1. Abrir el canal de comunicaciones y notificar a la red tanto su dirección (identificador bajo el cuál responderá) como su disposición para la atención de peticiones.

2. Permanecer a la escucha en su *socket* de comunicaciones, esperando la recepción de una petición de servicio por parte de un cliente.

3. Una vez recibida la petición, en caso de tratarse de un servidor interactivo, se atenderá dicha petición. En caso de disponer de un servidor concurrente, éste creará un proceso hijo que finalizará su ejecución cuando haya terminado de atender la petición del cliente.

4. Nuevamente se volverá al punto 2, es decir, se volverá a la espera de una nueva petición repitiéndose este ciclo de manera indefinida.

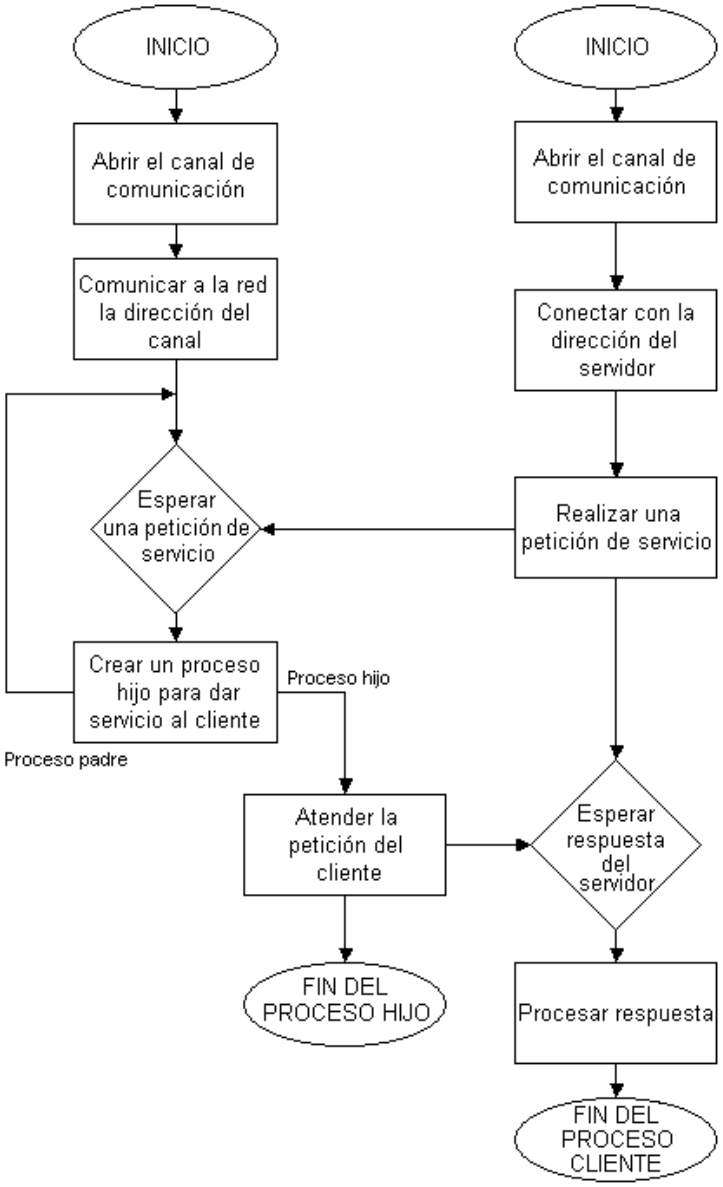


Figura 2.1: Diagrama de flujo de una aplicación cliente-servidor

Por su parte, el proceso cliente llevará a cabo la ejecución de las siguientes acciones:

1. Abrir el canal de comunicaciones y conectar con la dirección del servidor (esta dirección deberá ser conocida por parte del cliente y acorde con las dirección del conector que se esté empleando en la comunicación, ya sea una dirección *MAC Ethernet*, *IP*, puerto de nivel de transporte, etc.).

2. Enviar al servidor una petición de servicio y permanecer a la espera de obtener una respuesta.

3. Procesar la respuesta recibida, cerrar el canal de comunicaciones y terminar la ejecución.

2.2. *Python* y *Django*

Sintetizando la evolución que han seguido las aplicaciones *web* a lo largo de su historia [Andersson:2006], se puede afirmar que paulatinamente y desde su origen, en que las páginas *web* servidas contaban con un contenido íntegramente estático, se pasaría después a la inclusión de contenido dinámico en base a las acciones realizadas por el usuario, cobrando entonces fuerza el desarrollo de sitios basados en el lenguaje interpretado de programación *PHP*, o incluso los *scripts* de *shell* para los llamados *CGI-bin*. Llegando al momento actual, en que la conocida como *web 2.0* [Vlist:2007] desempeña un papel fundamental, donde la tendencia es que las aplicaciones *web* cada vez se parezcan más a las aplicaciones de escritorio y donde no basta sólo con el contenido dinámico, sino que también se busca la máxima interacción posible con el usuario, por ejemplo por medio de la tecnología *AJAX*. Y es en esta *web 2.0* donde además de la plataforma *Java Enterprise Edition*, fundamentalmente dos *frameworks* para el desarrollo, despliegue y mantenimiento, se han impuesto como los de referencia: *Rails*, empleando el lenguaje de programación interpretado *Ruby*, y *Django* [DjangoWebSite], empleando también otro lenguaje interpretado, concretamente, *Python* [PythonWebSite].

2.2.1. *Python*

Creado en Holanda por Guido van Rossum a finales de los años 80 y distribuido bajo una licencia de código abierto (*Python Software Foundation License*) compatible con la licencia *GPL*, *Python* [Lutz:2006] es un lenguaje de programación interpretado, ofreciendo por esto mismo las ventajas de los lenguajes propios de *scripting*, como puede ser un rápido desarrollo (si bien

es cierta su recomendación en aplicaciones relativamente pequeñas, dada la mayor dificultad y complejidad que se puede dar en la depuración de las mismas), además de buscar una mayor facilidad tanto de lectura del propio código, como en el diseño.

Todo esto unido a las múltiples librerías o *APIs* (*Application Programming Interface*) con que cuenta el lenguaje, su tratamiento de excepciones, su uso de clases y módulos, su gestión automática de memoria (la reserva de memoria cuando es necesaria, y su liberación cuando deja de emplearse, son operaciones transparentes al programador) o su uso de estructuras de datos sencillas y flexibles como diccionarios, listas o *strings* como parte integral del lenguaje, hacen que *Python* se suela considerar como un lenguaje interpretado moderno, especialmente al compararlo con otros anteriores como puedan ser *TCL*, *Perl* o *PHP*.

Python también es un lenguaje orientado a objetos (aunque realmente se podría considerar un lenguaje de programación multiparadigma, pues permite a los desarrolladores adoptar desde un estilo de programación funcional o uno estructurado, hasta la propia programación orientada a objetos), en el que todo son objetos, incluidas las propias clases. Además, es dinámica y fuertemente tipado, así como sensible a letras mayúsculas y minúsculas (*case sensitive*), y en el que la declaración de variables se realiza implícitamente, es decir, sin necesidad de declararlas de manera explícita con la consecuente comodidad para el programador, así como los peligros que esto puede conllevar.

2.2.2. *Django*

Dentro del desarrollo *web*, se tienen varias opciones a la hora de trabajar en este tipo de aplicaciones y/o servicios. Si se persigue la máxima libertad en el desarrollo, la mejor opción será trabajar con el lenguaje de *scripting PHP* o incluso con *Python* y sus librerías para el manejo del protocolo *HTTP* [RFC2616]. Sin embargo, esta opción puede hacer que el tiempo de desarrollo aumente considerablemente al tener que partir prácticamente desde cero. El caso opuesto sería el presentado por otras plataformas con funcionalidades y herramientas de serie, que facilitan sobremanera el trabajo así como se produce un gran ahorro en el tiempo de desarrollo, como puede ser el caso de *Zope*, o algunas otras plataformas de propósito específico como *Moodle*. Pero la desventaja de estas soluciones será la escasa libertad y flexibilidad con que contará el programador. Así, buscando un término medio o una relación de compromiso entre esa rapidez de desarrollo y la flexibilidad y libertad, se encuentran los entornos de desarrollo *web* como *Ruby on Rails*, *JavaEE* (*Java Enterprise Edition*) o el propio *Django* [Holovaty:2006, Holovaty:2007, DjangoCommunity], que escrito en *Python* y

siendo multiplataforma, originalmente fue pensado para "*The World Company*" (diario publicado en Lawrence, Kansas) y orientado a la gestión de sitios de noticias, quedando liberado públicamente en julio de 2005 bajo una licencia *BSD*.

De esta manera, se puede definir *Django* como un *framework* o *Entorno Integrado de Desarrollo* de aplicaciones *web* entre cuyas principales funcionalidades se encuentran:

- Herramientas para la gestión de la aplicación.
- *Framework* de capa de presentación.
- Manipulación de bases de datos mediante el mapeo de objetos de la aplicación a entradas relacionales.
- Seguridad (*XSS*, *SQL Injection*, etc.).
- Un sistema de serialización para generar y procesar instancias del modelo de la aplicación *Django* representadas en formatos como *JSON* o *XML*, *caché*, internacionalización, plantillas, etc.

Además, se debe tener en cuenta que el objetivo de *Django* es ofrecer un desarrollo muy rápido y sencillo, configurando un entorno integrado y completo (incluyendo un servidor *web* ligero, válido durante la fase de desarrollo pero siendo recomendado *Apache* durante la fase de producción junto a *mod_python*, que se trata de un módulo de *Apache* que incrusta el intérprete de *Python* en el servidor *web* (se debe recordar también que el desarrollo en *Django* se realizará siempre en lenguaje *Python* y contando con la versión 2.3 o superior de dicho lenguaje); lo mismo ocurre con la base de datos *SQLite3* que también incluye y muy cómoda para dicho desarrollo, siendo recomendada *PostgreSQL* [PostgreSQLWebSite] en la fase de producción, soportando también *MySQL*), buscando la máxima reutilización de código, evitando la replicación siempre que sea posible (*DRY*, *Don't Repeat Yourself*). El hecho de emplear un lenguaje interpretado como es *Python*, también permite la realización de cambios en caliente sin necesidad de rearrancar el servidor *web*, ahorrando con ello mucho tiempo, al igual que se aumenta la rapidez mediante las útiles y completas descripciones de los errores que se producen.

También se debe notar que habitualmente el desarrollo de toda la aplicación comienza por el propio diseño del modelo, es decir, se comienzan modelando los datos que se van a manejar en la aplicación, para acabar empleando el patrón de diseño *MVC* (*Modelo-Vista-Controlador*) [Selfa:2006].

Por último y entrando más en detalle, los elementos que configuran el núcleo de este *framework* son en primer lugar un mapeador de objetos del modelo (clases de *Python*) y la base de datos relacional [Miguel-Castaño:1999] en

la que realmente se almacenan dichos datos con los que opera la aplicación. Y al mismo tiempo, las partes con las que el programador realmente trabajará, que serán un manejador de *URLs* [RFC1738] en base a expresiones regulares, un sistema de vistas para el procesamiento de las peticiones de usuario y un sistema de plantillas que serán empleadas para generar dinámicamente el código *HTML* [W3CHTML] (o cualquier otro formato de información soportado) a ofrecer como respuesta al usuario. De hecho, esto hace a algunos autores afirmar que *Django* no emplea el patrón *Modelo-Vista-Controlador* sino un patrón *Modelo-Plantilla-Vista*, donde el *Modelo* de datos sigue siendo el *Modelo* en *Django*, pero serán las *Plantillas* de *Django* las que desarrollen la función de la *Vista*, así como el *Controlador* pasará a llamarse *Vista* en *Django*.

2.3. *Android*

Android [AndroidDeveloper, AndroidCommunity] es una plataforma *software* que incluye un sistema operativo destinado a dispositivos móviles y cuyo *kernel* está basado en los sistemas *Linux*. Inicialmente fue desarrollado por *Google* tras adquirir en julio de 2005 una pequeña empresa de California llamada *Android, Inc.* dedicada al desarrollo de *software* para teléfonos móviles. Desde este momento, comenzaron a surgir rumores sobre la pretensión de *Google* de querer entrar en el mercado de la telefonía móvil, y más concretamente con el lanzamiento de un teléfono móvil (*GPhone*, *Google Phone*). Sin embargo, el 5 de noviembre de 2007, Andy Rubin (Director de *Google* en Plataformas Móviles) diría que: "*[The] Android [Platform] - is more significant and ambitious than a single phone*". En esa misma fecha, se fundó la *Open Handset Alliance* (*OHA*), un consorcio de importantes compañías internacionales entre las que se encuentran fabricantes de dispositivos *hardware*, empresas de *software* y operadores de telecomunicaciones, como la propia *Google*, *HTC*, *Intel*, *Motorola*, *Qualcomm*, *T-Mobile*, *Sprint Nextel* y *NVIDIA*. Saliendo al mismo tiempo su primer producto, *Android*, y teniendo como objetivo el desarrollo de estándares abiertos para dispositivos móviles, a este consorcio se sumarían en diciembre de 2008 empresas como *Sony Ericsson*, *Vodafone*, *ARM Holdings PIc*, *Toshiba Corp* o *Garmin Ltd*, hasta configurar el total de 48 empresas integrantes de la *OHA* en la actualidad.

2.3.1. Principales Características

Desarrollado en la actualidad por la *Open Handset Alliance*, *Android* se distribuye bajo una licencia *Apache 2.0* y *GPLv2*, según lo cual se trata de un *software* libre y de código abierto. Esto hace que tanto empresas como

particulares puedan publicar las aplicaciones propietarias que hayan desarrollado para *Android*, sin necesidad de que sean aprobadas por *Google*, y pudiendo ser descargadas e instaladas gratuitamente o con un determinado coste, contando también con un sistema de votación similar al del portal *Youtube*, tal y como se puede observar en la figura 2.2, donde se muestra una captura de esta aplicación "*Android Market*" accesible en todos los teléfonos móviles *Android*.

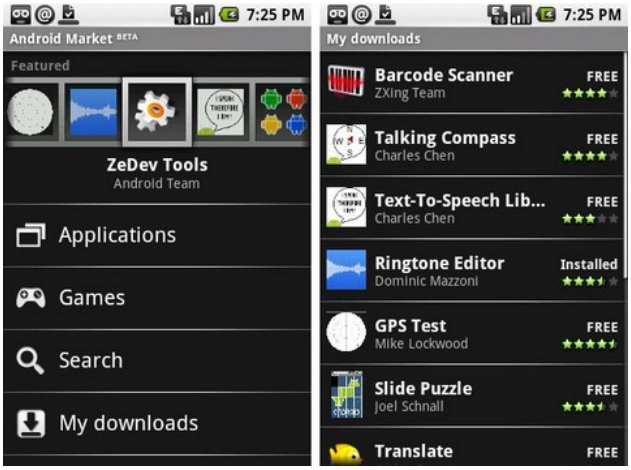


Figura 2.2: Captura de *Android Market* para la publicación y descarga de aplicaciones

Algunas otras de las características más relevantes de *Android* [Meier:2008, Burnette:2008, Gramlich:2008] se listan a continuación:

- Algunos aspectos relacionados con la propia arquitectura de la plataforma y que se estudiarán en detalle más adelante, como son la reutilización y reemplazo de componentes gracias al *framework* de aplicaciones, o la máquina virtual *Dalvik*.
- Incluye de manera integrada un navegador *web* (basado en el motor *WebKit*).
- Optimización de gráficos, incluyendo tanto una librería de gráficos *2D* como una en *3D*.
- Base de datos *SQLite* para el almacenamiento de datos estructurados.
- Telefonía móvil, *Bluetooth*, *EDGE*, *3G* y *Wi-Fi*, todo ello dependiente del *hardware*.
- Cámara, *GPS* (*Global Positioning System*), brújula y acelerómetro (también dependiente del *hardware*).

- Soporte para pantalla táctil.
- Soporte para formatos de vídeo, audio e imágenes (*MPEG4*, *H.264*, *MP3*, *AAC*, *AMR*, *JPG*, *PNG*, *GIF*, etc.)
- Facilita el desarrollo para la plataforma gracias al *plug-in* disponible para el *IDE Eclipse* [EclipseWebSite], así como las herramientas de depuración que incluye, código de sencillas aplicaciones a modo de ejemplo, tutoriales, o un emulador de dispositivo para la fase de desarrollo.

Figura 2.3: Emulador de dispositivo para *Android*

2.3.2. Arquitectura

Como se indicó al inicio de esta sección, *Android* es una plataforma *software* para dispositivos móviles que incluye un sistema operativo, un *middleware* [Tanenbaum:1995] y una serie de aplicaciones clave. La *SDK* (*Software Development Kit*) de *Android* incluye las herramientas y *APIs* necesarias para el desarrollo de aplicaciones sobre la plataforma *Android*, empleando para ello el lenguaje de programación *Java* [Naughton:1996]. Analizando las distintas capas de la arquitectura de *Android*, mostrada de manera intuitiva en la figura 2.4, se pueden apreciar tres niveles bien diferenciados:

- **Núcleo.** Como todo núcleo de un sistema operativo, actúa como interfaz para abstraer el *hardware* de cara al resto de pila de *software* instalada en el dispositivo o equipo. En el caso de *Android*, el núcleo depende de un *Linux 2.6*, realizando las funciones propias de seguridad, gestión de memoria, procesos, red, *drivers*, batería, gestión de periféricos (pantalla, teclado), etc.

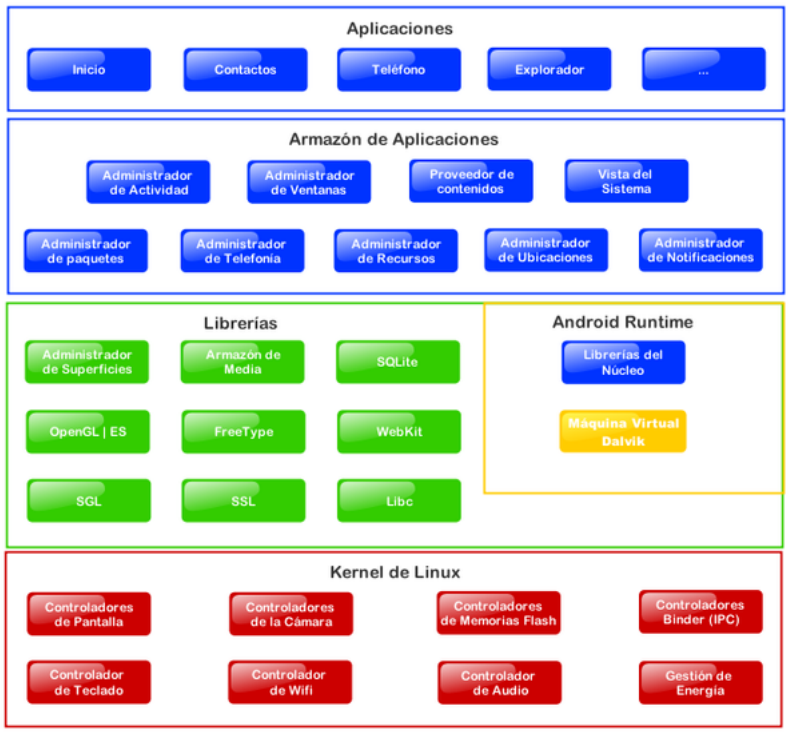


Figura 2.4: Arquitectura en capas de la plataforma *Android*

- ***Middleware.*** Se trata de una máquina virtual similar a las propias máquinas virtuales de *Java* (*JVM*, *Java Virtual Machine*) ofrecidas por *Sun Microsystems*, y que recibe el nombre de *Dalvik*. Incluye las librerías o *APIs* fundacionales de *Java* y por tanto, las funcionalidades básicas del lenguaje (realmente, se conserva la sintaxis del lenguaje *Java* [Naughton:1996], pero no se cumple con sus estándares, ni en su versión *J2SE* (*Java 2 Standard Edition*) ni en su versión *J2ME* (*Java 2 MicroEdition*) para dispositivos limitados en recursos). En *Android*, cada aplicación se ejecutará como un proceso con su propia instancia de la máquina *Dalvik*, estando dicha máquina virtual especialmente diseñada para un funcionamiento eficiente durante la ejecución de varias máquinas virtuales *Dalvik*. Igualmente, se incluye un conjunto de librerías del lenguaje *C/C++* usadas por determinados componentes de *Android*, como son la librería estándar de *C*, librerías de gráficos en *3D*, base de datos *SQLite*, etc., configurando así una serie de capacidades que serán ofrecidas al desarrollador a través del *framework* de aplicaciones.

  Profundizando en la *DalvikVM*, se trata de un intérprete que ejecuta ficheros en formato *Dalvik Executable* (*.dex*), que es un formato optimizado para realizar el almacenamiento y mapeo de memoria de manera eficiente. Es también una máquina virtual basada en registros y que

puede ejecutar clases compiladas por un compilador de *Java* que las haya transformado en formato nativo usando para ello la herramienta "*dx*" incluida en la *SDK* de *Android*. Así, la diferencia fundamental con una *JavaVM* empleada para las aplicaciones de escritorio habituales en un *PC*, es que se trata de una máquina virtual basada en *stack*, frente a *DalvikVM* basada en registro como se ha señalado, debido precisamente a que los procesadores de los dispositivos móviles están optimizados para una ejecución basada en registro. De hecho, las máquinas virtuales basadas en registro permiten una ejecución más rápida.

- **Aplicaciones.** Se incluye un *framework* de aplicaciones al que el desarrollador tendrá acceso por medio de un *API*, estando la arquitectura diseñada de tal manera que se consiga simplificar la reutilización de componentes, como puede ser el administrador de ventanas, el administrador de telefonía o el administrador de paquetes para que el usuario pueda reemplazarlos fácilmente, al mismo tiempo que cualquier aplicación puede publicar sus capacidades para que cualquier otra pueda hacer uso de ellas.

  Precisamente de este *framework* de aplicaciones hacen uso las aplicaciones clave o base que se suelen incluir con *Android*, como son un cliente de *e-mail*, programas de mensajería *SMS*, navegador *web*, mapas, telefonía, contactos, etc., estando todas estas aplicaciones escritas en *Java*, tanto las de serie como las desarrolladas por cualquier otro desarrollador para *Android*.

2.3.3. *Android SDK*

El hecho de que la programación en la plataforma *Android* se realice por medio del lenguaje *Java*, implica muchas ventajas, como es el hecho de que en la actualidad, existe una amplia comunidad de desarrolladores en dicho lenguaje, por lo que tan sólo será necesario que dichos desarrolladores comprendan una serie de nuevos conceptos propios de *Android*. Pero además, la programación para esta plataforma se hace mucho más sencilla teniendo en cuenta la existencia de la llamada *"Android SDK"*, es decir, el entorno de programación de aplicaciones para *Android*. Esta *SDK* consta de una serie de librerías para el desarrollo, tutoriales y un emulador de un dispositivo móvil con *Android*. Por supuesto, también se cuenta con un *plug-in* para un *IDE* (*Integrated Development Environment*) como pueda ser *Eclipse* [EclipseWebSite], con el cual se facilitará la tarea de compilación, ejecución y depuración. Y como ya se ha señalado con anterioridad, se emplea una máquina virtual llamada *Dalvik*, similar a las de *Sun Microsystems* para el lenguaje *Java* pero optimizada para una mejor gestión de memoria, de procesos, etc.

2.3.4. *Componentes de una Aplicación* Android

En *Android* existen cuatro posibles bloques de construcción para una aplicación [Gramlich:2008]:

- ***Activity.***

  Se trata del componente más común de una aplicación *Android*. Normalmente, será cada una de las pantallas que se muestren en la aplicación desarrollada. Cada actividad se implementa como una clase que hereda (*extends*) de la clase madre *Activity* y que ya está incluida en las *APIs* de *Android*. Esta clase, mostrará al usuario de la aplicación una interfaz (*GUI*) compuesta de *Views* y responderá a los eventos que dentro de ella se produzcan (pulsar un botón, pulsar un cuadro de texto para escribir en él, hacer *scroll*, etc.), por lo que se deberá lanzar una nueva *Activity* cada vez que se quiera cambiar de interfaz, consiguiéndose un código modular. Estas actividades a su vez, cuentan con un ciclo de vida desde que son creadas hasta que se destruyen, tal y como se muestra en la figura 2.5, ejecutándose en cada transición entre estados un determinado método, siendo los más destacables dada su importancia los que se describen a continuación:

  - *onCreate()*: ejecutado cuando la actividad ha sido lanzada (y por tanto creada) por primera vez.
  - *onPause()*: estado en el que la actividad se encuentra cuando se lanza otra nueva actividad (acción que se realizará, normalmente, desde la actividad que queda ahora parada), permaneciendo en segundo plano hasta que se regrese a su ejecución.
  - *onResume()*: tras un estado de pausa, se ejecuta este método de manera que vuelva a encontrarse en primer plano de ejecución.
  - *onDestroy()*: este método se ejecutará cuando se haya indicado explícitamente la finalización de la actividad. En caso de que esta finalización no sea señalada de manera explícita, lo habitual será que la *Activity* permanezca en segundo plano (tras ser pausada, pasará a estado de parada), a no ser que el propio sistema operativo necesite finalizarla para reasignar los recursos (fundamentalmente memoria) que ésta tuviera asignados.

  Como última mención al respecto, destacar que el diseño de las interfaces gráficas de usuario, pudiéndose realizar programáticamente, en *Android* también se permite la cómoda posibilidad de definirlas por medio de ficheros *XML*, donde se definen los componentes de la interfaz (botones, textos, cuadros de texto, imágenes, iconos, etc.), así como las propiedades de cada uno (tamaños, márgenes, colores, etc.).

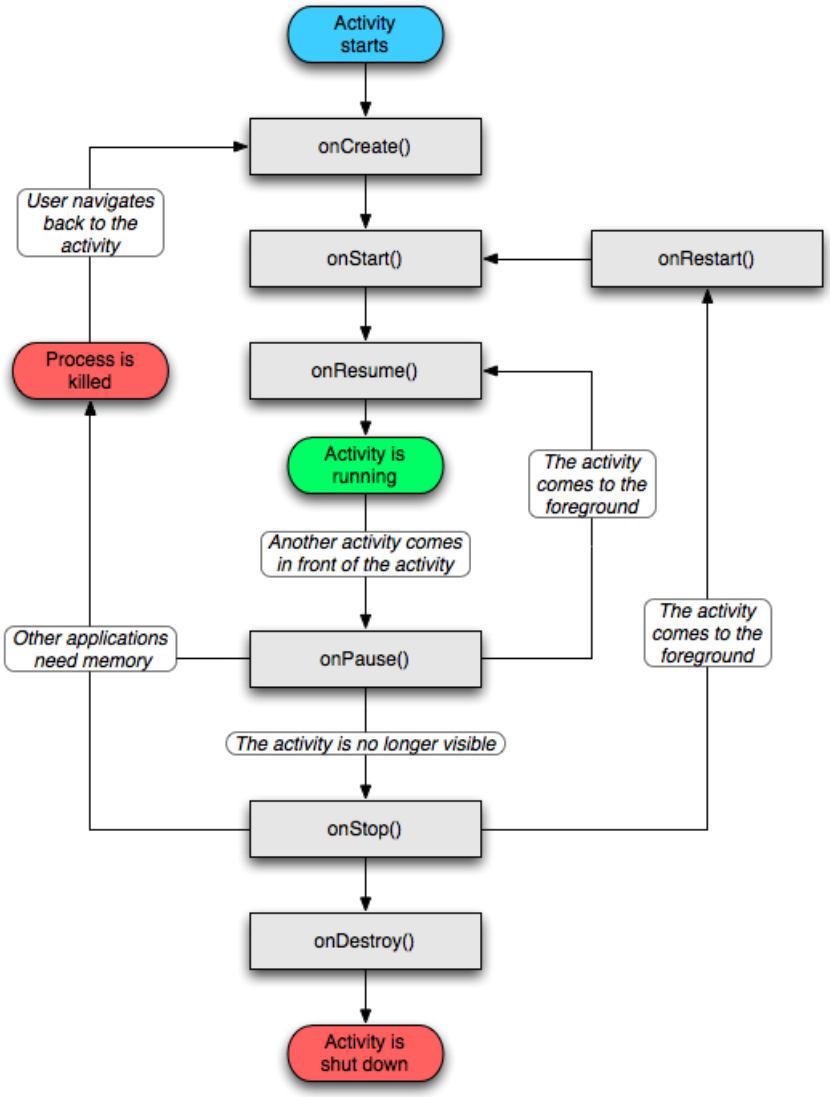


Figura 2.5: Ciclo de vida de una *Activity* de *Android*

- ***Service.***

  Un servicio será aquél código que se ejecuta en segundo plano y sin necesidad de contar con una interfaz de usuario. Un claro ejemplo de *Service* podría ser un programa que periódicamente comprueba la posición del terminal móvil por medio del *GPS* del mismo, para lanzar una notificación cuando el móvil haya llegado a un determinado lugar, se encuentra cerca de un determinado amigo, etc.

- ***Intent.***

  Este componente es utilizado fundamentalmente para pasar en la aplicación *Android* de una pantalla a otra (de una *Activity* a otra *Activity*).

- ***Broadcast Receiver.***

  Se trata de un componente que no realiza ninguna operación excepto cuando recibe un mensaje o anuncio en *broadcast*, activándose en ese momento para ejecutar el código que el desarrollador haya considerado oportuno ante el evento que se haya producido y de interés para ese *Broadcast Receiver*.

- ***Content Provider.***

  Fundamentalmente, las aplicaciones *Android* pueden almacenar sus datos de manera permanente por medio de ficheros o en una base de datos *SQLite*. Pero cuando realmente resulta interesante el uso de un *Content Provider*, es cuando se quiere o se necesita que los datos de la aplicación sean compartidos con otras aplicaciones. Así, un *Content Provider* es una clase que implementa un conjunto estándar de métodos para permitir a otras aplicaciones que almacenen y recuperen los datos que están siendo manejados por dicho *Content Provider*.

Por último, anotar que no toda aplicación debe contar con estas cuatro posibilidades de trabajo, pero sí será normal emplear algún tipo de combinación de ellas, siendo siempre listadas en el fichero *AndroidManifest.xml* (fichero de vital importancia en todo proyecto *Android*, en el que se declararán los componentes que integran la aplicación además de los permisos con los que ésta cuenta: uso de *Internet*, cámara, vibración, etc.).

2.3.5. Terminales *Android* en el Mercado

En la actualidad cada vez son más los nuevos terminales móviles con *Android* que aparecen en el mercado. Pero, destacando los hitos fundamentales dentro del corto historial de *Android*, en octubre de 2008 se lanzó el primero de estos teléfonos, concretamente en Estados Unidos y en Inglaterra. Se trata del *HTC Dream* [HTCDream], también conocido como *T-Mobile G1* y que cuenta con las siguientes características:

- Procesador *Qualcomm MSM7201A* a 528*MHz*.
- Memoria *RAM* de 192*MB* y memoria *ROM* de 256*MB*.
- Pantalla táctil de 3,17 pulgadas.
- Cámara de 3,2*Mpx* con *auto focus*.
- *GPS*, brújula digital y sensor de movimiento.
- Teclado *QWERTY*.
- Peso de 158 gramos.
- Lanzado en España por *Movistar* durante el segundo trimestre de 2009.

Figura 2.6: Teléfonos móviles *HTC Dream* (*G1*) y *HTC Magic* (*G2*)

El lanzamiento del *HTC Magic* [HTCMagic] (*T-Mobile G2*) también supuso un gran éxito, una versión evolucionada del *HTC Dream* sin teclado *QWERTY*, lanzado en España por *Vodafone*. Este terminal es muy similar en cuanto a *hardware* a su antecesor, sufriendo algunos cambios de diseño. Incluye $288MB$ de *RAM*, $512MB$ de *ROM* y *trackball* con botón *Enter* similar al de *Blackberry*. Incluía la liberación de *Android* denominada *Cupcake* en que se añadieron ciertas funcionalidades como el soporte para *Bluetooth stereo*, grabación y reproducción de vídeo o un nuevo navegador *web* con nuevas funcionalidades, además de un motor de *JavaScript* [ECMA262].

La aparición en el mercado de estos dos terminales de telefonía móvil, a parte de estar respaldada por un gran éxito entre los consumidores, vino a avalar el gran potencial aún por explorar que supone la plataforma *Android* en el desarrollo de nuevos servicios para terminales móviles, explotando el *hardware* de esos móviles de última generación, normalmente denominados como *smartphones*. Una de las mayores apuestas es precisamente la inclusión y uso de un sensor de campo magnético y que se puede utilizar para calcular la orientación del terminal a modo de brújula, lo que supone un gran éxito ya a día de hoy debido a las aplicaciones desarrollados aprovechando esto, destacando especialmente aquellas que tienen que ver con el concepto de realidad aumentada [Román-López:2009].

Pero sin duda alguna, la nueva vuelta de tuerca en el ámbito de estos terminales móviles la ha supuesto el lanzamiento del *Nexus One Phone*, un nuevo *smartphone* respaldado por *Google* e incluyendo la plataforma *Android*. Posee un procesador *Qualcomm* a $1GHz$, memoria *flash* de $512MB$ y una *microSD* de $4GB$ ampliable a $32GB$, con memoria *DRAM* de $512MB$. La pantalla táctil es de 3.7 pulgadas, cámara de $5.0Mpx$ con *auto focus*, peso de 130 gramos con batería incorporada, y conectividades inalámbricas *Wi-Fi*, *Bluetooth*, *GPRS*/*EDGE* y *UMTS* tribanda ($900MHz$, $1700MHz$ y $2100MHz$), además de puerto *USB* y *GPS*.

Figura 2.7: Terminales móviles *Nexus One Phone* y *iPhone*

En cuanto a la rivalidad ya presente en el mercado entre *Android*, apostando por el código libre y por el desarrollo de contenidos y servicios, frente al *iPhone* de *Apple*, apuesta directa por el diseño y el *hardware*, así como por un modelo de negocio basado en el pago por el desarrollo y publicación de aplicaciones, resulta de especial relevancia el informe realizado a comienzos de 2009 por la consultora *Informa Telecoms & Media*, según la cual en 2012 las ventas de teléfonos móviles basados en la plataforma *Android* superarán las ventas del *iPhone*, datos avalados también por *Gartner* o *AdMob* entre otros.

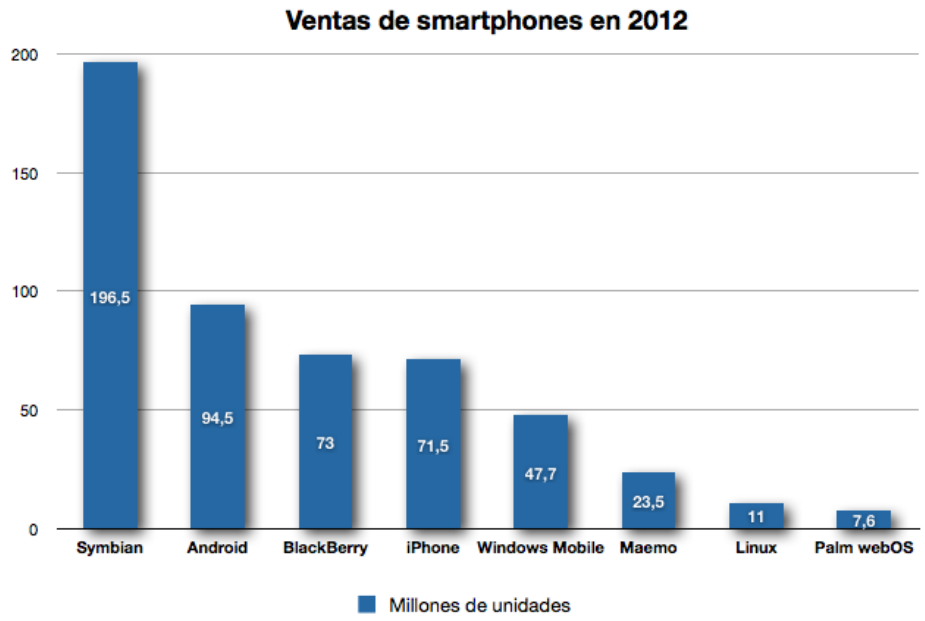


Figura 2.8: Estimación de ventas de *smartphones* para el año 2012 según *Gartner*

De hecho parece lógica esta estimación teniendo en cuenta el gran número de grandes fabricantes a nivel internacional que ya han lanzado al mercado sus primeros terminales con *Android*, o bien, ya han anunciado sus intenciones futuras de hacerlo, siendo el caso de algunas de las más destacadas: *Motorola*, *Sony Ericsson*, *Samsung*, *HTC* o *Acer*, además de *Lenovo* preparando un teléfono basado en *Android* y que soporte el estándar chino *3G TD-SCDMA*, junto al propio fabricante chino *Huawei* con la importante magnitud que supone dicho mercado. Por tanto, y pese al gran éxito de ventas que está cosechando el *iPhone*, la capacidad de fabricación y venta de estos dispositivos por parte de *Apple* es más que cuestionable si se tiene en cuenta que muy pronto deberá rivalizar con la capacidad de estas otras muchas empresas preparando terminales *Android*, situación en la que sí parece incuestionable la caída que sufrirá *Symbian* pese a las medidas que ya se están tomando desde *Nokia*.

2.4. *LibreGeoSocial*

LibreGeoSocial[1] es un proyecto desarrollado dentro del grupo *LibreSoft* del *Departamento GSYC* (*Grupo de Sistemas y Comunicaciones*, *Escuela Técnica Superior de Ingeniería de Telecomunicación* de la *Universidad Rey Juan Carlos*) consistente en una red social móvil y contando con un interfaz de realidad aumentada. En esta red social, todos los nodos se encuentran geolocalizados y podrán ser mostrados al usuario a través de una interfaz tradicional en forma de listas, o bien a través de la interfaz de usuario de realidad aumentada [Román-López:2009], que no sólo permitirá al usuario la visualización de *tags* a través de la cámara del teléfono móvil, sino también la creación de *tags* para determinados objetos del mundo real, etiquetarlos de manera virtual de modo que cualquier otro usuario que pase por sus inmediaciones pueda leerlo, etc.

En cuanto a la estructura interna del sistema que hace posible *LibreGeoSocial*, existen fundamentalmente dos componentes:

- ***LibreGeoSocial.*** Se trata de un *framework* desarrollado en *Python* y *Django* que facilita la creación de redes sociales móviles, incorporando la geolocalización de todos sus nodos.

- ***LibreGeoSocialApp.*** Se trata de una aplicación *Android* que explota la funcionalidad de la red social geolocalizada *LibreGeoSocial*. Utiliza dispositivos físicos como el *GPS* para obtener la posición de un determinado nodo en todo momento.

[1] Sitio *web* del proyecto: *http://libregeosocial.morfeo-project.org/*

Cabe destacar también un módulo adicional del proyecto que se denomina *LibreGeoSocial-OMF*. Este módulo es una adaptación de *LibreGeoSocialApp* en la que utilizando varios servicios de *Movistar* a través de las *APIs* de *OpenMovilForum*, se consigue ofrecer una alternativa para obtener datos interesantes en la aplicación (posicionamiento, envío de *SMS* y consultar agenda). Con esta adaptación se pretende conseguir la independencia de una red *GSM / UMTS* y de un dispositivo *GPS* para poder realizar acciones determinadas dentro de la red social móvil.

Por último, señalar que al ser *LibreGeoSocial* un proyecto de *software libre* y de código abierto, la disponibilidad del código fuente y de los módulos del proyecto permitirá personalizar *LibreGeoSocial* dependiendo del escenario al que se quiera aplicar: turismo, juegos de realidad aumentada, información diversa sobre el campus de una universidad (edificios, compañeros de clase, tiendas cercanas), etc.

Capítulo 3

Diseño del Sistema

La función de un buen software es hacer que lo complejo aparente ser simple.

Grady Booch

En este capítulo se presentan los aspectos más relevantes del diseño realizado en el presente Proyecto Fin de Carrera. En primer lugar, se comienza por un análisis genérico del sistema que permitirá configurar el servicio de telecomunicaciones que se persigue. Posteriormente, se analizarán los aspectos más relevantes del diseño realizado en el lado servidor, finalizando con el diseño del lado cliente.

3.1. Introducción

Para abordar el problema presentado se procederá a la elaboración de varios componentes que operando entre sí configurarán el sistema deseado. Puesto que el sistema a diseñar consta de varios módulos o componentes bien diferenciados, se realizará un análisis desglosado del diseño en cada caso, siendo el esquema general el presentado en la figura 3.1.

Como se puede observar en la figura, el sistema estará compuesto por tres módulos fundamentales:

- Una máquina que operando a modo de servidor, pondrá a disposición de cualquier cliente un servicio *web* a través del cual interoperar con la base de datos que será manipulada convenientemente por este servidor.

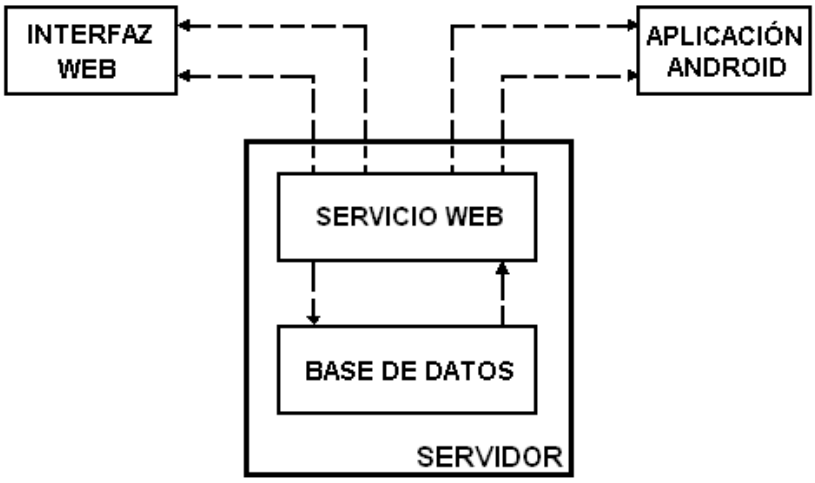


Figura 3.1: Esquema general del diseño del sistema

- Una aplicación *web* cuyo uso estará destinado a los organizadores de la *gymkhana* (para crear eventos con sus pruebas, monitorizar las acciones de los equipos durante el transcurso del evento, etc.) y que servirá a modo de interfaz entre dichos organizadores y el servidor que mantendrá toda la información en la base de datos.
- Una aplicación *Android* que se podrá instalar en los teléfonos móviles con dicha plataforma *software* y que será la interfaz a través de la cual los participantes de la *gymkhana* obtengan los enunciados de los retos a superar, envíen sus respuestas al servidor, etc.

Así pues, en los siguientes apartados se pasará a analizar el diseño realizado en cada uno de los módulos señalados.

3.2. Diseño del Servidor

En el caso del servidor, será necesario el diseño de la aplicación que ejercerá las funciones propias de dicho proceso, además del diseño del modelo de datos y que será el verdadero pilar para el correcto funcionamiento del sistema, motivo por el cual se analizará en primer lugar dada su vital importancia.

3.2.1. Modelo de Datos

Antes de abordar el diseño realizado de la base de datos que se va a emplear, resulta necesario estudiar someramente la base de datos relacional [Miguel-Castaño:1999] de la que ya se dispone en el proyecto *LibreGeoSocial*. Esto se debe a que, como se verá en apartados posteriores de este mismo capítulo, se va a llevar a cabo una integración del presente Proyecto Fin de Carrera dentro de dicho proyecto, por lo que además de enriquecer y aumentar las funcionalidades de éste, se podrán aprovechar y reutilizar funcionalidades ya desarrolladas en la red social móvil. Así pues, el diagrama *UML* [UMLWebSite] de la base de datos de *LibreGeoSocial* queda recogido en la figura 3.2.

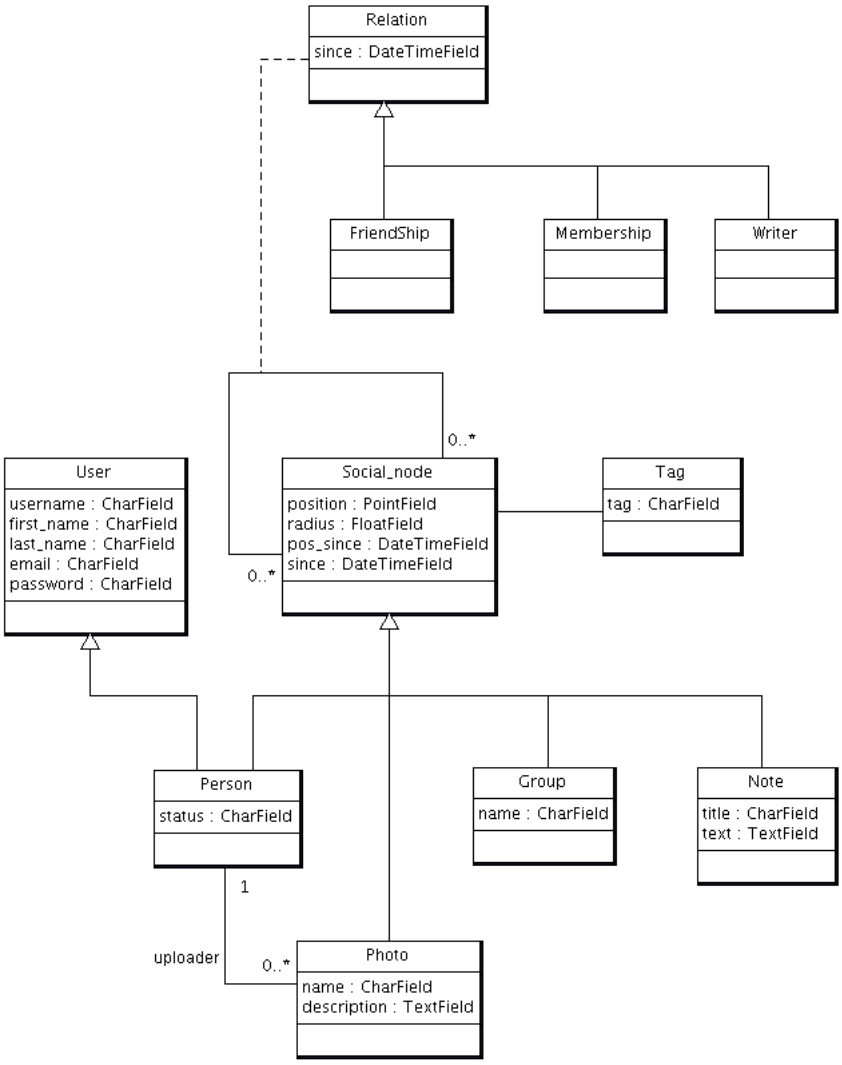


Figura 3.2: Diagrama *UML* de la base de datos del proyecto *LibreGeoSocial*

Como se puede observar en la figura 3.2, la estructura de la base de datos será la propia de una red social, pues como se ha señalado con anterioridad, *LibreGeoSocial* es precisamente una red social móvil (red social clásica a la que se añade la funcionalidad de tener a todos sus nodos geolocalizados). Así, en esta red social se pueden encontrar, fundamentalmente, usuarios-personas, grupos, notas y fotografías, heredando todos ellos las propiedades de un nodo de la red social (las relativas precisamente a la geolocalización). Y además, los nodos de la red social podrán estar interrelacionados entre sí de modo que, por ejemplo:

- Dos personas mantengan entre sí una relación de amistad.
- Una persona sea miembro de un grupo de la red social.
- Una persona escriba una nota dentro de la red social.
- Una persona suba una fotografía a la red social.

Una vez realizado este breve análisis, se está en disposición de afrontar el diseño realizado para la base de datos del Proyecto Fin de Carrera. Dicho diseño se presenta en la figura 3.3. En el diagrama *UML*, se puede observar que se incluirán dos nuevas entidades heredando de las personas de la red social. Estos tipos específicos de personas serán los organizadores y gestores de los eventos (*Manager*) y los miembros de los equipos que participen en cada una de esas *gymkhanas* (*TeamMember*). Los equipos (*Team*) estarán integrados por varios participantes, heredando los primeros las propiedades de los grupos de la red social, y como se estudió, también las propiedades de un *Social_ node* y por tanto, la posibilidad de geolocalización, al igual que ocurría con los organizadores y los participantes. Y al mismo tiempo, cada equipo contará con un marcador (*Scoreboard*) con su puntuación, retos respondidos correcta o incorrectamente, etc.

Por otro lado, una vez definido un evento o *gymkhana* (su lugar y fecha de celebración, textos de bienvenida y despedida a presentar a los participantes, etc.), habrá que definir los retos que formarán parte de la *gymkhana* (y cada uno de los cuales, podrá tener a su vez una serie de soluciones posibles, así como pistas conducentes a la resolución del reto, con un coste asociado). Precisamente estos retos serán a los que cada equipo ofrecerá sus respuestas (*Response*), siendo este elemento *Response* un nuevo componente más de la base de datos y que también heredará de *Social_ node* de manera que será un nuevo recurso geolocalizado de la red social. Y además, dependiendo del tipo de reto al que se responda (prueba de respuesta textual, prueba de respuesta fotográfica, prueba de geolocalización), esta respuesta hará referencia a un texto, a una fotografía o a la distancia en metros que separa al equipo del lugar al que se debe dirigir y que se indicó en el enunciado de la prueba.

Por último, dentro de las diferentes funcionalidades con que el sistema desarrollado cuenta y que aquí se comentan solamente de soslayo (pero en las cuales, por supuesto, se profundizará a lo largo de la memoria), se tiene un servicio de mensajería propio del sistema para la organización y participación de *gymkahanas*. Así, los mensajes (*Message*) serán también nodos geolocalizados de la red social móvil, y que podrán ser enviados por un organizador o un equipo de la *gymkhana*, teniendo como destinatario, dependiendo en cada caso, al *Manager* de la *gymkhana*, a un único equipo, o a todos los equipos inscritos.

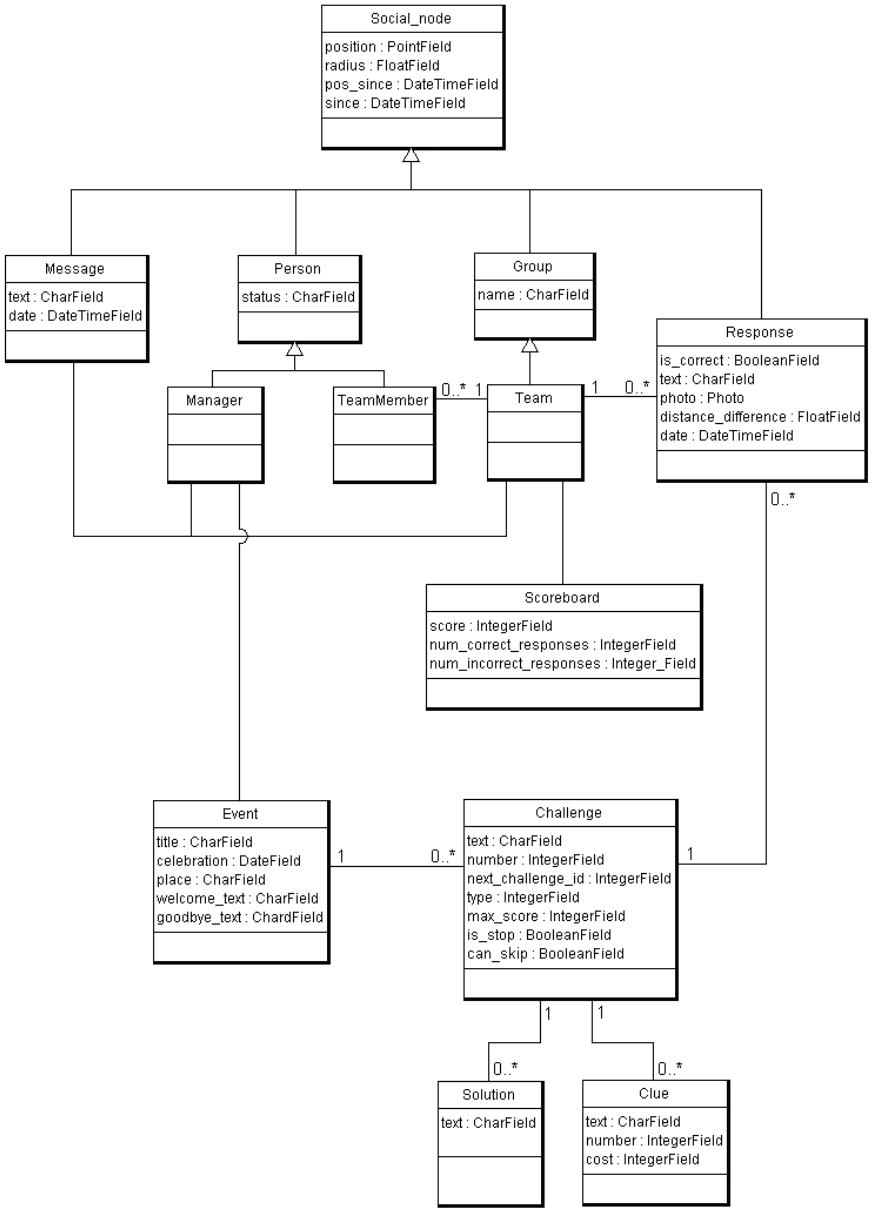


Figura 3.3: Diagrama *UML* del diseño realizado para la base de datos

3.2.2. Servicio *Web*

Integración en *LibreGeoSocial*

En primer lugar, se debe hacer especial hincapié en la integración llevada a cabo de este servicio de telecomunicaciones, dentro de la red social móvil *LibreGeoSocial*. Dicha integración se ha realizado a modo de *plug-in*, de manera que en cualquier momento se podrá incluir esta nueva funcionalidad dentro de *LibreGeoSocial*, con tan sólo copiar una serie de ficheros en el directorio adecuado dentro del árbol de directorios de la red social dinámica. Con esto se consigue por lo tanto una integración limpia desde el punto de vista de ingeniería de *software* y sin renunciar por ello a la reutilización de todas aquellas funcionalidades ya desarrolladas y que puedan resultar útiles a la hora de implementar otras nuevas. Por ejemplo, esto mismo ya se pudo observar en la sección anterior dedicada a la base de datos al reutilizarse funcionalidades propias de los grupos de la red social pero readaptadas a los equipos participantes en la *gymkhana*, etc.

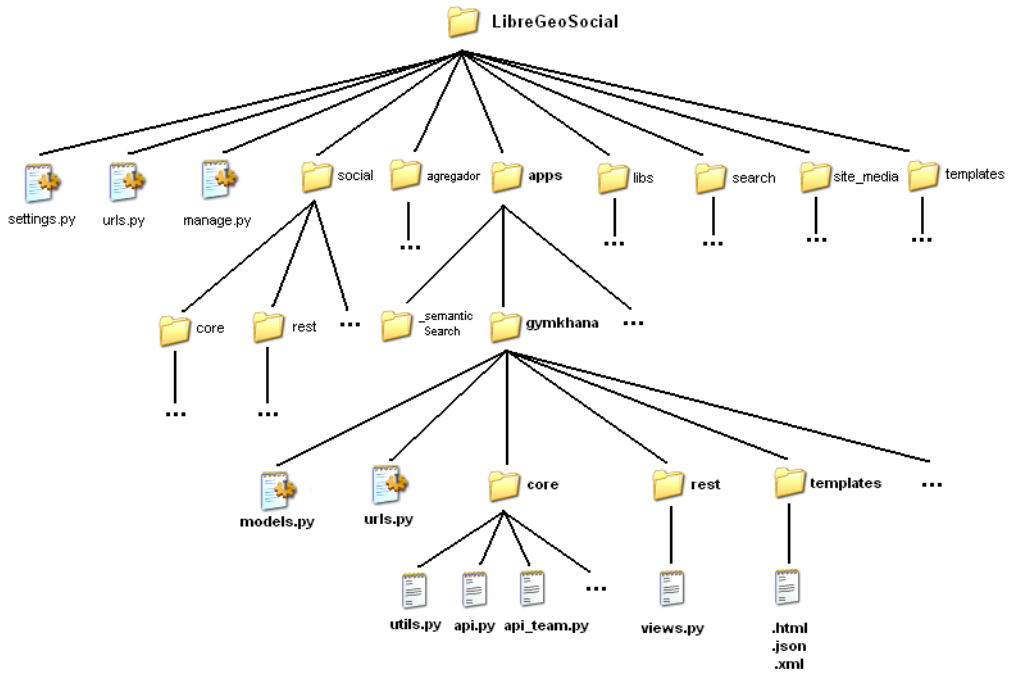


Figura 3.4: Diagrama del árbol de directorios de la integración del proyecto en el servidor de *LibreGeoSocial*.

Patrón de Diseño *MVC*

Como ya se estudió en el capítulo 2 destinado al Estado del Arte, el *framework* de desarrollo *Django* sigue el patrón de diseño conocido como *MVC* [Selfa:2006] (*Model-View-Controller*) pero con alguna particularidad que lleva a ciertos autores a definirlo realmente como un patrón o modelo *MTV* (*Model-Template-View*). Así, en el diseño seguido para estructurar el servidor, se ha decidido reforzar el uso de este patrón de diseño. Para ello, se ha desarrollado por un lado un *API REST* [Richardson:2007], y por otra parte, una librería que en adelante se denominará como *Core*.

De este modo, en el *Core* se realizarán las operaciones fundamentales de procesado, creación, modificación y destrucción sobre el modelo de elementos en la base de datos, mientras que el *API REST* actuará como interfaz de cara al cliente. Esto es, el *API REST* quedará encargada de recibir las peticiones provenientes de los clientes, y tras realizar una serie de comprobaciones de que en efecto esa petición está bien formada (*URL* [RFC1738] correcta; método *HTTP* [RFC2616] empleado es el adecuado en cada momento; formato en el que se solicita la respuesta está soportado por el servidor; autenticación y comprobación de permisos de cada usuario para la ejecución de las acciones solicitadas; paso de parámetros correctos tanto en su valor como en la cantidad de información necesitada para cada acción a ejecutar, etc.), despachará la petición haciendo entrar en ejecución aquellas funciones del *Core* que sean necesarias, ofreciendo a partir de ellas una respuesta en forma de documento que será generado a través de las plantillas destinadas a ello.

Aunando ahora los aspectos propios de la integración a modo de *plug-in* de este proyecto con el proyecto *LibreGeoSocial*, y con el uso de un *API REST* más un *Core*, se obtiene un esquema genérico como el mostrado en la figura 3.5.

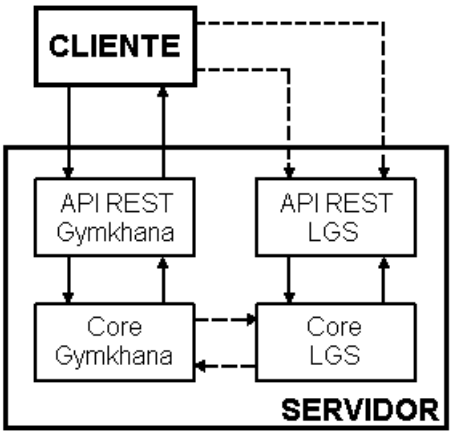


Figura 3.5: Diagrama de interacción entre cliente y servidor

Como se puede observar en la figura 3.5, las *API REST* interaccionarán con sus respectivos *Core* en cada petición que se reciba, para que sea éste último el que ejecute las operaciones que el cliente indicó en dicha petición de servicio. Así, lo habitual será que el cliente realice las peticiones sobre el *API REST* destinada al servicio aquí desarrollado. Sin embargo, habrá ciertas operaciones que el cliente deseé realizar pero que al estar ya implementadas en el *API REST* y *Core* del proyecto *LibreGeoSocial*, se reutilizarán de una manera limpia y sin causar interferencias sobre las mismas, debido a la integración a modo de *plug-in* que se está realizando.

Explicado de otra manera, con el diseño realizado en el servidor:

- El modelo de datos presentado en la sección anterior seguirá siendo la representación a modo de objetos de todas las tablas de la base de datos, y por lo tanto, la base fundamental de operación para el funcionamiento del servicio y definido en el fichero *models.py* de todo proyecto de *Django* [Holovaty:2006].

- La vista estará compuesta por el *API REST* (en este caso, el habitual fichero *views.py* de todo proyecto *Django*) y las llamadas plantillas o *templates* de *Django*, actuando siempre como interfaz con el cliente tanto para la recepción de peticiones como para la generación de los documentos en formato *HTML* [W3CHTML], *XML* [W3CXML, XMLWebSite], *JSON* [RFC4627, JSONWebSite], etc., a modo de resultado.

- El controlador se corresponderá con el *Core* de la aplicación, que se encarga de la manipulación del modelo de datos de acuerdo con el procesamiento que se indicó desde la vista, y cuya respuesta servirá como base para la generación del documento resultante a enviar al cliente.

Por lo que se consigue no sólo mantener el patrón de diseño *MVC/MTV*, sino que su uso queda reforzado con la consecuente acumulación de ventajas, proviniendo todas ellas del hecho de desagregar las diferentes funciones a desempeñar en diversos componentes independientes entre sí, siendo las ventajas obtenidas más destacadas:

- Un mismo modelo podría tener múltiples vistas y/o controladores.

- Las vistas son automáticamente notificadas de cambios en el modelo.

- Los modelos no tienen que sufrir modificación alguna para adaptarse a nuevos tipos de vistas o controladores.

- La desagregación de los tres componentes (*Model-View-Controller*) hace que el diseño y/o implementación de cada uno de ellos se pueda realizar en paralelo con el resto.

- Permite una mejor escalabilidad del sistema.

Pero, por contrapartida, también se encontrarán algunos inconvenientes, como por ejemplo:

- La complejidad del sistema aumenta al separar sus distintos componentes o conceptos.

- La cantidad de ficheros a desarrollar y mantener aumenta considerablemente.

URLs y Arquitectura *REST*

La implementación del servidor en un *API REST* y un *Core* queda justificada no sólo por las ya citadas ventajas del patrón *MVC* ampliamente extendido en aplicaciones y servicios *web*, sino también por las que acarrea el uso del estilo arquitectural para sistemas distribuidos (y especialmente para aplicaciones cliente-servidor) conocido como *REST* [Richardson:2007] (*REpresentational State Transfer*), ampliamente extendido en la *web* estática [Andersson:2006] (ficheros y directorios) pero tan sólo parcialmente en la *web* programática (aplicaciones y servicios *web*), y estando basado en un conjunto de reglas o normas:

1. La primera de estas reglas hace referencia a un aspecto tan importante en las aplicaciones y servicios *web* como es la definición de *URLs* (fichero *urls.py* del proyecto *Django*). Más concretamente, esta primera regla indica que cada recurso debe contar con su propia *URL*, entendiéndose por recurso cualquier tipo de información que se quiera hacer visible, independientemente de que se trate de documentos, imágenes, perfiles de personas, grupos de personas, etc. Así pues, el diseño llevado a cabo se puede observar en los ejemplos presentados a continuación, donde se tiene que dentro del servicio de *gymkhana*, se encuentran diversos eventos a cada uno de los cuales se referenciará mediante su identificador único, del cual colgarán igualmente equipos que ofrecerán sus respuestas a cada una de las pruebas del evento, etc., y pudiéndose finalmente indicar una acción a ajecutar sobre el recurso referenciado en la *URL*, de entre el conjunto de operaciones soportadas en el *API REST*.

```
http://server/resource/[identifier/]action/

http://server/gymkhana/event/list/
http://server/gymkhana/event/create/
http://server/gymkhana/event/1/team/list/
http://server/gymkhana/event/1/team/2/delete/
http://server/gymkhana/event/1/challenge/5/skip/
http://server/gymkhana/event/1/challenge/6/show/
```

2. La comunicación en una arquitectura *REST* debe ser sin estado, de modo que una petición del cliente debe contener toda la información necesaria para que el servidor pueda procesarla y ofrecer una respuesta (consúltese [RFC2965] para el estudio de los mecanismos de gestión del estado en *HTTP*). Puesto que no puede basarse en información previa asociada a la comunicación (como pueda ser la información de sesión), la manera de transmitir al servidor toda la que éste necesita, será bien mediante la propia construcción de las *URLs* (indicando el recurso sobre el que se desea realizar una determinada acción, etc.) y/o pasando valores adicionales como parámetros de la petición.

3. Cada recurso deberá contar con múltiples representaciones posibles, teniendo en cuenta que una representación muestra el estado actual de un recurso en un formato dado. Esto es, que una misma información podrá servirse en distintos formatos de documentos, ya sea *HTML*, *XML*, *JSON*, texto plano, formatos propietarios, etc.

4. Los recursos deben tener hiperenlaces a otros recursos, cosa que los formatos más utilizados como pueda ser *HTML*/*XHTML* [W3CXHTML] permiten, convirtiéndose en cierta manera en uno de los secretos de su éxito. También es de importancia contar con uniformidad en la presentación o interfaz del estado de los recursos.

5. Existe un conjunto estándar de métodos, que pese a ser bastante reducido, permite la realización de todas y cada una de las operaciones que se necesiten realizar sobre un recurso del sistema. Estos métodos son: *GET* (para obtener información, sin cambiar estado y siendo una operación idempotente), *POST* (para crear un recurso conocida la *URL* de un constructor de recursos, cambiando el estado y sin ser una operación idempotente), *PUT* (destinado a la actualización o creación de un recurso conocida su *URL*, cambiando el estado y siendo una operación idempotente) y *DELETE* (permite eliminar un recurso conocida su *URL*, cambiando el estado y siendo una operación idempotente). Realmente, estos métodos son precisamente parte de los definidos en el estándar *HTTP*. Sin embargo, el *WWW* ha hecho que se popularice el

uso del método *GET* para realizar peticiones de documentos, ficheros, etc., mientras que el método *POST* sea el utilizado para la transferencia de grandes cantidades de datos y habitualmente asociado al uso de formularios *HTML*. Esto ha hecho que en la actualidad, los clientes *HTTP* (como por ejemplo los *browsers* o navegadores) no soporten los métodos *PUT* y *DELETE*, por lo que no resulta sencillo cumplir con esta última regla *REST* (y siendo por tanto uno de sus principales inconvenientes, junto al hecho de que estos métodos *PUT* y *DELETE*, suelen ser cortados en los *firewalls* en caso de ser utilizados), tal y como ha ocurrido en este Proyecto Fin de Carrera.

Así, algunas de las ventajas que se obtendrán debido al diseño *REST* realizado, son las siguientes:

- Los mundos de las aplicaciones *web* y de los servicios *web* quedan unificados. De esta manera, se podrá crear tanto una aplicación *web* para que el servicio sea accesible por medio de un navegador, como responder a las peticiones de un hipotético cliente en cualquier formato que se soporte en el sistema, ya sea *JSON* o *XML* (además del ya citado *HTML*).

- Mejora considerable de la escalabilidad.

- Se maximiza la reutilización, al contar cada recurso con un identificador único propio.

- Rapidez y eficacia al configurarse un servicio *web* (*Web Services*) basado en el uso únicamente de *HTTP* y del formato de datos deseado (*HTML*, *XML*, *JSON*, etc.), especialmente en comparación con otras arquitecturas como por ejemplo *SOAP*, que supone una infraestructura completa basada en *XML* y con su propio protocolo para el intercambio de datos entre dos puntos finales.

3.3. Diseño de los Clientes

A continuación se detallarán los aspectos generales del diseño llevado a cabo en las aplicaciones *web* y *Android* [AndroidDeveloper] destinadas a los usuarios del servicio (organizadores de *gymkhanas* y participantes, respectivamente).

3.3.1. Interfaz *Web*

En el caso de la interfaz *web* que será accesible por parte de los organizadores de las *gymkhanas* a través de cualquier navegador o *browser*, su diseño no consumirá un tiempo importante dentro del desarrollo del Proyecto Fin de Carrera (aunque sí lo hará su implementación, como se verá en el capítulo 4).

Esta interfaz *web* no será más que una serie de páginas *HTML* por medio de las cuales el usuario, tras previa autenticación, podrá realizar acciones aisladas de una manera privilegiada (usuario administrador) cuya finalidad última será la manipulación de la base de datos con tal de dejar todo preparado para la participación en la *gymkhana*: creación de eventos, creación de pruebas, creación de equipos, suscripción de equipos a eventos, etc.

Igualmente, se deberá dotar a esta aplicación *web* de una serie de comportamientos determinados con tal de permitir un cómodo control, seguimiento y monitorización de todos los equipos participantes en la *gymkhana*. De este modo, ciertas informaciones deberán ser actualizadas periódicamente sin que el usuario perciba esas recargas: geolocalización de los equipos, respuestas recibibas en el servidor, mensajes internos del propio servicio para *gymkhanas* intercambiados entre equipos o de consultas a los organizadores, etc.

3.3.2. Aplicación *Android*

Como se ha venido señalando en apartados anteriores de la memoria, el tercer componente del sistema a construir será una aplicación para teléfonos móviles con sistema operativo *Android*. Esta aplicación *Android* instalada en los terminales que así lo soporten, será la empleada por parte de los participantes en la *gymkhana*, siendo sus principales usos el envío de respuestas a un reto dado, o la disposición de un sistema de mensajería propio para transmitir rápidamente cualquier tipo de duda o comunicar posibles problemas a la organización del evento.

Integración en *LibreGeoSocialApp*

Al igual que se estudió en el caso de la integración del servidor, en el servidor del proyecto *LibreGeoSocial* (apartado 3.2.2), la aplicación *Android* destinada a los equipos participantes de la *gymkhana* también será integrada en el proyecto. Concretamente, será integrada dentro del módulo denominado *LibreGeoSocialApp*, que es la aplicación *Android* de dicho proyecto y que se encarga de explotar la funcionalidad de la red social geolocalizada *LibreGeoSocial*. Y nuevamente, se integrará a modo de *plug-in*, de manera que bastará con copiar una serie de ficheros dentro de un paquete determinado

del proyecto para que la nueva funcionalidad quede disponible y operativa, o simplemente retirar ese paquete *Java* cuando así se requiera. Todo esto se hará además de manera que no se interfiera en ningún momento ni con la red social *LibreGeoSocial*, ni con la aplicación *LibreGeoSocialApp*. Además de esto, se ganará la posibilidad de reutilizar funcionalidades ya desarrolladas, y todo ello consiguiendo una integración limpia con un fácil mantenimiento y conservando unas buenas propiedades de escalabilidad.

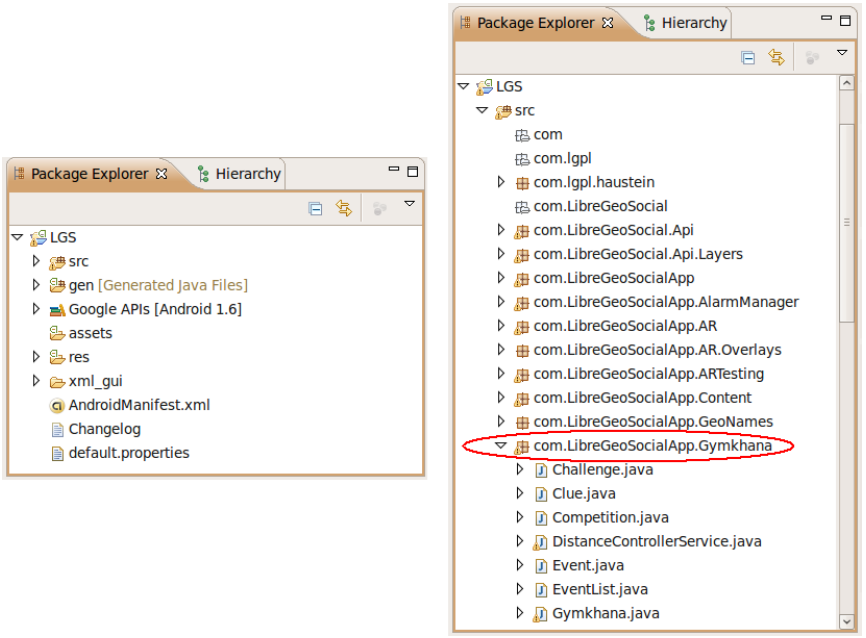


Figura 3.6: Diagrama del árbol de directorios de la integración del proyecto en la aplicación *Android LibreGeoSocialApp*.

Diagrama de Flujo

En el desarrollo de aplicaciones *Android*, existe un concepto fundamental que es el de *Activity* [Meier:2008, Burnette:2008, Gramlich:2008]. Un *Activity* de *Android* (es decir, una clase de la aplicación que hereda de la clase *Activity*) es, en términos coloquiales, cada pantalla que se le presenta al usuario dentro de una determinada aplicación, y en la que el desarrollador podrá introducir una *Interfaz Gráfica de Usuario* (*GUI*) destinada a dicho usuario, estando a su vez esa *GUI* definida en una instancia de otra clase (la clase *View* [AndroidDeveloper]).

Por lo tanto, dado que la aplicación *Android* será la utilizada por los usuarios del sistema (los participantes de la *gymkhana*), y por ende, la imagen última que se pueda generar en ellos sobre el mismo, será en este módulo en el que se deberá tener especial cuidado, tanto en las acciones que el usuario podrá realizar, como en la manera de ejecutarlas, su presentación, etc.

Puesto que cada *Activity Android* supone una nueva pantalla de la aplicación, ha llegado el momento de definir la verdadera lógica que se desea seguir en el servicio, pues hasta el momento en la interfaz *web* tan sólo se ejecutarán acciones aisladas que provocarán la ejecución de determinadas funciones del servidor. Pero en el caso de la aplicación *Android* será donde se deba definir el flujo del sistema, pues no se debe olvidar que se está trabajando en todo momento sobre el protocolo de comunicaciones *HTTP* (y por tanto, el cliente será siempre quien inicie la comunicación con el servidor y solicitando la ejecución de determinadas acciones con un cierto orden preestablecido en forma de peticiones de servicio).

Por consiguiente, la parte fundamental del diseño aquí presentado consistirá precisamente en la elaboración del diagrama de flujo que seguirá la aplicación *Android* y que servirá como base para la posterior implementación de la aplicación en cuanto a flujo de pantallas presentadas al usuario (se hace notar que en la figura 3.7 tan sólo se presenta el diagrama de flujo que seguirá la aplicación de la *gymkhana* una vez lanzada desde *LibreGeoSocialApp*, en la cual está integrada).

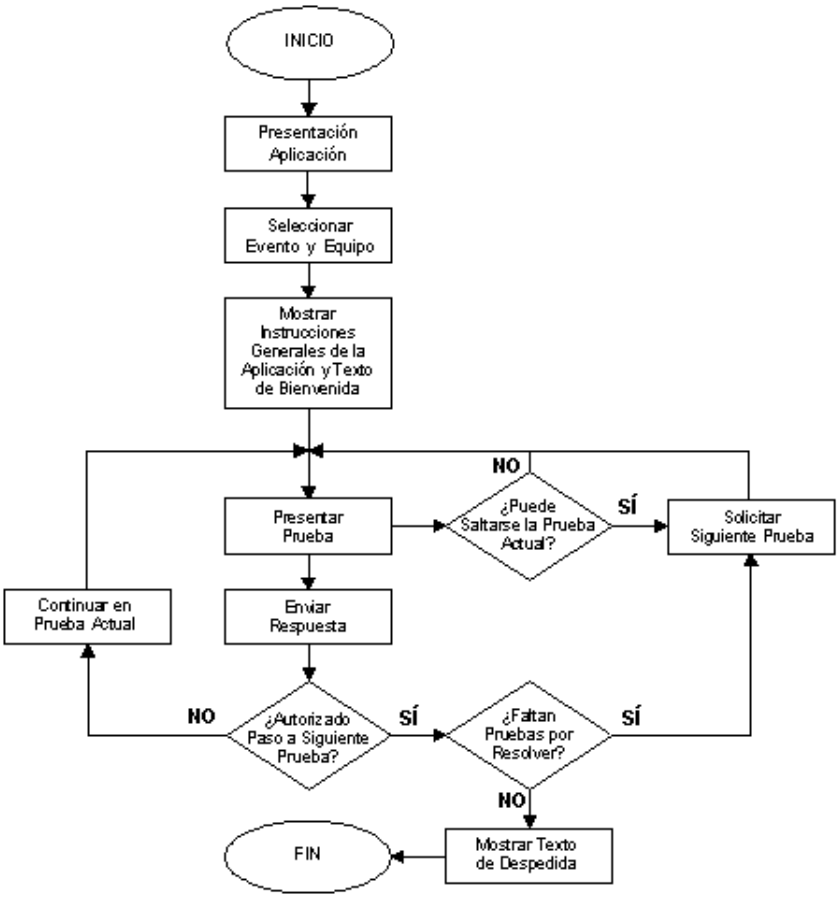


Figura 3.7: Diagrama de flujo de la aplicación en el cliente *Android*

Como se puede observar en la figura 3.7, se recoge un diagrama de flujo simplificado de la aplicación *Android* que cumple con la funcionalidad que se desea conseguir. Si bien es cierto que este diagrama tan sólo hace referencia al eje vertebrador de la aplicación (desde la selección de evento y equipo, hasta la presentación de cada prueba y finalización de la *gymkhana* una vez que todas las pruebas hayan sido superadas), en cualquier momento se podrá romper ese flujo habitual, haciéndolo llevar hacia un nuevo diagrama de flujo que se corresponderá con el de cada una de las funcionalidades adicionales que se quieran añadir. Un ejemplo de esto último es que durante el transcurso de una prueba, un equipo podrá optar por realizar una consulta sobre la misma a los organizadores. Para ello, bastará con que el flujo de la aplicación sea llevado hacia esa nueva funcionalidad consistente en un sistema de mensajería propio de la aplicación aquí desarrollada, mostrando nuevas pantallas a modo de *GUI*, etc.

Capítulo 4

Implementación

Cuando llegue la inspiración,
que me encuentre trabajando.

Pablo Ruiz Picasso

Una vez implementado el sistema descrito en el capítulo anterior y cumpliendo con los requisitos funcionales marcados como objetivo, en este capítulo se presentan los aspectos más destacados de esa implementación resultante.

4.1. Introducción

Tal y como se estudió en el capítulo anterior, dedicado al diseño del sistema, el problema ha sido resuelto mediante la agregación e interacción de varios módulos funcionales, todos ellos muy diferentes tecnológicamente entre sí. Para el análisis de la implementación final del sistema diseñado, se tomará como punto de partida la figura 4.1 y que al mismo tiempo servirá a modo de recordatorio de la estructura deseada.

Como se puede apreciar en la figura, finalmente se implementó un servidor basado en un *framework* de desarrollo *web Django*, y una base de datos *PostgreSQL*. En cuanto a la elección del *framework*, se seleccionó *Django* además de por las múltiples bondades que éste presenta y que se indicaron someramente en el capítulo 2 dedicado al Estado del Arte, porque es el *framework* empleado en el proyecto *LibreGeoSocial*. El uso de cualquier otro *framework* o tecnología (*Ruby on Rails*, *JSPs* y *Servlets* de *Java*, etc.) habría sido igualmente válido, pero en este caso, se habría renunciado a la integración del presente Proyecto Fin de Carrera en *LibreGeoSocial*. Y además, habría conllevado un mayor esfuerzo y trabajo al tener que realizar la implementación completamente desde cero y sin poder reutilizar ciertas funciona-

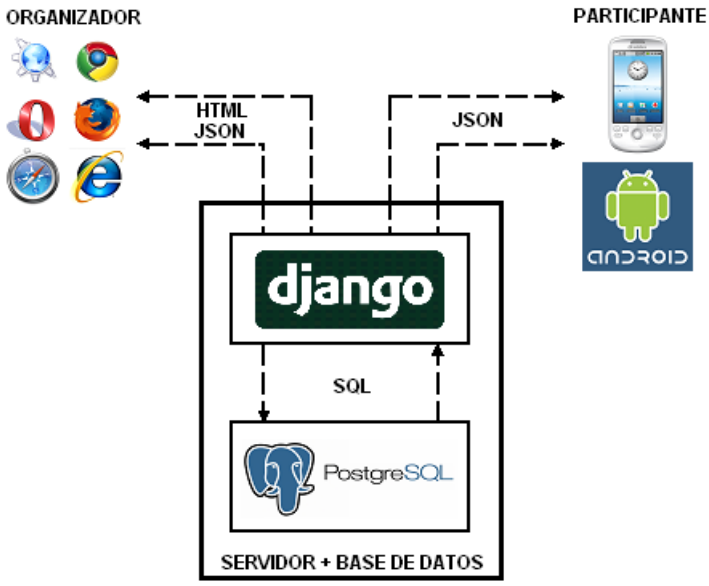


Figura 4.1: Esquema general del sistema

lidades propias de la red social ya desarrolladas y que resultarán de utilidad en el presente proyecto. Por su parte, se eligió el gestor de bases de datos *PostgreSQL* [PostgreSQLWebSite] además de por ser el empleado en *LibreGeoSocial*, la base de datos recomendada para *Django* en fase de producción y por tratarse de un *software* libre (al igual que todas las plataformas involucradas en el proyecto), debido a su soporte para geolocalización, pues se debe tener presente que todos los nodos de la red social (usuarios, pero también grupos, fotografías, notas, etc., y por lo tanto también equipos de la *gymkhana*, mensajes internos de la aplicación, etc.) estarán geolocalizados.

En cuanto a los clientes, se tiene por un lado la implementación de una interfaz *web* que será empleada por los organizadores de la *gymkhana* y accesible a través de cualquier navegador *web*. Y por el otro lado, se tiene la implementación de una aplicación *Android* que será soportada por aquellos terminales de telefonía móvil que cuenten con el citado sistema operativo, y empleada por los equipos de la *gymkhana* para participar en ella.

Puesto que la estructura del servicio es la de una arquitectura cliente-servidor [Márquez-García:1996], como el propio nombre indica, cualquiera de los dos tipos de cliente realizará peticiones a ese servidor. Para llevar a cabo la comunicación e interacción entre dichos clientes y servidor, en todo momento se empleará el protocolo de comunicaciones *HTTP* (*HyperText Transfer Protocol*) definido en la *RFC 2616* en su versión *1.1* [RFC2616] (para *HTTP/1.0*, véase la *RFC 1945*), por lo que, como se puede entender, siempre será el navegador *web* o la aplicación *Android* el ente que comience la comunicación con el servidor.

Respecto a los formatos con los que se codificará la información transmitida a nivel de aplicación entre los extremos, en el caso del navegador y su comunicación con el servidor se realizará fundamentalmente a través de *HTML* puesto que se pretende la visualización de páginas *web*. Pero habrá ciertas funciones para las cuales se empleará también el lenguaje de marcado *JSON* [JSONWebSite, RFC4627], dada la facilidad de su parseo en *JavaScript* [Flanagan:2007, ECMA262] cuando se está trabajando con *AJAX* [Zakas:2006, Eguíluz-Pérez:2007] (en este aspecto se profundizará en posteriores apartados del presente capítulo) frente a otros estándares que se podrían haber empleado como por ejemplo *XML* [W3CXML] o incluso un lenguaje propietario. Por su parte, la aplicación *Android* se comunicará por medio de *JSON* (igualmente se podría trabajar por ejemplo con *XML*, dado que se está ofreciendo un servicio *web* y sólo será necesario que el servidor soporte el formato de respuesta solicitado en la petición *HTTP* del cliente), pues se tendrá todo lo necesario para generar en base a ello las consecuentes *GUI* de la aplicación, además del sencillo parseo de los documentos *JSON* cuando se trabaja con lenguaje *Java* [Naughton:1996], como es el caso de *Android*.

Un aspecto fundamental de estas peticiones de servicio reside en el hecho de que lo habitual será que requieran la comunicación última con una base de datos [Miguel-Castaño:1999], ya sea para realizar una consulta, creación de registros, modificación o actualización, supresión, etc. Para ello, *Django* tras la recepción y procesado de la petición, realizará la manipulación oportuna sobre la base de datos *PostgreSQL* en aras de generar una respuesta que se enviará al cliente a modo de resultado. Además, esta comunicación entre servidor *web* y base de datos se llevará a cabo por medio del lenguaje *SQL* (*Structured Query Language*), aunque su uso no será necesario en la implementación realizada puesto que *Django* encapsula dichas peticiones *SQL* por medio de su propio lenguaje con sintaxis *Python* (gracias a la definición de los modelos de datos en el proyecto *Django* para que sean mapeados con la información almacenada en la base de datos, tal y como se señaló en el capítulo 2), haciendo el trabajo menos tedioso y más amigable al desarrollador.

4.1.1. Servidor

Una vez realizado el análisis general a modo de introducción, se debe señalar que si bien en el capítulo dedicado al diseño (capítulo 3) se prestó especial atención al estudio del modelo de datos y del servidor en sí (su integración en *LibreGeoSocial*, modelo *MVC*, definición de *URLs*, etc.), en el caso de la implementación no se realizará mayor estudio.

Esto se debe a que la creación de la base de datos, en base al paso del modelo entidad-relación (mostrado en el apartado de diseño del modelo de datos del capítulo 2) al modelo relacional, o dicho de otra manera, el paso a tablas

y columnas de la base de datos, no cuenta con una especial relevancia dentro del Proyecto Fin de Carrera. Lo mismo ocurre en el caso de la lógica del servidor, pues una vez analizada la solución y estructuración que se va a adoptar para el mismo (*API REST*, *Core*, etc.), carece de interés la implementación en código fuente de las funcionalidades que debe soportar dicho servidor y que realmente se irán desgajando a medida que se presente la implementación de los clientes (tales como el listado, muestra, creación, edición, eliminación, etc., de *gymkhanas* o eventos, pruebas, equipos, respuestas, mensajes, etc., así como operaciones relacionadas con el *login* y *logout*, comprobación del soporte de formatos de respuesta, permisos de usuario, etc.). Por tanto, lo realmente relevante en términos de la implementación llevada a cabo en este proyecto, son las aplicaciones cliente desarrolladas y con las que los usuarios del sistema interaccionarán en primera instancia, independientemente de la estructura interna del sistema.

4.1.2. Interfaz *Web*

Tal y como se ha explicado a lo largo de la memoria, la implementación de una aplicación *web* está justificada por el hecho de que será la manera ideal para que los organizadores procedan a la creación de eventos, pruebas, etc., así como la monitorización de las acciones de los equipos durante el transcurso de cada evento.

En un primer momento se contempló la posibilidad de implementar en *Android* esta aplicación destinada al *Manager*. La razón de esto residía principalmente en poseer un único interfaz desde el cual pudiera interactuar todo tipo de personas involucradas en el evento (organizadores y participantes). De esta forma también se tenía la posibilidad de que el organizador contara con total movilidad dentro del área de realización de la *gymkhana*.

Sin embargo, finalmente se descartó esta posibilidad y se creyó mucho más oportuna la implementación de estas funcionalidades por medio de una aplicación *web* en lugar de una aplicación *Android*. Esto se debe a que a la hora de introducir la información sobre las *gymkhanas* en la base de datos (normalmente se tratará de una gran cantidad de textos largos: retos, soluciones, pistas, etc.), resulta mucho más cómodo y rápido hacerlo en un ordenador a través del navegador, que en el teléfono móvil por medio de la pantalla táctil o en el mejor de los casos, del pequeño teclado *QWERTY* que pueda incluir el terminal. Esto en principio puede provocar una falta de movilidad para los organizadores, pues deberán estar sujetos a su ordenador en un puesto con acceso de red. Pero teniendo en cuenta los pequeños ordenadores portátiles llamados *mini/netbook* cuyo uso prolifera en la actualidad, así como los *pen-drive* (llamados comercialmente *módem USB*) destinados a ordenadores portátiles por medio de los cuales los operadores de telefonía

ofrecen acceso a su red de datos, hace que este inconveniente pase a un segundo plano. Y además, se debe tener en cuenta que la aplicación *web* será accesible también desde el teléfono *Android*, puesto que podrá accederse a ella por medio del navegador *web* que incorpora de serie. En cualquier caso, lejos de negarse el interés del desarrollo de una versión de la aplicación *Android* para el control de la participación en la *gymkhana*, queda propuesta su implementación como una futura línea de trabajo, que teniendo en cuenta la implementación del sistema a modo de servicio *web*, no requerirá de cambios en el servidor sino tan sólo una adaptación de esa aplicación *web* a Android.

Pasando al análisis de la implementación llevada a cabo finalmente, se realizará su estudio en base a la visualización de las páginas *web* más interesantes de la aplicación. Cabe mencionar que todas las páginas *web* han sido formateadas por medio de hojas de estilo *CSS* (*Cascading Style Sheets* [W3CCSS21], recomendación del *World Wide Web Consortium*, *W3C*), por lo que en cualquier momento se podrá cambiar la apariencia de la página sin necesidad de modificar el código *HTML*.

Creación de *Gymkhanas*

Al intentar acceder a la aplicación, el sistema requerirá al usuario que se autentique mediante un *nick* y una contraseña (figura 4.2). Estos datos se corresponderán precisamente a los de un usuario de la red social móvil *LibreGeoSocial*. De modo que de cara a un futuro, cualquier usuario de *LibreGeoSocial* podría acceder a esta aplicación *web* y por medio de la cual, informarse sobre próximos eventos, así como subscribirse a un determinado equipo para participar en una determinada *gymkhana*. Pero en cuanto a la parte que atañe ahora mismo, se tiene que para poder entrar en la aplicación de manera que se tenga acceso a todo tipo de acciones propias de un organizador de *gymkhanas* (creación y eliminación de eventos, etc.), ese usuario de *LibreGeoSocial*, deberá tener además la categoría de *Manager* dentro del sistema (véase el diseño del modelo de datos, en el capítulo 3).

Una vez que el *Manager* haya accedido al sistema, lo habitual será que proceda a la creación de una nueva *gymkhana*. Esta creación se realizará mediante la cumplimentación y envío de un formulario *HTML* con una serie de campos que se corresponderán precisamente con los datos del modelo para este tipo (título del evento, fecha, lugar, textos de bienvenida y cierre de la *gymkhana*). Cabe destacar también que si bien en el servidor *Django* se realizan todo tipo de comprobaciones sobre los datos recibidos con cada petición antes de acceder a la base de datos, con tal de cerciorarse de que todo es correcto (se tiene toda la información necesaria, con el formato adecuado, etc.) para evitar cualquier tipo de error en el sistema, estas comprobaciones también las realizará el navegador (indicadas por la aplicación *web*).

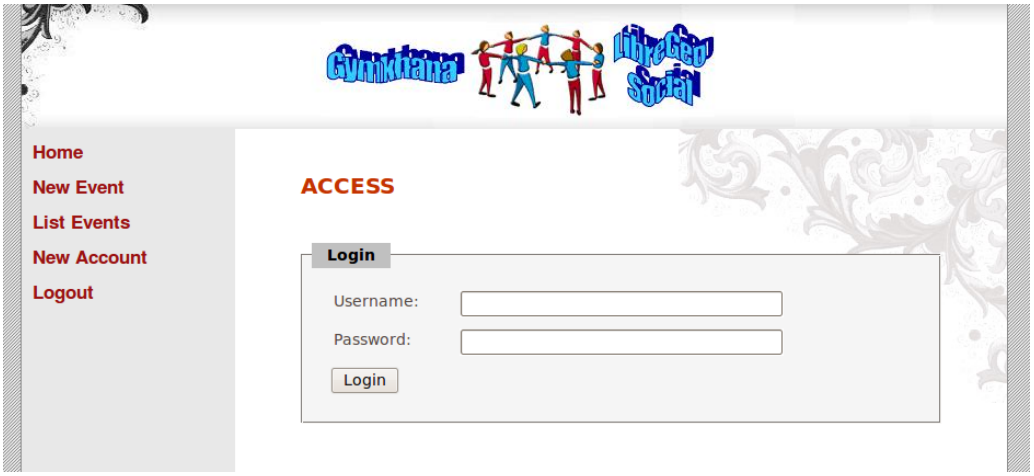


Figura 4.2: Página *web* para la autenticación de usuarios

Esas comprobaciones se realizarán por medio de *JavaScript*, código descargado junto con la página *web* y ejecutado ante ciertas acciones del usuario, como es en este caso ante el envío del formulario pulsando sobre el botón destinado a este fin (véase la figura 4.3). La razón de dar redundancia a estas comprobaciones es simple: obviamente es necesario en el servidor puesto que se debe evitar todo tipo de posibles errores en tiempo de ejecución, e incluso, evitar las posibles acciones de usuarios malintencionados que falsifiquen peticiones de servicio, manipulen la *URL* manualmente, etc. Pero la razón de llevar a cabo este control también en el interfaz reside en que siempre será más amigable una aplicación que avisa al usuario de las acciones que no está realizando correctamente y del error que está cometiendo, que una aplicación en la que simplemente se presenta una página de error sin información sobre lo que realmente ha sucedido.

Una vez creado un evento, o seleccionado uno de los presentados en la lista de creados con anterioridad, se podrá pasar a una nueva página *web* en la que se dispone de toda la información relativa al evento en cuestión. Una de las funciones primordiales a realizar en esta página será la creación de los retos a superar en cada una de las *gymkhanas*. Como se puede observar en la figura 4.4, tras introducir el texto con el enunciado de la prueba, el administrador deberá seleccionar el tipo de reto que se va a crear de entre las tres opciones existentes en la actualidad y que se estudiarán más detenidamente en el apartado de la memoria 4.1.3. Es decir, prueba de respuesta textual y su subtipo de realidad aumentada, de respuesta fotográfica y de geolocalización. Además de esto, y dependiendo del tipo de reto seleccionado, se podrán introducir algunos parámetros adicionales, como por ejemplo si se trata de una prueba que los equipos podrán saltarse yendo al siguiente reto, si se necesita una respuesta válida para considerar la prueba como superada, sus posibles soluciones o las pistas que ayuden a su resolución y superación.

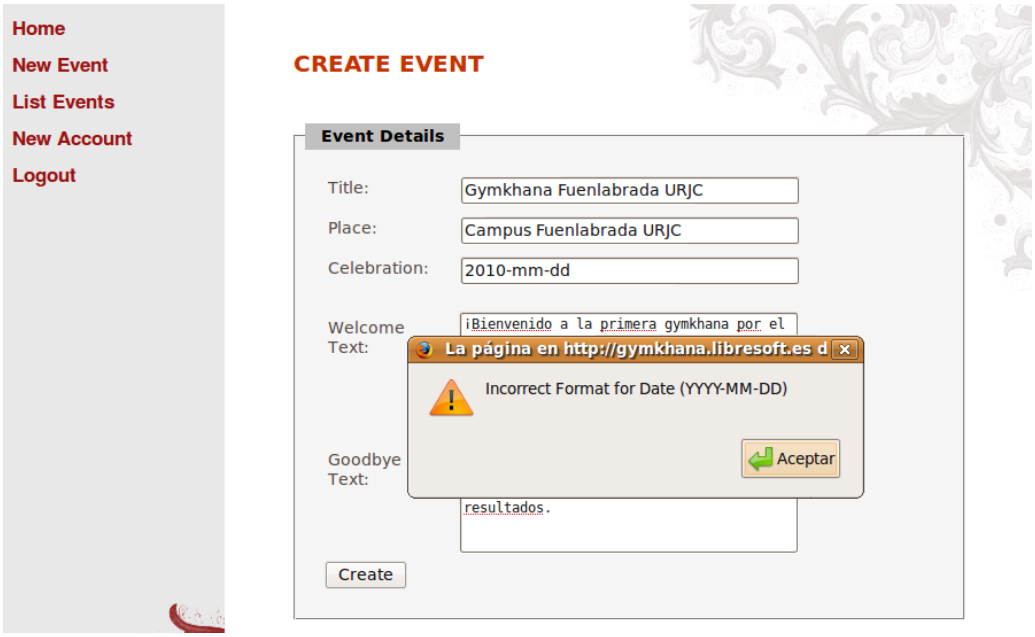


Figura 4.3: Página *web* para la creación de eventos con detalle de una alerta de *JavaScript*

Pero existe un parámetro adicional de especial interés. Este parámetro hace referencia a la posición o lugar que el reto ocupará dentro de la *gymkhana*. La implementación del recorrido que realizarán los equipos a través de las pruebas, se ha implementado a modo de lista enlazada [Weiss:2000]. De esta manera, el recorrido será circular (véase la figura 4.5), finalizando cuando se regrese a la prueba de inicio (cosa que se encargará de detectar el sistema), por lo que se podrá configurar a cada equipo para que inicie la *gymkhana* en un lugar distinto y por lo tanto, evitando que los participantes coincidan durante su participación.

Por supuesto, una vez creadas todas las pruebas de la *gymkhana*, se podrán consultar para comprobar que han sido configuradas tal y como los creativos las diseñaron. E igualmente, se podrán dar de alta en la aplicación equipos para su participación en la *gymkhana* (configurando siempre el reto en el que comenzarán su recorrido en la misma) así como subscribir a la *gymkhana* sobre la cual se está trabajando a aquellos equipos interesados en participar y que ya existan en *LibreGeoSocial* debido a que participaron en una *gymkhana* anterior. Del mismo modo, se podrá consultar la lista de equipos inscritos en la *gymkhana*, e incluso, seleccionando cada uno de ellos se pasará a una nueva página *web* en la que además de disponer de información detallada de cada equipo, también se podrán consultar los usuarios que pertenecen al mismo, así como las respuestas ofrecidas a cada reto planteado.

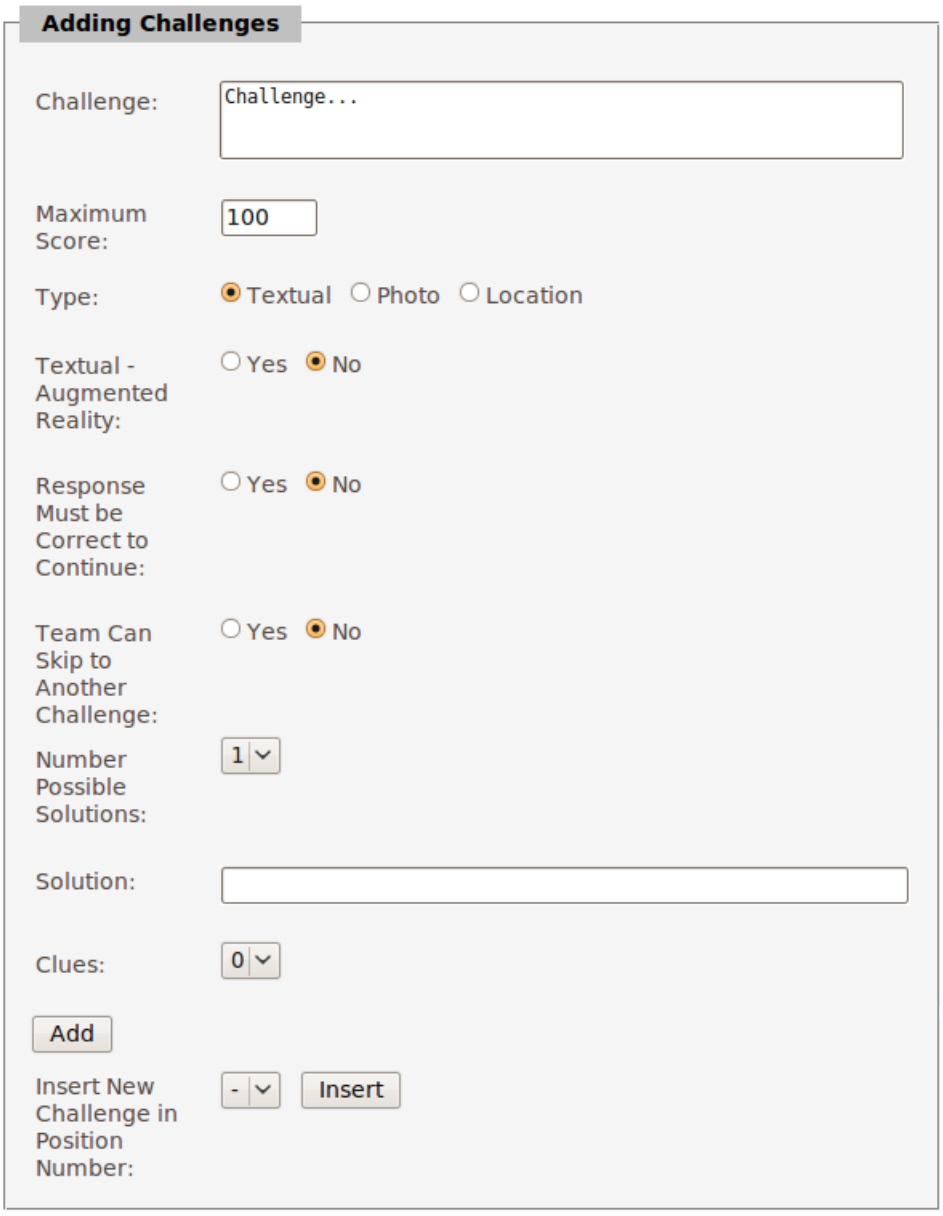


Figura 4.4: Detalle del formulario *HTML* para la creación de retos

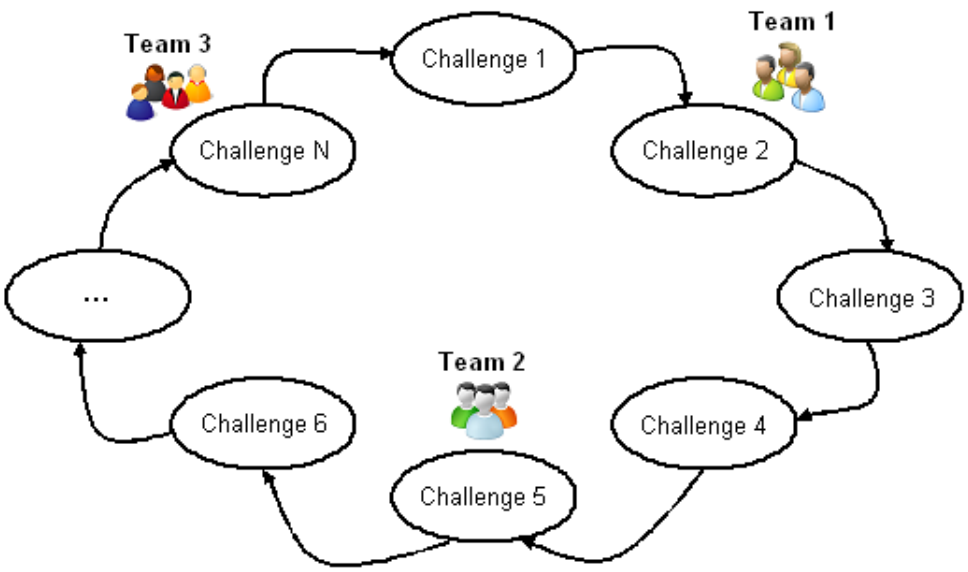


Figura 4.5: Ejemplo del escenario de una *gymkhana* recorrida por varios equipos, con las pruebas enlazadas circularmente

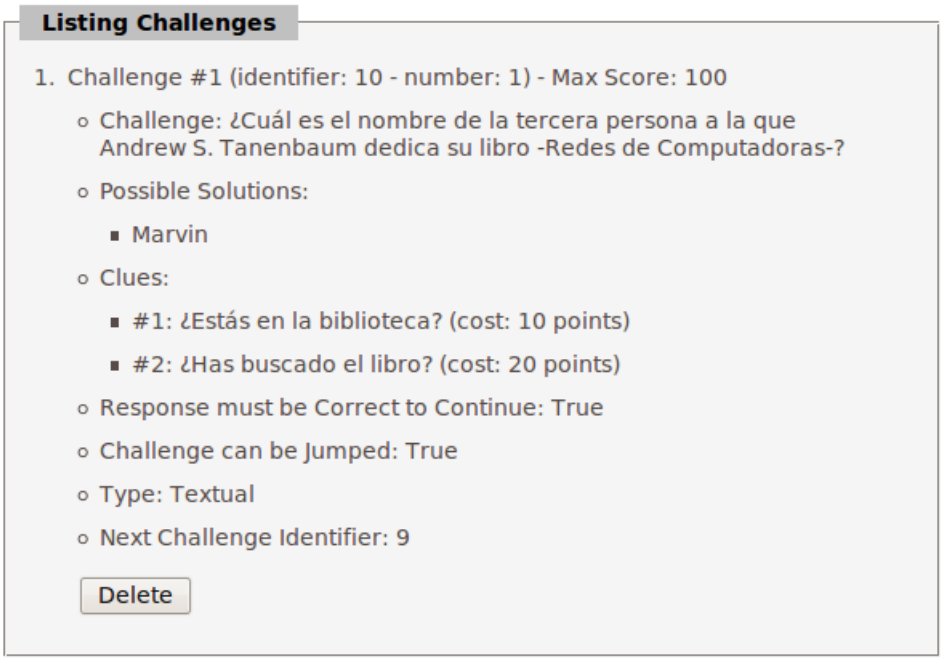


Figura 4.6: Detalle de la presentación de retos una vez creados

Control y Monitorización de *Gymkhanas*

Una vez creada la *gymkhana* e inscritos los equipos participantes, llegado el momento de la celebración del evento, lo que resultará interesante de cara al organizador será contar con la posibilidad de realizar un seguimiento a distancia de todas las acciones llevadas a cabo por esos equipos. Para ello, se cuenta con una página *web* adicional en la que se presentará al administrador toda la información que pueda resultar de utilidad. Entre esta información, se encuentra:

- Una tabla con la clasificación de los equipos en base a los puntos logrados incluyendo el número de respuestas correctas e incorrectas.
- Un *GoogleMaps* en el que cada marcador sobre el mismo indicará la ubicación geográfica de cada equipo participante en la *gymkhana*.
- Las respuestas que cada equipo ha enviado hasta el momento.
- Los mensajes intercambiados entre equipos y *Manager* empleando el servicio de mensajería interno de la propia aplicación para *gymkhanas*.

Realizando a continuación un análisis más detallado de la implementación de cada uno de los módulos indicados, antes de nada, y de manera genérica, se debe destacar el uso de la tecnología *AJAX*. Una vez cargada la página *web* (código *HTML* vía *HTTP*), el uso del objeto *XMLHttpRequest* [Vlist:2007, Zakas:2006, Eguíluz-Pérez:2007] de *AJAX* permitirá que el navegador, sin que el usuario lo perciba y sin necesidad de recargar la página *web* íntegramente, realice peticiones *web* al servidor. Esto permitirá que en esas peticiones *web* transparentes para el usuario, se solicite cierta información susceptible de cambio a lo largo de determinados períodos de tiempo, y actualizando también de manera transparente el contenido de la *web* con esa información ahora actualizada.

En esta página *web* habrá tres informaciones que serán solicitadas periódicamente por medio del objeto *XMLHttpRequest* de *AJAX* que se ha señalado. La primera de esas informaciones es la referida a cada equipo participante. Entre la información que el servidor transmitirá como respuesta a dicha petición, se encontrará el nombre de los equipos participantes en la *gymkhana*, la puntuación que han obtenido hasta el momento así como las respuestas que han ofrecido de manera correcta o incorrecta. Con esta información se estará en disposición de todo lo necesario para elaborar la tabla clasificatoria que se presenta en la parte superior de la *web* (figura 4.7).

Debajo de la citada tabla con la clasificación de los equipos según la puntuación obtenida por cada uno, se encuentra un *GoogleMaps*. Para incluir estos mapas, es necesario en primer lugar obtener una clave para hacer uso desde el servidor de la *gymkhana* del *API* de *GoogleMaps*[1]. El moti-

[1]Sitio *web*: *http://code.google.com/intl/es-ES/apis/maps/signup.html*

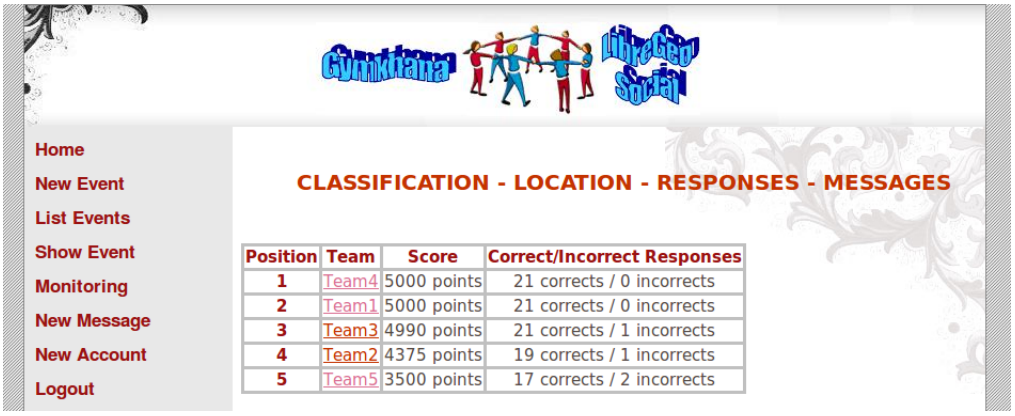


Figura 4.7: Tabla de clasificación de los equipos participantes en la *gymkhana*

vo de incluir este mapa reside en el hecho de que gracias a la información solicitada al servidor en *background* relativa a los principales datos de cada equipo, se encuentra la geolocalización de cada uno de ellos. Con esta información de la ubicación de cada equipo obtenida vía *AJAX*, y junto con el *GoogleMaps* incrustado en la *web*, se dará origen a un *mashup* [Pimpler:2009, MashupTutorial, Vlist:2007] en el que presentar la información de manera muy clara e intuitiva al organizador de la *gymkhana*. Este *mashup* consistirá precisamente en incluir sobre el mapa un marcador relativo a cada equipo y en la ubicación en que cada uno de ellos se encuentre. Además, cuando el usuario pulse sobre cada uno de esos marcadores, aparecerá en pantalla un desplegable en el que se incluirá la información más relevante del equipo situado en ese lugar, tal como el nombre, coordenadas geográficas, puntuación y número de respuestas correctas e incorrectas.

El tercer bloque que se puede encontrar en esta página *web* incluye las respuestas ofrecidas por los equipos (véase la figura 4.9). Al igual que en los casos anteriores, esta información será solicitada periódicamente al servidor por medio de la tecnología *AJAX* y sin que el usuario lo perciba, actualizando el contenido del elemento * * [W3CHTML] destinado a esta sección con la nueva información recibida una vez parseada y generado el nuevo código *HTML* que deberá ocupar ese lugar en la página. En este bloque, el organizador podrá consultar cada respuesta dada por un equipo a cada reto, viendo si la respuesta textual es correcta, si la fotografía se corresponde con lo solicitado (además, por medio de un efecto *JavaScript* generado gracias a la librería *overLIB* [OverLIBWebSite], cada vez que el usuario posicione el cursor del ratón sobre la imagen, ésta se verá ampliada, útil en el caso de que el *Manager* necesite visualizar la fotografía a un mayor tamaño) o la distancia a la que se encontró el equipo de la ubicación señalada en la prueba de geolocalización. Por supuesto, también se podrá consultar si la respuesta se ha catalogado como correcta o incorrecta.

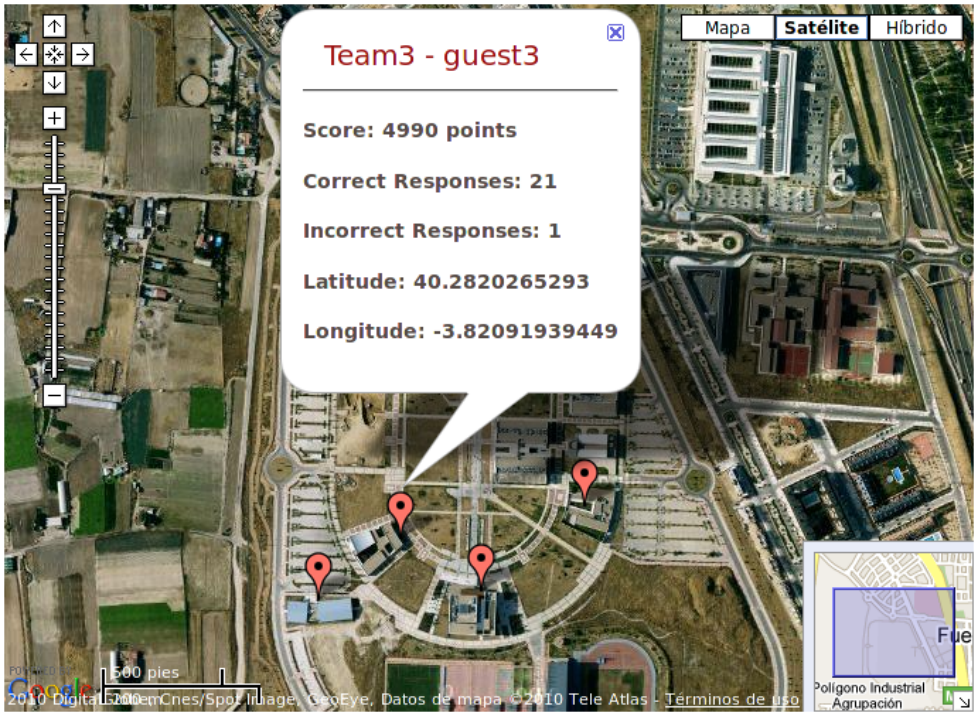


Figura 4.8: Detalle del *mashup* de *GoogleMaps* con la geolocalización e información de cada equipo participante

Además, en cualquier momento el organizador podrá cambiar la respuesta de incorrecta a correcta y viceversa. Esto se debe a que en una prueba de respuesta textual, pueden existir múltiples formas distintas de ofrecer una respuesta a un reto o de escribir dicha respuesta, y que no fueran contempladas en el servidor en el momento de la corrección, pero una vez analizada, pudiera darse como una respuesta válida. Esto resulta de vital importancia en las pruebas fotográficas. Dada la imposibilidad de realizar una verificación automática de la corrección o no de una imagen cualquiera como respuesta a un reto, se tomará como partida que la respuesta fotográfica es incorrecta, siendo tarea del *Manager* de la *gymkhana* la verificación de la respuesta y su posterior cambio al estado de correcta en caso de ser así.

El cuarto y último bloque de información que se podrá consultar en la *web* contiene los mensajes del servicio de mensajería interno implementado para facilitar la comunicación en cada *gymkhana*. Igualmente se obtendrán periódicamente cada cierto tiempo los mensajes intercambiados en la aplicación y se mostrarán al organizador por medio de esta interfaz *web*. Sin embargo, dada la importancia de este servicio así como su complejidad, se realizará un análisis en profundidad de esta funcionalidad en el apartado 4.1.4.

Listing Challenges-Responses

Team: **Team1(identifier: 1)**
Challenge 1 (identifier: 8)
Challenge: ¿Cuántos escalones tiene la escalera de caracol más famosa del Campus de Fuenlabrada?
Possible Solutions:

- 52
- Cincuenta y dos

Response must be Correct to Continue: True
Response: 52
Is Correct: True
Geolocation: latitude: 40.2813559771; longitude: -3.81960511208; altitude: 676.0
Date: 2010-02-11 15:59:42.184963

Change to Correct/Incorrect

Delete Response

Team: **Team3(identifier: 3)**
Challenge 6 (identifier: 107)
Challenge: Dirígete al Aulario Polivalente IV (al punto marcado en el mapa).
Response must be Correct to Continue: False
Distance Difference: 28.241186 meter/s
Is Correct: True
Geolocation: latitude: 40.2813881636; longitude: -3.82190108299; altitude: 731.0
Date: 2010-04-16 11:13:50.213254

Change to Correct/Incorrect

Delete Response

Team: **Team3(identifier: 3)**
Challenge 7 (identifier: 108)
Challenge: Hazte una foto con algún alumn@ de la Escuela de Danza Alicia Alonso, debidamente ataviad@ con su ropa de ensayo.
Response must be Correct to Continue: False
Response:

Is Correct: True
Geolocation: latitude: 40.281291604; longitude: -3.82224440575; altitude: 732.0
Date: 2010-04-16 11:15:58.951297

Change to Correct/Incorrect

Delete Response

Figura 4.9: Detalle de la presentación de las respuestas de los equipos a cada reto

4.1.3. Aplicación *Android*

En este punto se analizará la implementación llevada a cabo de la aplicación *Android* destinada a la participación en la *gymkhana* por parte de los equipos inscritos. Como se puede entender, los equipos participantes en la *gymkhana*, a través de esta aplicación podrán obtener los retos a superar, ofrecer las respuestas a los mismos, comprar pistas en caso de que necesiten ayuda para superar una prueba en concreto, etc. Como se indicó en el capítulo 3 dedicado al diseño del sistema, esta aplicación *Android* será integrada a modo de *plug-in* en la aplicación *LibreGeoSocialApp* del proyecto *LibreGeoSocial*, consistente en una red social móvil con interfaz de realidad aumentada. Así pues, una vez implementada e integrada la aplicación, para hacer uso de ella tan sólo se necesitará accederla a través de *LibreGeoSocialApp*. Es decir, tal y como se puede observar en las capturas de pantalla del emulador[2] presentadas en la figura 4.10, una vez arrancada *LibreGeoSocialApp* en el móvil, se procederá a la autenticación del usuario (*nick* más *password*) en el servidor. En caso de que el participante de la *gymkhana* no disponga ya de un usuario creado para el acceso a la red social, siempre podrá crear uno nuevo por medio de las opciones que se le ofrecen. Una vez el usuario se haya autenticado y accedido a *LibreGeoSocial*, tan sólo deberá buscar entre los menús la opción que lo lleve hacia la aplicación destinada a las *gymkhanas*.

Figura 4.10: Pantallas de autenticación en *LibreGeoSocial* y opciones del menú de la red social móvil

[2]En adelante, todas las imágenes de la aplicación *Android* serán capturas de pantalla del emulador dada la mejor visualización que se obtiene respecto a fotografías del propio teléfono.

Llegados a este punto, la primera pantalla que verá el usuario será la relativa a la presentación de la aplicación (figura 4.11), así como diversa información sobre la aplicación y el desarrollador con tan sólo pulsar la opción "*About*" accesible desde el menú de esta pantalla. Cuando el usuario pulse sobre el botón destinado al comienzo, se pasará a dos pantallas consecutivas que se podrían catalogar como de configuración. En la primera de ellas, tras solicitar al servidor la información de todos los eventos existentes en la base de datos, se presentará al usuario una lista con los títulos o nombres de cada una de esas *gymkhanas* en las que se podrá participar. La *GUI* de esta pantalla ha sido implementada por medio de la clase *ActivityList*, de modo que una vez que el usuario pulse sobre la *gymkhana* en la que desea participar (acción que además se le notificará por medio de un *Toast* del *API* de *Android*, es decir, un pequeño desplegable en la pantalla que tras un breve instante de tiempo, desaparecerá tras haber informado al usuario), automáticamente se pasará a la siguiente pantalla (recuérdese que en *Android*, el paso de una pantalla a otra estará asociado al paso de una clase y por tanto de una *Activity* a otra, acción que se llevará a cabo por medio de un *Intent* que indicará cuál es la siguiente actividad que se debe lanzar en la aplicación).

Figura 4.11: Captura completa del emulador de *Android* para su versión *1.6* con la presentación de la aplicación para *gymkhanas* dentro de *LibreGeoSocial*

En esa siguiente pantalla, se indicará al usuario que su próxima acción deberá ser la selección del equipo del que quiere formar parte (nuevo *Toast* de *Android* para realizar la notificación al usuario). La lista de equipos inscritos en la *gymkhana* seleccionada por el participante en la pantalla anterior, será solicitada al servidor *web* inmediatamente después de la selección de esa *gymkhana*, información que como en todos los casos, será siempre recibida en formato *JSON* para su posterior parseo en lenguaje *Java* empleado

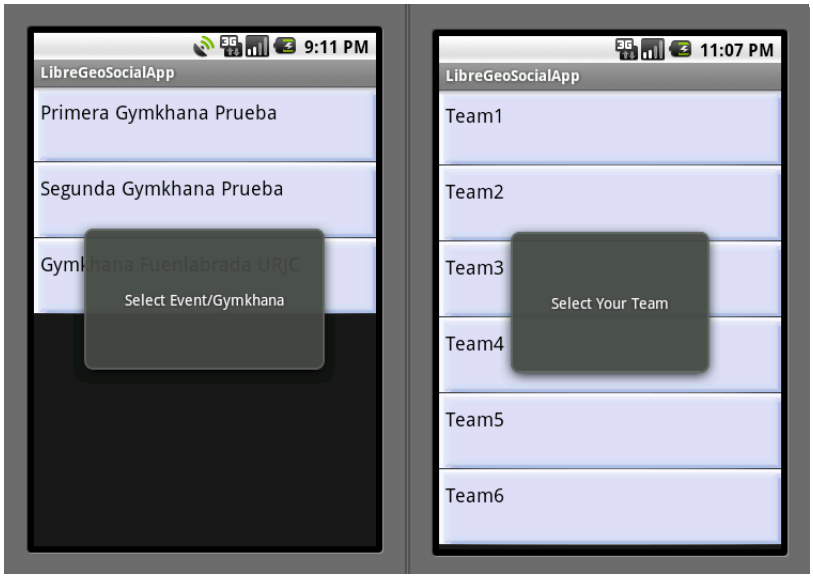


Figura 4.12: Pantallas para la selección de *gymkhana* y equipo

en *Android* (y normalmente, cada objeto recibido en formato *JSON*, ya sea un evento, un equipo o por ejemplo un mensaje, será mapeado a un objeto *Java* modelado según una clase propia de la aplicación y que en cierto modo, se corresponderá con los modelos de datos con que trabaja el servidor). Igualmente, se presentará la lista de equipos posibles gracias a una *GUI* conseguida mediante la herencia de la clase *ActivityList*.

Una vez que el usuario haya seleccionado la *gymkhana* y su equipo, se pasará a una nueva pantalla en la que antes de nada, se mostrará un diálogo a modo de alerta (clase *AlertDialog* de *Android* destinada a este tipo de notificaciones de usuario, tomando el primer plano de ejecución independientemente de la *Activity* que estuviera siendo ejecutada en ese momento). En este cuadro de diálogo, se informa al usuario sobre aspectos relacionados con el uso del terminal así como de la propia aplicación, cosa más que necesaria con tal de que el participante que se encuentra por primera vez ante un teléfono *Android*, consiga explotar todas las funcionalidades de la aplicación y sin encontrar problemas con el manejo del terminal. Cuando el usuario pulse sobre el botón de *OK* de este *AlertDialog*, podrá leer el texto de bienvenida a la *gymkhana* que el organizador configuró a través de su interfaz *web*. Así pues, lo último que el usuario deberá hacer antes de afrontar la primera prueba de la *gymkhana*, será pulsar sobre el botón que indica el comienzo de la misma (véase la figura 4.13).

En la nueva clase *Challenge* de la aplicación *Android* (nueva *Activity* y por tanto, nueva pantalla de la aplicación) se presentará al usuario fundamentalmente el reto que deberá superar. Como se ha indicado a lo largo de la memoria, actualmente existen cuatro tipos distintos de pruebas para las

Figura 4.13: Diálogo con información sobre el uso del terminal y de la aplicación, y texto de bienvenida a la *gymkhana*

gymkhanas organizadas con este sistema: pruebas de respuesta textual y su subtipo de realidad aumentada, de respuesta fotográfica y de geolocalización. Pero la interfaz propia de cada tipo de prueba así como los aspectos diferenciadores de cada una de ellas en términos de implementación, serán presentados con detalle en subapartados posteriores.

Además del reto a superar, tal y como se puede observar en la figura 4.14 el usuario podrá consultar también el nombre de usuario con el que está participando en la *gymkhana*, así como si el equipo y el evento que seleccionó son los correctos. Esta información se recoge en la *GUI* por si fuera necesaria durante el transcurso de la *gymkhana*, y principalmente, para evitar posibles problemas futuros como por ejemplo con la propia pantalla táctil en el momento en que el usuario realizó la selección de evento y equipo. De hecho, esto sucedió en el primer experimento realizado en un escenario real, lo que motivó la inclusión de esta información en las pantallas. Por otro lado, también se pueden observar tres pestañas en la parte superior de la pantalla:

- La primera de ellas, ya comentada y mostrada siempre por defecto, la que recoge la información relativa al reto planteado en cada momento, así como su puntuación, etc.
- Una pantalla en la que el equipo dispondrá de un mapa de *Google* en el que además, podrá consultar en todo momento su geolocalización. Esta pantalla cobrará especial relevancia en las pruebas de geolocalización, como se estudiará más adelante.

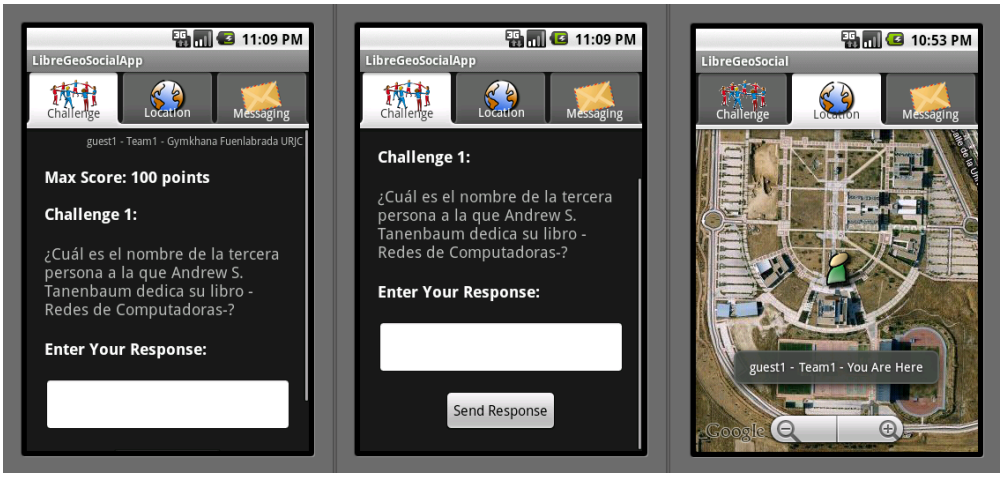


Figura 4.14: Pantallas para la presentación del reto a superar y consulta de la geolocalización del equipo en *GoogleMaps*

- Una última pestaña destinada al servicio de mensajería interno del sistema para la organización y realización de *gymkhanas*, y que será estudiado en profundidad en el subapartado de la memoria 4.1.4.

Durante el uso de la aplicación, será recomendable que el usuario consulte las opciones que en cada pantalla se le puedan ofrecer por medio del menú (botón físico del terminal móvil). Por ejemplo, en todas y cada una de las pantallas que a partir de ahora se presenten al usuario, siempre se dispondrá en el menú de una opción de ayuda o información sobre el uso y finalidad de esa pantalla de la aplicación. Pero también se incluirán, dependiendo del tipo de pantalla en que el usuario se encuentre en cada momento, diversas opciones, como pueda ser de configuración ("*Settings*") en el caso del servicio de mensajería, o mucho más importantes aún todas las que el usuario tendrá a su disposición en la pantalla de presentación de pruebas.

Como se puede ver en la figura 4.15, en determinadas situaciones será totalmente imprescindible que el usuario consulte el menú de esta pantalla de presentación de retos con tal de conseguir avanzar a lo largo de la *gymkhana*. Además de la opción con la ayuda (opción "*Information*"), el equipo podrá dar por finalizada su participación en la *gymkhana* siempre que lo quiera o que no contemplen posibilidad alguna de superar los retos restantes y cuya resolución ya afrontaron sin éxito (opción "*Finish*"). También se pondrán a disposición del equipo una serie de pistas conducentes a la superación de cada prueba (siempre y cuando el organizador creyera oportuno contar con ellas y las introdujera durante la creación de esa prueba en concreto) y que serán adquiridas a cambio de algunos de los puntos obtenidos por el equipo en la *gymkhana* (opción "*Give Me a Clue*"). E igualmente, el equipo podrá consultar las pistas adquiridas en cualquier momento (opción "*Show Clues*").

Figura 4.15: Captura de las posibles opciones del menú de cada reto planteado

Por último, existen dos opciones más y que están estrechamente ligadas. La primera de ellas hace referencia a la posibilidad de que el equipo quiera saltar al siguiente reto, dejando el actual pendiente de resolución, ya sea porque no saben resolverlo, porque lo consideran demasiado complejo y costoso en cuanto a tiempo, o por cualquier otra razón (opción "*Skip Challenge*"). En cualquier caso, esta posibilidad igualmente se presentará en aquellas pruebas que fueran configuradas por el *Manager* como posibles retos a postponer durante el transcurso del evento. En relación a esto, siempre que el equipo se haya saltado alguna prueba a lo largo de su recorrido por toda la *gymkhana*, se presentará una opción adicional en el menú. En esta opción adicional (opción "*Retry Challenge*"), se dará al equipo la posibilidad de retomar cualquiera de los retos que en algún momento decidió saltarse.

Cabe señalar como nota general a toda la implementación, que debido a los resultados obtenidos en el primer experimento realizado con el sistema aquí propuesto, se decidió incluir una barra de progreso ante determinadas acciones dentro de la aplicación, como se puede observar en la figura 4.16. Esta barra de progreso (clase *ProgressDialog* del *API* de *Android*) es presentada al usuario en cada ocasión que el teléfono realiza una petición *HTTP* al servidor, ya sea para solicitar la lista de *gymkhanas* o equipos, para el envío de una respuesta a una prueba o la solicitud del siguiente reto a superar, etc. Esto se debe a que, dada la alta latencia que supone la conexión con el servidor a través de la conexión *UMTS* [Kaaranen:2006] del móvil, los usuarios percibían una lentitud en la aplicación que en determinados momentos incluso podría asimilarse como un cuelgue o fallo en la misma, cuando realmente la ejecución estaba realizándose y de manera siempre correcta.

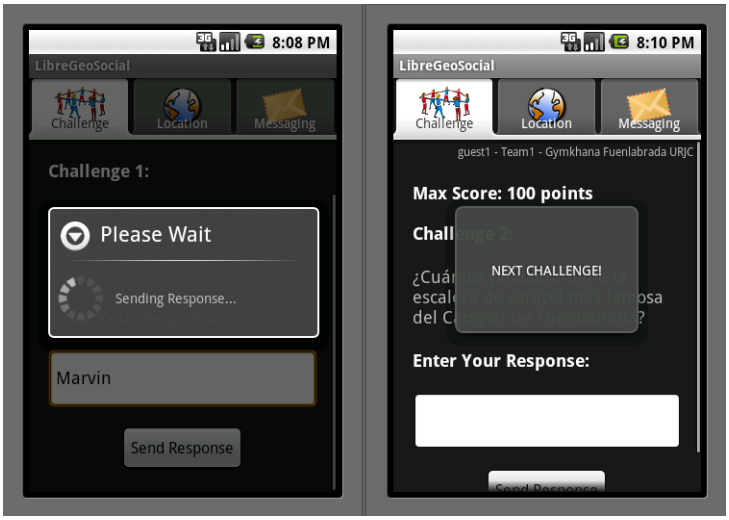


Figura 4.16: Captura del uso de diálogos de progreso y notificaciones de usuario

Finalmente, una vez que los equipos hubieran superado todos los retos de la *gymkhana*, o dándola por finalizada mediante la opción "*Finish*" del menú, se presentará una nueva ventana de despedida en la que se dará al equipo cualquier tipo de información relativa, por ejemplo, al punto de reunión para conocer el resultado de la *gymkhana*, etc., así como de manera provisional hasta que el organizador supervise todos los resultados, el número de respuestas correctas e incorrectas que ha cosechado el equipo en la *gymkhana* (véase la figura 4.17). Tras pulsar el botón para finalizar, se volverá a la red social móvil *LibreGeoSocial*.

Figura 4.17: Pantalla de finalización de la *gymkhana*

Tipos de Retos

En los cuatro siguientes subapartados se analizará meticulosamente la implementación llevada a cabo en el cliente *Android* para cada tipo de prueba que se puede encontrar actualmente en el sistema:

- Retos de respuesta textual.
- Retos de respuesta fotográfica.
- Retos de geolocalización.
- Retos de realidad aumentada.

Para realizar este análisis, se necesitará en algunos puntos hacer referencia a la implementación llevada a cabo de esto mismo pero en el lado servidor.

Retos de Respuesta Textual

Como se indica en el propio título de la sección, a modo de denominación de este primer tipo de pruebas, los retos de respuesta textual serán aquellos ante los cuales el equipo deberá responder mediante el envío de un *string* o cadena de texto que será evaluada como correcta o incorrecta en el servidor, y en última instancia supervisada por el organizador o *Manager*. En la figura 4.18 se puede apreciar que tras el enunciado del reto, se presenta un cuadro de texto en el que el usuario deberá introducir su respuesta (ya sea por medio del teclado virtual que se desplegará sobre la pantalla táctil cuando se pulse sobre el cuadro, o por medio del teclado físico *QWERTY* en caso de que el terminal lo tenga incorporado). En caso de que el usuario pulse el botón para enviar la respuesta sin haber introducido ningún texto, este error le será notificado mediante un *Toast*.

Para la validación automática en el servidor de la respuesta ofrecida por un equipo, se tiene que, como ya se vio en el apartado de este capítulo destinado a la implementación del servidor, el organizador podrá introducir varias respuestas posibles. Esto se debe a que una misma respuesta podría ser escrita de múltiples formas diferentes pero todas ellas igualmente válidas. Un claro ejemplo de esto sería el nombre de una persona. Concretamente, las posibilidades a la hora de responder a la pregunta presentada en la figura 4.18 serían las siguientes:

- Alfred Hitchcock
- A. Hitchcock
- A.Hitchcock
- Hitchcock

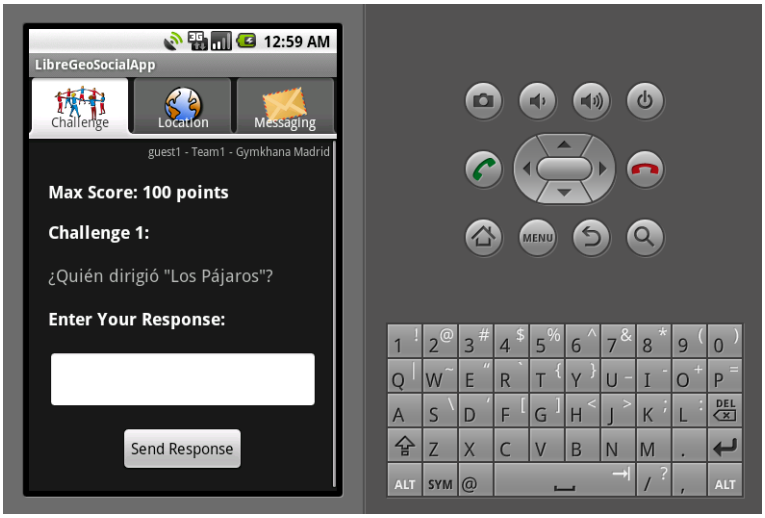


Figura 4.18: Pantalla para la presentación de retos de respuesta textual

Todas las respuestas hacen referencia a la misma persona, por lo que la solución no es única y de ahí la posibilidad de que el organizador contemple varias respuestas alternativas pero igualmente válidas. De este modo, será suficiente que la respuesta dada por el equipo coincida con alguna de esas posibles soluciones contempladas para que el servidor la detecte como válida. Además, también se puede tener en cuenta la posibilidad de que el usuario se equivoque en algún carácter a la hora de introducir la respuesta. Por esto mismo, resulta interesante introducir en el servidor algún tipo de heurística basada en algoritmos como el de la *Distancia de Levenshtein* [Yujian:2007] o cualquiera de sus variantes mediante el cual calcular la diferencia entre dos cadenas de caracteres (la posible solución y la respuesta del equipo) en base al número mínimo de operaciones necesarias para transformar uno de los dos *string* en el otro. En cualquier caso, siempre se tendrá en último lugar la posibilidad de que el organizador supervise las respuestas textuales ofrecidas por los equipos, y su corrección en caso de que hayan sido mal verificadas por el servidor de manera automática.

Por otro lado, dentro de las pruebas de respuesta textual, podrá haber pruebas en las que el organizador requiera una respuesta correcta por parte del equipo para darla como buena y por tanto que se pase al siguiente reto, o bien, que independientemente de que la respuesta sea o no correcta, se pase a la siguiente prueba del recorrido en la *gymkhana*. En caso de requerirse una respuesta correcta para continuar con la siguiente prueba, pero el equipo haya enviado una respuesta inválida, le será notificado esto mismo al equipo. Sin embargo, cuando la respuesta sea correcta, y en aquellos retos en los que el organizador no requiere una respuesta correcta, simplemente se notificará al equipo el paso a la siguiente prueba y sin ofrecer mayor información sobre la resolución del reto.

Retos de Respuesta Fotográfica

Este tipo de reto será todo aquél en el que se pida al equipo que envíe una fotografía de un determinado lugar, monumento, edificio, persona, objeto, etc. En la figura 4.19 se puede observar la interfaz gráfica a través de la que el equipo hará frente a la prueba. Además del propio enunciado del reto a superar, se pueden apreciar tres botones y una imagen:

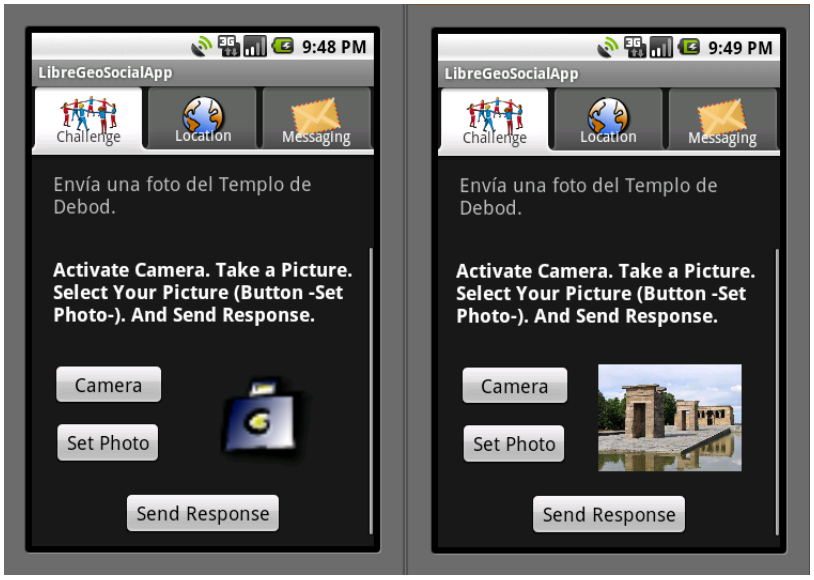


Figura 4.19: Pantalla de presentación de retos fotográficos antes y después de que el usuario seleccione la fotografía a enviar como respuesta

- El botón "*Camera*" mediante el cual se podrá lanzar la cámara de fotos digital incorporada en el teléfono. Cuando la cámara es lanzada desde la propia aplicación, por *software*, el móvil ofrece al usuario la posibilidad de realizar la fotografía, y una vez captada, se darán dos opciones más: dar la fotografía como buena, en cuyo caso, se cancelará la vista de la cámara fotográfica para retomar la ejecución de la aplicación desde la cual fue lanzada; y la posibilidad de volver a hacer la fotografía, repitiéndose el proceso hasta que el usuario quede satisfecho con la fotografía realizada y pulse la opción *OK*. Una vez que se retorne a la ejecución de la aplicación *Android*, el desarrollador se encargará de almacenar la fotografía en un fichero con formato para imágenes, en la memoria externa del teléfono y a partir del *BitMap* (clase del *API* de *Android*) que la actividad de la cámara siempre devuelve.

  Cabe señalarse en este punto la existencia de un más que probable *bug* de *Android*, como bien demuestran los múltiples hilos al respecto en los principales foros para desarrolladores en esta plataforma. Este *bug*

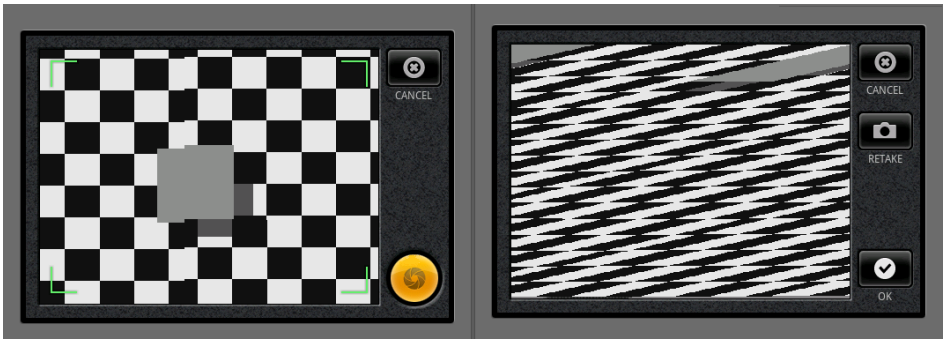


Figura 4.20: Simulación del uso de la cámara lanzada desde la *GUI* de un reto fotográfico

consiste en que una vez que la actividad de la cámara es finalizada y se vuelve a la ejecución de la aplicación principal, el *BitMap* que se devuelve representa una versión escalada de la imagen captada, concretamente, una imagen de alrededor de $6KB$. Según la documentación de *Android*, esto se hace por defecto, pero si se quiere obtener la imagen original, sin ningún tipo de escalado ni manipulación, basta con pasar a la actividad el parámetro $EXTRA\_OUTPUT$ acompañado de un valor que se refiere al lugar donde se quiere almacenar la fotografía dentro del sistema de ficheros. Sin embargo, tal y como se ha podido comprobar en la práctica, esto no ocurre así y por lo tanto, he aquí donde se encuentra ese posible *bug*.

En un primer momento esto se consideró como un inconveniente, por lo que se buscó una alternativa. Esa alternativa fue suprimir ese botón para el lanzamiento de la cámara desde la aplicación, de modo que fuera el usuario quien tuviera que lanzarla manualmente (y posteriormente, finalizar su ejecución) pulsando el botón *hardware* del teléfono destinado a la activación de la cámara, en cuyo caso, las fotografías son almacenadas sin necesidad de intervención del desarrollador y además, a tamaño completo. Sin embargo, dada la incomodidad que supondría para el usuario esta dinámica, más teniendo en cuenta el período de adaptación que el usuario necesita ante cada cambio de teléfono móvil, así como la probable supresión del *bug* en liberaciones futuras de versiones de *Android*, esta posibilidad fue descartada. De hecho, existe un motivo adicional por el que esta alternativa dejó de contemplarse, en favor de la implementación original y finalmente presentada. Esto es, que dado el pequeño tamaño de las fotografías una vez escaladas, el envío al servidor será prácticamente inmediato. Teniendo además en cuenta que tal y como se ha comprobado en la práctica, el tamaño de las fotografías una vez que el organizador las consulta a través de su interfaz *web*, es más que suficiente para esta aplicación. E incluso, en el

hipotético caso de que algún teléfono contase con una tarificación por datos en base al tráfico cursado en la red, el ahorro económico que el envío de unos pocos *KiloBytes* supondría, frente al envío de $1MB$-$2MB$, podría ser sustancial.

- El botón "*Set Photo*" da paso en la aplicación a la galería de imágenes almacenadas en el teléfono. A través de esa galería, el usuario podrá seleccionar la fotografía que desea enviar como respuesta al reto con tan sólo pulsar sobre ella y tras lo cual, automáticamente se volverá a la ejecución de la aplicación principal.

- Inicialmente, a la derecha (contemplando el teléfono móvil de frente) de los botones "*Camera*" y "*Set Photo*", se muestra un pequeño icono con una cámara de fotos. Una vez que el usuario haya seleccionado la fotografía a enviar como respuesta, en su lugar se mostrará una pequeña miniatura de dicha imagen, gracias a lo cual el usuario podrá asegurarse de que en efecto se enviará aquella que seleccionó.

- Por último, se encuentra el botón para enviar la respuesta ("*Send Response*"), que en este caso, será la fotografía seleccionada, teniéndose especial cuidado en el control de errores, pues en caso de que no se haya seleccionado ninguna imagen en el momento de pulsar sobre este botón, el usuario será notificado de ello y por lo tanto, no se procederá a realizar ningún envío.

Como apreciación final, dada la imposibilidad de verificar una respuesta fotográfica de manera automática en el servidor, por defecto, dicho servidor considerará todas las respuestas a este tipo de pruebas como incorrectas. Será tarea de la organización de la *gymkhana*, monitorizar y validar las respuestas que los equipos estén ofreciendo en cada momento, cosa que podrá hacer fácilmente a través de la interfaz *web* que ya se analizó y donde por supuesto, se le presentarán las imágenes recibidas como respuesta a una prueba.

Retos de Geolocalización

El último tipo de reto que actualmente se puede plantear será el de geolocalización [Román-López:2009]. Estas pruebas consistirán en solicitar al equipo que se dirija a un determinado lugar. Ese lugar vendrá determinado por las coordenadas geográficas (longitud y latitud) que el organizador introdujera en el sistema en el momento de creación de la prueba a través de su interfaz *web*. Dada la práctica imposibilidad de que el equipo alcance exactamente las coordenadas fijadas como objetivo (además del propio error del *GPS* [Angulo-Martínez:2009] del teléfono móvil que puede estar en torno a los diez metros, por los propios errores en el cálculo de la posición que

se pueden generar en un entorno urbano con edificios altos y que limiten la visibilidad directa con los satélites, etc.), también se establecerá un radio de distancia máximo en metros en torno al lugar objetivo, dentro del cual se considerará la prueba como superada.

En este tipo de pruebas el equipo no deberá realizar ningún envío de respuesta como se vio en las pruebas textuales y fotográficas, sino que todo el proceso se realizará de manera automática y transparente para el usuario de la aplicación, como de hecho se puede comprobar en la figura 4.21, donde se presenta la *GUI* para estos retos. En el momento en que el cliente *Android* solicite la información relativa al reto a superar, y ese reto sea de geolocalización, el servidor enviará también junto con esa información, las coordenadas geográficas acompañadas del radio máximo que puede separar al equipo de esas coordenadas para dar el reto como superado. A partir de este momento, será en el propio teléfono móvil donde, gracias al *LocationListener* (clase del *API* de *Android*) para la lectura de la geolocalización del terminal[3], se calcule y consulte periódicamente (empleando un *Service* de *Android* ejecutando en *background*) la distancia en metros entre las coordenadas geográficas del lugar objetivo y las coordenadas geográficas del equipo.

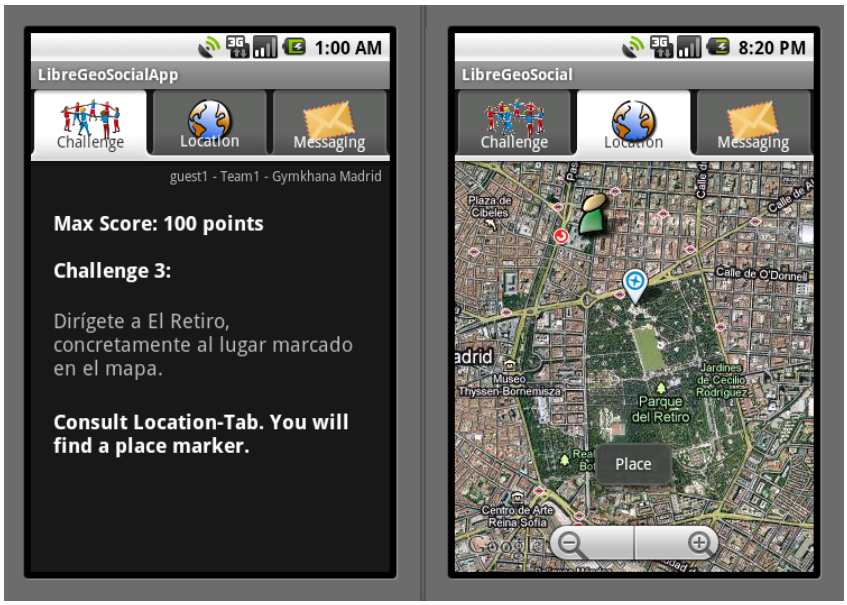


Figura 4.21: Pantalla para la presentación de un reto de geolocalización y consulta del lugar de destino

[3]Inicialmente se utilizó el *LocationListener* ya implementado en *LibreGeoSocial*. Sin embargo, su uso se tuvo que desechar (en este aspecto se profundiza en el apartado 5.4.1) dado que en él se realizaban operaciones adicionales propias de la red social móvil, hecho que ralentizaba la ejecución de la aplicación para *gymkhanas*, por lo que finalmente se implementó y empleó un nuevo *LocationListener* más ligero sólo para esta aplicación.

El cálculo de la distancia en metros entre dos pares de coordenadas geográficas no es un problema trivial, como de hecho demuestra la rama de la geodesia dentro de la geología. Así, si se quisiera conseguir un cálculo exacto de esa distancia, el problema dependería de muchas variables de cuyos valores posiblemente no se dispondría, y aplicando distintos cálculos según los múltiples escenarios que se pudieran presentar: para segmentos cortos, la trayectoria geodésica (línea de mínima longitud que une dos puntos en una superficie dada, y estando contenida en dicha superficie) difiere poco de la trayectoria plana, es decir, la línea recta; para una esfera, la trayectoria geodésica son los círculos máximos (los meridianos son geodésicas pero no así los paralelos, salvo en el caso del Ecuador), siendo éste el modelo a emplear en el caso de separaciones grandes (varias decenas de kilómetros); a lo que además hay que añadir que la forma de la Tierra es la propia de un esferoide oblato, la posible presencia de obstáculos, etc.

Sin embargo, tal y como se ha podido comprobar experimentalmente, existe una expresión matemática que se puede emplear de manera genérica y consiguiendo resultados bastante precisos. Esta expresión es conocida como *Fórmula Haversine* [Sinnott:1984] y toma su nombre de la *Función Haversine* (ecuación 4.1, función trigonométrica de uso habitual entre navegantes). Así, el *versine* (o *versed sine*) de un ángulo θ es $1 - cos\,(\theta)$. El *haversine* es la mitad del *versine*, o sea, $(1 - cos\,(\theta))\,/2$. Realizando el desarrollo matemático se puede demostrar que:

$$haversin\,(\theta) = (1 - \cos\,\theta))\,/2 = \sin^2\,(\theta/2) \tag{4.1}$$

El desarrollo de la *Fórmula Haversine* queda recogido en las ecuaciones 4.2 a 4.9. Se asume una Tierra esférica de radio R (tomándose como valor del radio medio $6367.449Km$, teniendo en cuenta que el radio ecuatorial es de $6378.137Km$ y el radio polar de $6356.7523Km$) y se parte de las localizaciones de dos puntos en coordenadas esféricas (es decir, latitud y longitud, trabajando en todo momento en radianes, nunca en grados).

$$R = 6367.449Km \tag{4.2}$$

$$lat\,(rad) = \frac{\pi}{180} \cdot lat\,(grad) \tag{4.3}$$

$$long\,(rad) = \frac{\pi}{180} \cdot long\,(grad) \tag{4.4}$$

$$\Delta lat\,(rad) = lat_2\,(rad) - lat_1\,(rad) \tag{4.5}$$

$$\Delta long\,(rad) = long_2\,(rad) - long_1\,(rad) \tag{4.6}$$

El resultado intermedio de la ecuación 4.7 es el cuadrado de la mitad de la distancia en línea recta entre los dos puntos.

$$a = \sin^2(\Delta lat/2) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2(\Delta long/2) \tag{4.7}$$

Por otro lado, la función trigonométrica *atan2* recibe dos parámetros y se trata de una variación de la función *arcotangente* en la que se persigue el cálculo del ángulo cuya tangente es el cociente de sus argumentos, es decir, devuelve el ángulo en radianes desde el origen de coordenadas hasta un punto cuyas coordenadas están determinadas por los propios parámetros pasados a la función. El resultado intermedio de la ecuación 4.8 ofrece el círculo de máxima distancia en radianes.

$$c = 2 \cdot atan2\left(\sqrt{a}, \sqrt{(1-a)}\right) \tag{4.8}$$

Finalmente, la distancia obtenida como resultado en la última ecuación (ecuación 4.9) vendrá dada en kilómetros.

$$d\,(Km) = R \cdot c \tag{4.9}$$

Tratándose el problema aquí planteado de algo ya recurrente en las aplicaciones y servicios que se desarrollan en la actualidad basados en el trabajo con el *GPS*, también se puede hacer uso de algunos métodos ya incluidos en la *API* de *Android*. Concretamente, se trata del método estático *distanceBetween(double startLatitude, double startLongitude, double endLatitude, double endLongitude, float[] results)* de la clase *Location*, que recibiendo dos pares de longitudes y latitudes (tipo de dato *double* de *Java*) y un *array* de tipo *float* en la que se almacenarán los resultados, computa la distancia aproximada en metros (y en línea recta, es decir, la ruta más corta posible entre los dos puntos geográficos) entre las dos localizaciones pasadas como parámetros. En cualquier caso, se hace constar que en la práctica, el uso de este método del *API* o el cálculo mediante la expresión matemática citada (*Fórmula Haversine*), ofrece diferencias en los resultados inapreciables, por lo que de hecho, resulta muy probable que sea esta misma expresión la implementada en dicho *API*.

Así, una vez que ese *Service* (en la aplicación implementada, se trata de la clase *ServiceDistanceController* que hereda de la clase *Service*) haya sido arrancado tras analizarse en la *Activity* de presentación del reto que se trata de una prueba de geolocalización, periódicamente se comprobará la distancia entre objetivo y equipo. Cuando el *Service* detecte que esa distancia es menor al radio máximo establecido, se deberá parar dicho servicio puesto que su ejecución ya no será necesaria hasta la siguiente prueba de geolocalización en caso de haberla. También se deberá enviar al servidor una petición de

servicio *HTTP* en la que se indique que el objetivo ha sido alcanzado y la distancia a la que se encontró el equipo en el momento de la notificación, con tal, todo ello, de poder pasar al siguiente reto, que normalmente consistirá en la realización de un determinado juego, fotografía, etc., en el lugar hasta el que se ha desplazado el equipo.

Pero una de las peculiaridades de *Android* es que sólo desde la actividad que está ejecutando en primer plano, se pueden lanzar otros elementos como por ejemplo un *AlertDialog* o una nueva *Activity* para que tomen el primer plano de la ejecución en la *GUI*. Dado que dentro de la aplicación para la *gymkhana*, el usuario podrá estar consultando cualquier pantalla o *Activity* en el momento en que el *Service* detecta que se ha alcanzado el objetivo (puede encontrarse en la propia pantalla de presentación del reto, en el mapa de *Google* realizando alguna consulta sobre el lugar o en alguna pantalla del servicio de mensajería interno), y puesto que desde el Service no se tendrá conocimiento de cuál de esas actividades está ejecutando en *foreground*, se realizará una notificación a todas ellas de manera simultánea. Esto es, que ante este tipo de pruebas, todas las *Activity* registrarán un *BroadcastReceiver*. El *ServiceDistanceController* por lo tanto, deberá notificar a todas las actividades que se alcanzó el lugar, cosa que hará mediante el envío de un mensaje en *broadcast*. Todas las actividades que hubieran registrado un *BroadcastReceiver* serán notificadas de un nuevo mensaje de este tipo, lo procesarán en los métodos propios de ese *BroadcastReceiver* que se ejecutarán automáticamente ante la recepción de dicho mensaje de *Android*, y para finalizar, se llevará el flujo de la aplicación hacia la ejecución de las tareas pertinentes para enviar la respuesta al servidor y pasar al siguiente reto.

Por último, sólo señalar la relevancia que tendrá para este tipo de retos, la pestaña de la aplicación en la que el equipo dispondrá de un mapa en el que consultar su posición, como ya se mostró en la figura 4.21. Para el uso de estos mapas en *Android* fue necesaria la obtención de una clave, al igual que el mapa empleado en la aplicación *web*, así como hacer uso del *Google Android Maps API*, *API* que en un principio estaba incluída dentro del *API* de *Android* pero que en la actualidad se puede encontrar independiente de dicha plataforma. Pero el equipo, además de consultar su posición en el mapa, podrá también consultar un marcador adicional que indicará el lugar al que se debe dirigir, siempre y cuando el organizador permitiera esto, pues en el momento de la creación del reto, también podrá optar por no mostrar ayuda adicional a los equipos.

Retos de Realidad Aumentada

Se debe señalar en primer lugar que este último tipo de reto, basado en el módulo de realidad aumentada [Román-López:2009] desarrollado dentro del proyecto *LibreGeoSocial*, no fue probado y puesto en marcha durante los experimentos realizados en el presente Proyecto Fin de Carrera dado que su implementación se llevó a cabo una vez finalizados los mismos. No obstante, como se explicará a continuación, la principal diferencia con un reto de respuesta textual es que el enunciado del propio reto deberá ser obtenido a través del módulo de realidad aumentada de *LibreGeoSocial*, enviando la respuesta textual del mismo modo que en el primer tipo de reto expuesto. Y dado que el correcto funcionamiento de dicho módulo de realidad aumentada está probado de manera exhaustiva por el grupo de desarrolladores de *LibreGeoSocial*, el funcionamiento adecuado de este tipo de pruebas queda garantizado con su puesta a punto en el emulador y en el teléfono *HTC Dream* (*G1*) empleado para el desarrollo del proyecto.

Como se puede entender de lo señalado con anterioridad, en este tipo de retos la dinámica para los organizadores de la *gymkhana* variará ligeramente. En las *gymkhanas* tradicionales es habitual realizar pruebas consistentes en que los grupos participantes busquen un objeto escondido por la organización en algún lugar geográfico (de hecho ésta es la base de la práctica del *geocaching*), estando este tipo de reto en consonancia con esto mismo. Es decir, en este caso, el organizador, antes del momento de la celebración de la *gymkhana* deberá desplazarse hasta el lugar en el que desea que se celebre y resuelva el reto. Una vez allí, deberá crear un nuevo recurso virtual a través del interfaz de *LibreGeoSocial* que supone el módulo de realidad aumentada disponible en el proyecto (véase la figura 4.22). Ese recurso podrá ser una nota de texto (por ejemplo una adivinanza, un acertijo, o simplemente el propio enunciado del reto, etc.), una fotografía o incluso un audio. Por lo demás, el *Manager*, a través del interfaz *web* para la creación de retos, indicará que la prueba es de respuesta textual y del subtipo de realidad aumentada, escribiendo el enunciado del reto en sí de la manera convencional.

En el momento de la participación en la *gymkhana*, se indicará al equipo que debe hacer uso del módulo de realidad aumentada de *LibreGeoSocial* (se podrá arrancar sin mayores complicaciones por medio del botón virtual introducido en la *GUI* de la aplicación, tal y como se puede observar en la figura 4.23) para encontrar un determinado recurso virtual en una zona determinada a la que posiblemente se les habrá llevado previamente a través de un reto de geolocalización. Una vez que el equipo haya localizado ese recurso, ya tendrá en su poder toda la información necesaria para saber el reto que deben resolver. Encontrada pues la solución, tan sólo deberán retirar el módulo de realidad aumentada e introducir la respuesta textual de la misma manera que se hace en el tipo de reto genérico de respuesta textual.

Figura 4.22: Etiquetado de un nuevo recurso virtual a través del interfaz de realidad aumentada de *LibreGeoSocial*

De este modo, se tiene que este tipo de reto no es más que un avance de las muchas posibilidades que puede ofrece un módulo de realidad aumentada, no sólo para la aplicación de *gymkhanas* sino para cualquier aplicación en general. Además, su uso en la aplicación para *gymkhanas* podrá tener un claro carácter divulgativo entre los participantes, dado que se trata de un avance tecnológico puntero y aún sin una clara penetración de uso y conocimiento entre el gran público.

Figura 4.23: Pantalla para la presentación de retos de respuesta textual con uso del módulo de realidad aumentada de *LibreGeoSocial*

4.1.4. Servicio de Mensajería Interno

Durante el curso del Proyecto Fin de Carrera, existe un aspecto cuya importancia siempre se ha tenido presente. Concretamente, se trata de la posibilidad de permitir a los equipos y los organizadores estar en permanente contacto. Teniendo en cuenta que se dispone de un teléfono móvil con acceso además a redes de datos, se podría pensar en el uso de las llamadas telefónicas, *SMS*, correo electrónico o incluso sistemas de mensajería tipo *chat* como pueda ser *Google Talk*. Sin embargo, estas opciones se consideraron como una incomodidad para los usuarios y organizadores, dado que se debería realizar un intercambio de números de teléfono y/o correos electrónicos, además de la tarificación por esas llamadas/*SMS* cuando se dispone, en condiciones normales, de una tarifa plana de datos que permitirá tener ese medio de comunicación sin necesidad de incrementar el coste ya sea de la organización o de los propios usuarios en caso de que participen con su propio teléfono *Android*. Así, se optó finalmente por la implementación de un servicio de mensajería propietario e interno del propio sistema para *gymkhanas*. Por medio de este servicio de mensajería se conseguirá que además de no contar con un incremento en la factura del teléfono, no será necesario el intercambio de direcciones de correo electrónico para emplear por ejemplo *Google Talk*. Y lo que resulta más importante aún, que el acceso al servicio de mensajería quedará integrado dentro de la aplicación *Android* para la *gymkhana* sin necesidad de abandonar la misma y hacer uso de cualquier otra aplicación del teléfono, así como el organizador de la *gymkhana* durante el transcurso de la misma, podrá visualizar y/o responder cómodamente a todos los mensajes desde su aplicación *web* para el control y monitorización, todo ello, sin necesitar en su caso ningún recurso adicional.

Como en el resto de las comunicaciones remotas llevadas a cabo en el sistema, la entrega de mensajes se hará posible empleando el protocolo de comunicaciones *HTTP*. Y por supuesto, estos mensajes estarán codificados según documentos *JSON* con duplas del tipo *<nombre_del_parámetro>*-*<valor>* para el caso de la aplicación *Android*, y para el interfaz *web* del *Manager*, en formato *HTML* (haciendo uso de formularios) para la publicación de mensajes, y también formato *JSON* para la lectura periódica de los mismos por medio de la tecnología *AJAX*. Por otro lado, tal y como se estudió en el capítulo 3, concretamente en el apartado dedicado al diseño del modelo de datos, todos los mensajes intercambiados serán almacenados en la base de datos del sistema. Esto quiere decir que el servicio de mensajería implementado tiene la estructura de un sistema centralizado que se encarga de almacenar esos mensajes, y de transmitirlos cuando un cliente se los solicita, frente a los servicios de mensajería en que los mensajes no son almacenados sino que se transmiten directamente entre los extremos de la comunicación.

Comenzando por la implementación llevada a cabo en la interfaz *web* del *Manager* de la *gymkhana*, en la figura 4.24 se puede apreciar una captura de pantalla en la que se muestra cómo el organizador podrá enviar mensajes. Además del campo de texto para escribir el cuerpo del mensaje que se desea enviar, se podrá elegir entre mandar el mensaje a un equipo en concreto seleccionable a partir de su nombre, o a todos los equipos participantes de la *gymkhana* junto con el propio organizador. Esto permitirá, dependiendo de la finalidad en cada momento, responder a una duda que un equipo pudiera presentar en un momento determinado, o dirigirse a todos los equipos de la *gymkhana* para hacer una aclaración que podría interesar a todos ellos.

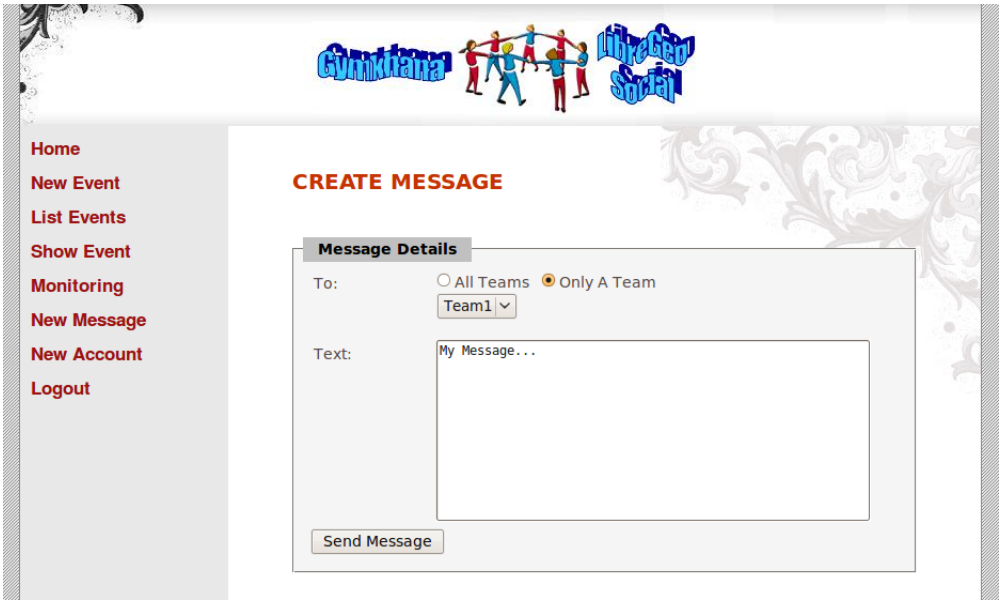


Figura 4.24: Detalle del interfaz *web* del organizador para el envío de mensajes

En cuanto al acceso del *Manager* a los mensajes a través de su aplicación *web*, será similar al ya presentado para las respuestas a retos. En la figura 4.25 se puede observar esta interfaz, donde el organizador podrá eliminar todos los mensajes intercambiados en el evento que está monitorizando, eliminar un mensaje en concreto, responderlo o simplemente consultarlo, además del propio texto, su remitente y su/s destinario/s. En cuanto a la recarga de los mensajes en la interfaz *web* a medida que estos vayan siendo enviados y respondidos, se conseguirá, como ya es habitual, por medio de la tecnología *AJAX* y el objeto *XMLHttpRequest*, enviando periódicamente una petición *HTTP* al servidor para solicitar todos los mensajes publicados en esa *gymkhana* y actualizando el campo de la página *web* destinado a la presentación de los mismos, realizando todas estas operaciones de forma transparente al usuario y mostrando una alerta con *JavaScript* solamente si se ha detectado la entrada de un nuevo mensaje.

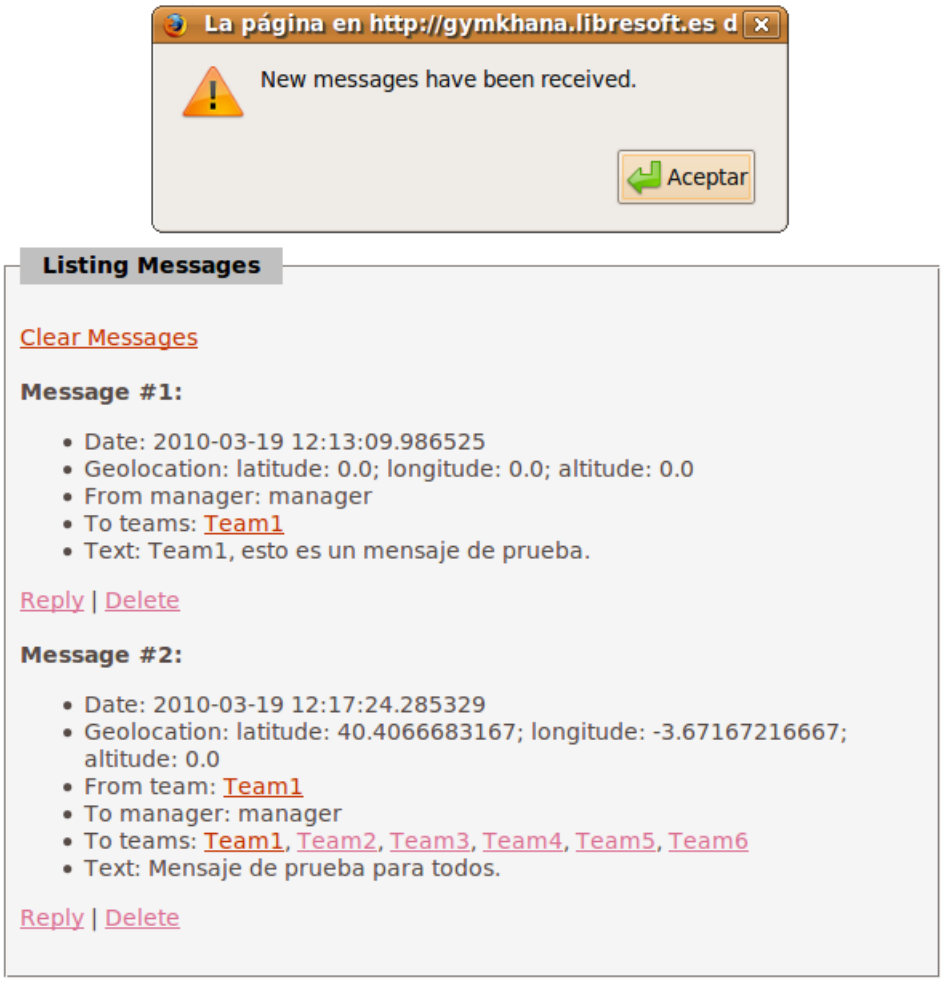


Figura 4.25: Detalle del interfaz *web* del organizador para la visualización de mensajes

Por otro lado, la implementación del servicio de mensajería en la aplicación *Android* se analizará a continuación, teniendo como punto de partida la figura 4.26. En esta captura de pantalla del emulador se puede contemplar la *GUI* con que el usuario interaccionará para hacer uso de este servicio. Cuando el usuario pulse sobre la tercera de las pestañas de la aplicación *Android*, encontrará tres opciones distintas:

- Redactar un nuevo mensaje.
- Consultar los mensajes recibidos.
- Consultar los mensajes enviados.

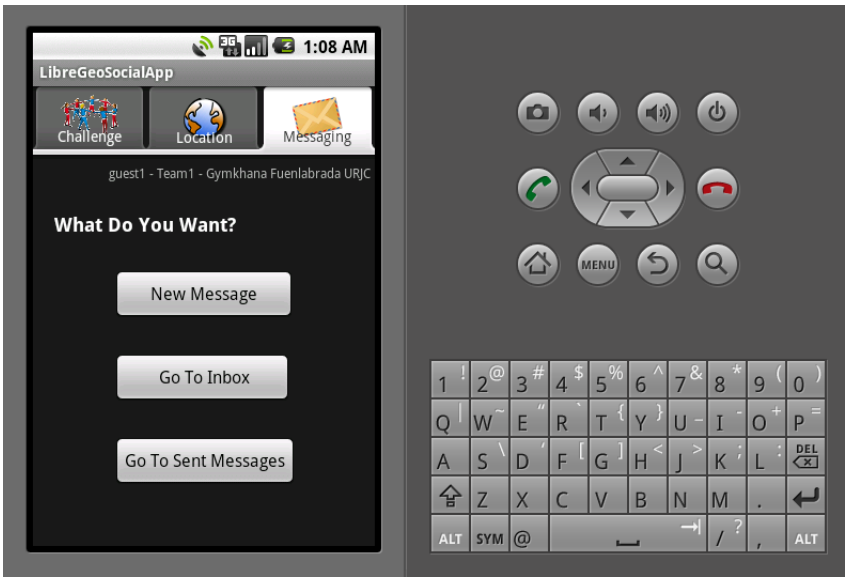


Figura 4.26: Pantalla principal del sistema de mensajería interno

La interfaz de usuario para la redacción de un nuevo mensaje es la mostrada en la figura 4.27. En ella, el usuario deberá seleccionar los destinatarios del mismo por medio de un *Spinner* del *API* de *Android*, redactar el cuerpo del mensaje y finalmente enviarlo pulsando sobre el botón virtual destinado a tal fin. Una vez que el usuario haya realizado estas acciones, será notificado mediante un mensaje en pantalla de si el envío se realizó correctamente o si por el contrario, se produjo algún problema por el cual su transmisión no se pudo completar.

Cuando el usuario quiera consultar los mensajes que ha recibido (o quizás también los que él mismo ha enviado hasta el momento) se pasará a una nueva pantalla en la que empleando la clase *ActivityList* de *Android*, se presentará una lista con un resumen informativo sobre los mensajes obtenidos desde el

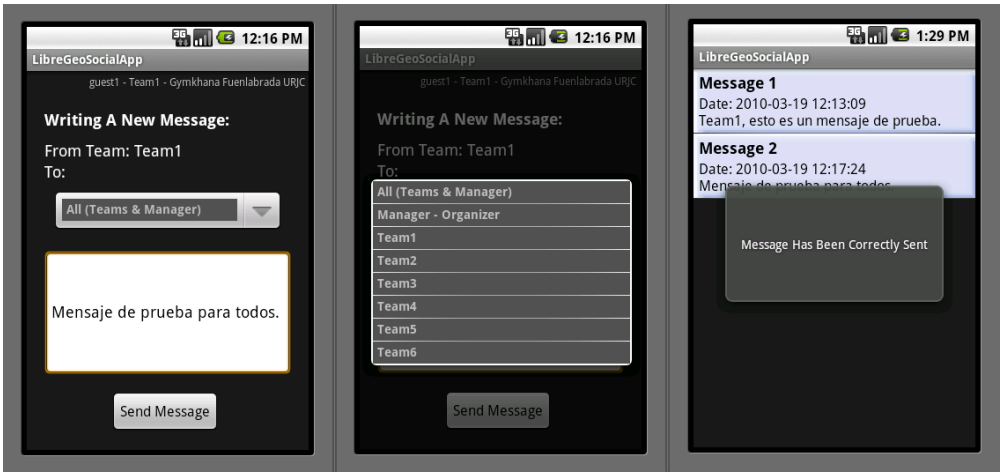


Figura 4.27: Pantalla para el envío de mensajes por parte de los equipos participantes

servidor. Cuando el usuario seleccione uno de los mensajes listados pulsando sobre el mismo en la pantalla táctil, se pasará a una nueva *Activity* que se corresponderá con la de visualización de mensajes. En ella, se podrá observar la fecha y hora del mensaje, el remitente y destinatarios así como el propio texto del mensaje. También se ofrece la posibilidad de responder al mensaje con tan sólo pulsar sobre el botón destinado a ello en la *GUI*, lo cual llevará automáticamente al usuario a una nueva pantalla en la que redactar el mensaje de respuesta (véase la figura 4.28).

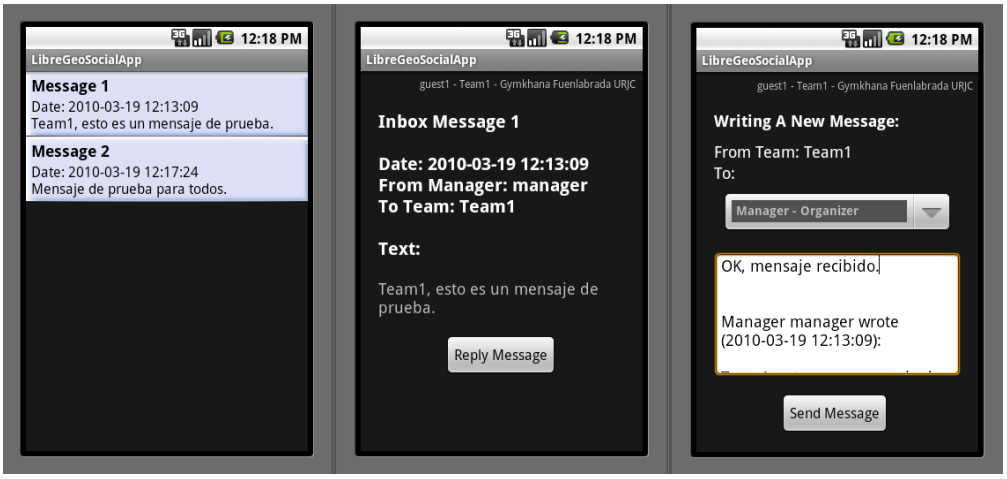


Figura 4.28: Pantallas para la visualización de la lista de mensajes recibidos / enviados y el detalle completo de cada mensaje

Respecto a la recepción de mensajes, existe una opción en el menú de esta pantalla de especial relevancia (véase la figura 4.29). Se trata de la opción de configuración (*"Settings"*), en la que se podrá activar/desactivar un chequeo automático y periódico en segundo plano[4]. Es decir, nuevamente un *Service* de *Android* ejecutará periódicamente una determinada tarea (enviar una petición *HTTP* al servidor *web* para procesar si se ha recibido algún mensaje nuevo desde la última consulta realizada) y llevando a cabo una serie de acciones consecuentes con el resultado obtenido.

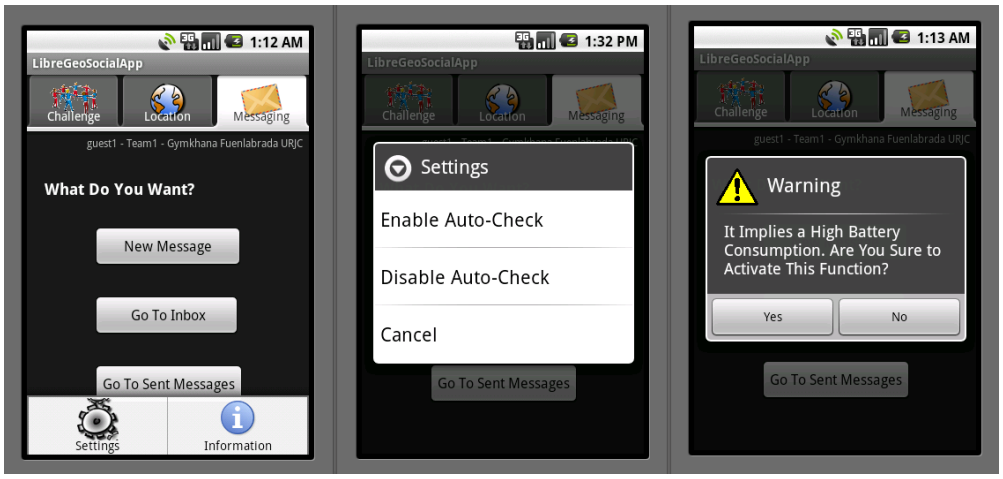


Figura 4.29: Activación del chequeo automático de recepción de nuevos mensajes

En caso de que el servicio en *background* detecte la recepción de nuevos mensajes, el usuario será notificado (clase *Notification* del *API* de *Android*) de ello tanto con un *Toast* como con una notificación en la barra de estado. El usuario podrá desplegar la barra de estado para visualizar la notificación en cualquier momento y acceder a los mensajes recibidos (figura 4.30).

Esta manera de chequear el estado de una determinada variable es la referida habitualmente como la técnica del *polling*. Y precisamente la implementación mediante este mecanismo presenta un inconveniente teniendo en cuenta el desarrollo para una plataforma destinada a teléfonos móviles cuyos recursos están limitados. El inconveniente se refiere al hecho de que establecer conexiones (normalmente *UMTS* en el caso del sistema implementado pues es la tecnología que permite conectividad total, cosa que no ocurriría con una conexión *Wi-Fi*) con el servidor continuamente para la transmisión de información, acaba suponiendo un consumo de batería considerable, aspecto

[4]Inicialmente esta opción se configuraba siempre como desactivada. Pero dadas las conclusiones obtenidas en los experimentos realizados, ahondándose en este aspecto durante su análisis en el capítulo 5, se consideró más oportunó activar este *auto-check* por defecto pese a que su uso implique un mayor consumo de batería en el móvil.

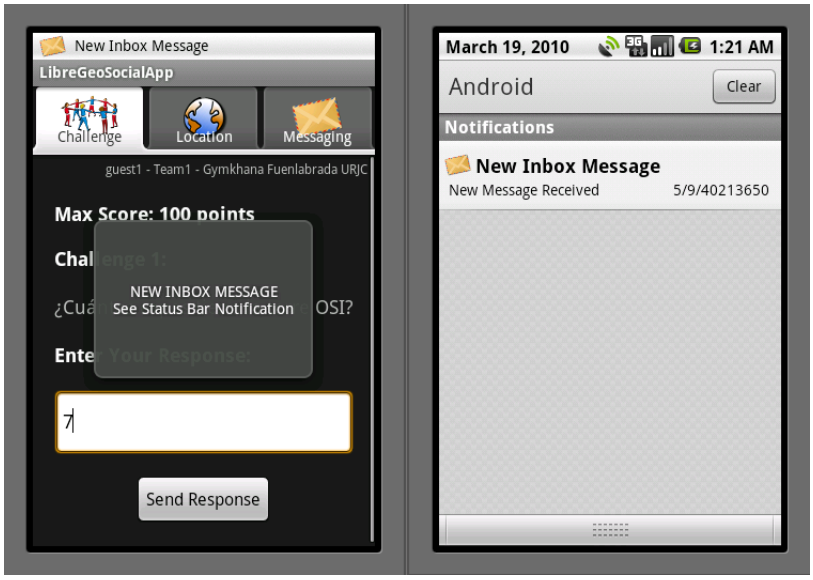


Figura 4.30: Notificacion de la recepción de nuevos mensajes en el sistema de mensajería interno

preocupante teniendo en cuenta que durante la *gymkhana* también se hará uso de la cámara, *GPS*, etc.

Así, la primera medida a tomar será disminuir la frecuencia con que el teléfono móvil realizará ese chequeo de nuevos mensajes. Por contra, se perderá inmediatez en el servicio de mensajería, por lo que se deberá encontrar un término medio entre el consumo de batería y la inmediatez en la entrega de mensajes. Finalmente, tal y como se pudo comprobar experimentalmente en las pruebas realizadas con los voluntarios (analizadas en el capítulo 5), se consideró que el chequeo se realizará con una periodicidad de alrededor de un minuto (por supuesto, de cara al futuro y dependiendo de las necesidades que puedan surgir, este período podría ser configurable por el propio usuario con tan sólo añadir un nuevo menú de configuración en la aplicación).

Debido a este fuerte inconveniente que supone el consumo de batería y pérdida de inmediatez en la entrega de los mensajes, se decidió buscar una alternativa de implementación. De hecho, la solución óptima para la resolución de este problema en términos de inmediatez sería la basada en el uso de notificaciones vía *PUSH* sobre el teléfono *Android*. Por medio de estas notificaciones *PUSH* (implementadas normalmente sobre una conexión *TCP* [RFC793, Tanenbaum:2003, Stallings:2004] de larga duración establecida entre cliente y servidor sobre la cual se emplean protocolos como *IMAP* o *XMPP* para realizar las notificaciones), el servidor inicia la comunicación con el cliente sin esperar a que este último solicite si hay alguna información nueva para él. Se trata por tanto de un mensaje que una aplicación servidora envía a una aplicación cliente indicando que tiene alguna información nueva

y disponible para que la acceda, lográndose con ello inmediatez en la comunicación. Pero a su vez, esta solución presenta un gran inconveniente: si bien se consigue inmediatez en la entrega de los mensajes, el consumo de batería será similar o incluso superior al que se tendría por medio del *polling*. Esto se debe a que cliente y servidor deberán conservar establecida de manera permanente una conexión *TCP*, para lo cual se deberán enviar periódicamente mensajes *Keep Alive* de *TCP* por dicha conexión con tal de que no sea cerrada por el *middleware* de comunicaciones/sistema operativo de cada extremo.

Por otro lado, el desarrollador *Android* podría servirse de las notificaciones *PUSH* de correos electrónicos entrantes sin introducir por tanto un consumo de batería adicional. La desventaja de esta alternativa reside en la necesidad de disponer de un sistema automático para el envío de correos electrónicos (o bien de *SMSs* si se quisiera, pues la notificación de su recepción igualmente podría ser interceptada en el teléfono), así como aumentar el grado de complejidad de la aplicación *Android*, puesto que habrá que programar un interceptor de esas notificaciones de manera que cuando se detecte en ellas una determinada cabecera, sean eliminadas con el consecuente aviso al usuario de una determinada información recibida y relativa a la aplicación. Y por supuesto, en caso de no tratarse de una notificación propia del sistema para *gymkhanas* sino de un correo electrónico (o *SMS*) ajeno al mismo, se respetará su integridad y se hará llegar igualmente al usuario para su consulta en cualquier momento.

No obstante, se buscó una alternativa a la solución finalmente adoptada y que mantuviera el espíritu de la política basada en el uso del *PUSH* de manera que contando con inmediatez en la comunicación, no se tuviera un aumento aparejado en el consumo de batería. Esta alternativa fue la posibilidad de abrir un *socket* en el cliente *Android* (clase *ServerSocket* del *API*) en un determinado puerto en el que se permanecería a la escucha. Por su parte, el servidor, cuando detectase la entrada de un nuevo mensaje destinado a un determinado equipo de la *gymkhana*, por medio del sencillo cliente *HTTP* que *Django* incorpora, se encargaría de realizar una petición de este tipo al teléfono móvil correspondiente a ese equipo. Mediante esta metodología, se evitaría el *polling* realizado en el cliente, por lo que además de un presumible ahorro de batería en el móvil (mejorando el tiempo de respuesta en el envío de los mensajes), también se rebajaría la carga de trabajo en el servidor *web*. Pero muy pronto se encontraron múltiples inconvenientes a esta solución alternativa, y que implicaron su descarte. Entre los inconvenientes más destacados se encuentran:

- El servidor deberá mantener un registro actualizado de las direcciones *IP* [RFC791] y puertos en que cada uno de los móviles con que se está participando en la *gymkhana*, mantiene abierto el *socket* de comunicaciones destinado al servicio de mensajería.

- Dependiendo de la política seguida por cada operador de telefonía a la hora de asignar direcciones *IP* a los teléfonos móviles para que realicen sus conexiones de datos, se podría contar con una dirección *IP* privada por lo que el establecimiento de una conexión *HTTP* desde el servidor *web* del sistema hasta el teléfono móvil *Android* sería de difícil realización (contando también con posibles *NAT*, *proxys*, etc.).

- Aunque el operador de telefonía asignase a los móviles direcciones *IP* públicas y por tanto accesibles desde el exterior, se debe tener también en cuenta la presencia de posibles *firewalls* en la red de comunicaciones del operador, por lo que las conexiones entrantes a un móvil podrían ser filtradas, o bien se podría estar filtrando por puertos, etc.

- Si bien en una red *UMTS* [Kaaranen:2006] el problema de la movilidad se resuelve a nivel radio, conservando el teléfono la misma dirección *IP* durante toda la conexión *3G*, independientemente de que se cambie de *Nodo B* (*BTS*, *Base Transceiver Station* en termilogía *GSM*) a través del cual se obtiene el acceso a la red, la dirección *IP* asignada al móvil será la misma durante toda la conexión. Esto provoca por un lado cierta simplificación, puesto que sería suficiente con que el teléfono *Android* notificase al servidor al inicio de la *gymkhana* en qué *IP* y puerto permanece a la escucha. Pero por otro lado, se necesitaría cierto aumento en la complejidad del sistema puesto que se deberían contemplar posibles desconexiones de la red (cosa totalmente probable, especialmente si el equipo, para superar una prueba necesitase realizar un desplazamiento en metro, en coche atravesando un túnel, etc.). Esto es, que tras una desconexión de la red *UMTS* y posterior reconexión, el móvil podría tener asignada una dirección *IP* distinta a la que recibió en un primer momento, por lo que sería necesario detectar continuamente estos posibles cambios así como notificarlos al servidor del sistema para que actualice los registros almacenados en su base de datos.

- El hecho de abrir un *socket* de comunicaciones en el terminal *Android* supone una posible vulnerabilidad para la seguridad de la aplicación y del teléfono que podría ser utilizada para poner en compromiso el correcto funcionamiento del móvil.

- En cuanto a aspectos teóricos de la solución adoptada, se ha señalado a lo largo de la memoria que el sistema implementado cumple con la arquitectura cliente-servidor (navegador/*Android* - *Django*). Sin embargo, esta arquitectura quedaría en cierto modo quebrada por esta solución, en la que cliente y servidor intercambiarían sus papeles para esta funcionalidad dado que el servidor *Django* ejercería a modo de cliente (iniciando una comunicación *HTTP*) del móvil *Android* que permanece a la escucha en un *ServerSocket*.

Capítulo 5

Despliegue y Resultados

No te lamentes de tus errores, aprende de ellos.

Bill Gates

En este capítulo se presenta el proceso de pruebas experimentales realizadas en un escenario real con el fin último de corroborar el correcto funcionamiento del sistema desarrollado en el Proyecto Fin de Carrera. También se analizarán los resultados obtenidos en cada experimento, en base a las respuestas a una encuesta de los voluntarios que acudieron a cada uno.

5.1. Introducción

Tras la implementación llevada a cabo del diseño del sistema especificado, el siguiente paso consiste en la realización de todo tipo de pruebas de caja blanca y caja negra para comprobar el correcto funcionamiento del servicio de comunicaciones en su integridad, desde las funcionalidades y módulos más superficiales, hasta los más importantes y vitales para el proyecto. Al mismo tiempo, un punto fundamental será la realización de pruebas por medio del emulador de teléfonos móviles *Android*, para finalmente probar la aplicación en un teléfono móvil real (concretamente un *HTC Dream* o *G1* de *T–Mobile/Movistar*, con el que se trabajará durante toda la fase de desarrollo y mediante conexión *Wi-Fi* al no contar con tarjeta de telefonía-datos).

Una vez comprobado *a priori* el correcto funcionamiento del sistema, llegó el momento de desplegarlo (instalando y configurando el servidor del servicio y la base de datos en una máquina virtual que contase con una conexión de red permanente y estable, dirección *IP* pública, etc.) para los posteriores experimentos a modo de pruebas en un escenario no controlado, lejos de

laboratorios y contando con varios terminales móviles *Android*, junto a un grupo de personas ajenas al propio proyecto y que serán los participantes en estas *gymkhanas* de prueba. Por lo tanto, se combinan dos factores: escenario real y usuarios que pueden realizar usos inesperados del terminal telefónico y del servicio.

La misión de estas personas, participantes en los experimentos de manera voluntaria y altruista, será la de ofrecer su visión a nivel del usuario final al que se dirige el servicio. Con tal de recopilar la mayor cantidad de información posible, proveniente de este grupo de personas y ofreciendo su percepción sobre la aplicación y la experiencia vivida, se procedió a la elaboración de una encuesta *on-line* por cada experimento, desarrolladas todas ellas gracias a un módulo de *Moodle* y disponibles en la página *web* del Departamento *GSYC*[1] de la *Universidad Rey Juan Carlos*. Adicionalmente, tras la *gymkhana* y dentro de un ambiente distendido, se grabó en vídeo a los voluntarios respondiendo a diversas preguntas que se les formularon sobre la experiencia.

Una vez expuesta la metodología seguida en la realización de las pruebas, es el momento de presentar los experimentos y analizar los resultados cosechados.

5.2. Primer Experimento

Los datos del experimento son los siguientes:

- **Objetivos primordiales del experimento:**
  - Comprobar el funcionamiento correcto, fiable y libre de errores de la aplicación *Android* y del sistema en general.
  - Comprobar la facilidad de uso de la interfaz gráfica de usuario para personas ajenas al proceso de diseño e implementación de la aplicación.
  - Recopilar la mayor cantidad de información posible que pueda resultar de utilidad para perfeccionar el sistema.
- **Fecha:** 11 de febrero de 2010.
- **Lugar de la citación:** Campus de Fuenlabrada de la *Universidad Rey Juan Carlos*. Laboratorios 004 y 005 del edificio Laboratorio II.
- **Hora de la citación:** 16:00.

[1] Sitio *web*: *http://gsyc.escet.urjc.es/moodle/*

- **Personas reunidas:** acuden a la cita un total de ocho voluntarios para la realización del experimento, y dos ingenieros para la organanización, control y monitorización de la *gymkhana*. Los voluntarios se distribuyen en tres equipos: dos equipos de tres miembros y un equipo de dos.
- **Perfil de los voluntarios:** estudiantes de ingenierías (dependientes de la *Escuela Técnica Superior de Ingeniería de Telecomunicación*).
- **Recursos materiales:** se cuenta con un total de tres terminales móviles *Android* (concretamente *HTC Magic* (*G2*) de *Vodafone*) con tarifa plana de datos vía *UMTS*.
- **Pruebas de la *gymkhana*:** se plantean un total de seis pruebas breves de dificultad baja en cuanto a su resolución, combinándose las posibilidades que ofrece la aplicación: pruebas textuales, pruebas fotográficas, pruebas de geolocalización, uso de mapas, envío de mensajes, etc.

El orden seguido en la sesión se presenta a continuación:

- **Hora 16:10 -** Tras la llegada de todos los citados, comienzo de una breve presentación del experimento realizado y justificación de su necesidad para el proyecto. Breve comentario sobre el manejo del terminal móvil entregado a cada grupo (pantalla táctil, botones, teclado virtual, etc.) y pautas a seguir en caso de error en la aplicación.
- **Hora 16:30 -** Reparto de teléfonos móviles y comienzo de la breve *gymkhana* propuesta (núcleo del experimento) por todo el recinto del Campus de Fuenlabrada de la *Universidad Rey Juan Carlos*. Al mismo tiempo que los equipos resuelven las pruebas, los organizadores controlan las acciones de los equipos (ubicación, respuestas a retos, mensajes) y contemplan la hipotética aparición de posibles errores en el registro del servidor.
- **Hora 17:10 -** Regreso de los equipos al laboratorio una vez completada su participación en la *gymkhana*. Recogida de telefónos móviles. Comienzo en ese mismo momento de la cumplimentación por parte de cada voluntario de la encuesta de *Moodle*.
- **Hora 17:30 -** Comienzo de una merienda a modo de gratificación a los voluntarios por su buena disposición para la participación en el experimento.
- **Hora 17:50 -** Comienzo de las entrevistas realizadas a los voluntarios al mismo tiempo que se realiza una grabación de vídeo.
- **Hora 18:20 -** Despedida y agradecimientos a los voluntarios. Fin de la sesión.

5.2.1. Resultados

Como se ha señalado, los resultados del experimento se analizarán en base a las respuestas dadas por los voluntarios a una encuesta realizada al final del experimento, consistente en un total de cuarenta preguntas y estructuradas en cinco bloques temáticos bien diferenciados según los cuales se realizará el posterior análisis.

Al mismo tiempo, se pensó en la formulación de las consultas de manera que se pudiera obtener la mayor cantidad de información útil posible. Esto es, estructurando las respuestas dentro de una gradación específica en cada caso, lo que permitirá un fácil procesado y análisis de resultados, frente a la posibilidad de dar libertad en todo momento al usuario para que introduzca un texto que muy probablemente sería intratable de una manera estadística sin antes realizar un tratamiento subjetivo de los datos, y en muchos casos, incluso sin obtener respuesta dado el mayor esfuerzo que requeriría al voluntario frente a la posibilidad de simplemente marcar una casilla (pero por supuesto, permitiendo que al final de la encuesta se indique cualquier tipo de matización a las respuestas dadas, comentarios, sugerencias, etc.). Por otro lado, dentro de esa gradación en las respuestas, se intentó que las posibilidades entre las que el usuario pudiera elegir fuese siempre una cantidad de opciones par (Malo-Regular-Bueno-Muy Bueno). Con esto se busca que el usuario siempre se decante hacia un lado u otro de la balanza (si ha quedado satisfecho o no), impidiendo que se acoja a respuestas neutras como podría ocurrir en el caso de presentar cinco posibles respuestas. De esta manera, el ingeniero puede mapear las respuestas como un "No me ha gustado nada", "Hay que mejorar mucho", "Aún se puede hacer mejor", "Me ha gustado mucho" respectivamente.

Los resultados de la encuesta realizada a los voluntarios a la finalización de la actividad, han sido recogidos en su integridad en el apéndice A, sección A.1 para el caso de este primer experimento. En este apartado, tan sólo se presentarán para su análisis los aspectos más destacados de la información recogida por medio de la encuesta realizada.

- **Manejo del terminal de comunicaciones.**

  Durante la explicación introductoria al experimento, no se quiso profundizar en el uso y posibilidades de la aplicación *Android* desarrollada, pero sí se ofreció un comentario general sobre el uso del terminal (botones, pantalla táctil, teclado virtual, etc.). De esta manera, se podrán identificar claramente si los problemas que los usuarios pudieran encontrar durante la prueba, fueron relativos a la propia aplicación o más bien en relación al uso de un teléfono móvil nuevo para el usuario. Así, consultando las respuestas obtenidas en las cinco primeras preguntas de la encuesta, parece más que necesaria la entrega de una serie de

pautas a modo de instrucciones de uso del teléfono. De hecho, pese a esa explicación sobre el uso de la pantalla táctil, uso del *scroll* en pantalla, retirada del teclado virtual, etc., algunos voluntarios declararon haber notado cierta incomodidad con el uso del teléfono y por factores ajenos a la propia aplicación. Según declaraciones de algunos de los voluntarios:

- Entrevistado 1: "*[...] El móvil, eso de que sea todo táctil no me llega a convencer. Si se pudiera utilizar algo con botones para bajarlo* (refiriéndose al *scroll*) *[...]*"

- Ingenieros: "*Tú móvil, el que usas habitualmente, ¿tiene muchos botones, no es un móvil táctil?*"

- Entrevistado 1: "*Mi móvil tiene muchos botones, sí.*"

Otra prueba de que los principales problemas encontrados están en relación con el propio manejo del terminal, es que la mayoría de los voluntarios no fueron conscientes de la recepción de los mensajes internos enviados por el *Manager* de la *gymkhana*. Para notificar la recepción de nuevos mensajes, se optó por notificar este evento por medio de la barra de estado del teléfono, práctica habitual en *Android* como mecanismo para notificar la recepción de *SMSs*, actualizaciones, etc. Sin embargo, dado el desconocimiento de este factor por parte de los voluntarios, así como la manera de desplegar las notificaciones desde la barra de estado del teléfono, hizo que tan sólo un equipo se pudiera comunicar de manera fluida con la organización de la *gymkhana*. Sobre este mismo asunto se profundizará más adelante.

- **Calidad estética de la aplicación *Android*.**

  En base a las respuestas dadas a las preguntas 6 a 13 ambas incluidas, se puede afirmar que la percepción mayoritaria de los usuarios es de una buena interfaz gráfica, siempre mejorable pero en principio con las características deseables en una *GUI* (claridad en la presentación de contenidos, facilidad de uso, intuitiva, original, etc.), motivo por el cual su mejora pasará a un segundo plano durante el proceso de perfeccionamiento, en favor de solucionar las incidencias producidas en otros aspectos de la aplicación y que se analizarán más adelante.

  - Ingenieros: "*La interfaz en el móvil, ¿te ha parecido difícil de utilizar?*"

  - Entrevistado 3: "*No porque en cuanto lo coges, te haces a ello, no había ningún problema.*"

- **Interacción de la aplicación con el usuario.**

  En cuanto a la interacción, valorándose principalmente la velocidad de ejecución de la aplicación ante las acciones del usuario, existen aspectos que han resultado de mayor relevancia:

  - Entrevistado 2: "*[...] Lo táctil un poco lento, un poco duro. Igual que el iPhone es muy sensible al tacto, en Android parece que cuesta más.*"

  - Ingenieros: "*Cuando dices que iba lento...*"

  - Entrevistado 2: "*¿La aplicación o el Android?*"

  - Ingenieros: "*La aplicación.*"

  - Entrevistado 2: "*La aplicación iba lenta. El botón de aceptar y de enviar de vez en cuando daba la sensación de que no funcionaba.*"

  Tras investigar lo comentado por el voluntario (hecho corroborado también por las respuestas generales a las preguntas 18, 19 y 20), se tuvo que esa supuesta lentitud hasta el punto de que pudiera asemejarse a un cuelgue de la aplicación, realmente se debía a cada momento en que se enviaba una petición *HTTP* al servidor (envío de respuestas, obtención de un nuevo reto, etc.). Este problema es por tanto ajeno a la pantalla táctil del teléfono y al propio sistema operativo. La solución adoptada fue la introducción de barras de progreso (*ProgressDialog* del *API* de *Android*) mostradas en primer plano en el hilo [Oaks:1999] principal de la aplicación, arrancándose un nuevo hilo de ejecución que realiza el envío de las peticiones *HTTP* subyacentes, solucionándose de esta manera el problema aquí encontrado.

  Otro problema detectado tiene que ver con la notificación de la recepción de mensajes internos de la *gymkhana*, teniendo la pregunta 24 relación con esto así como los siguientes comentarios de los voluntarios entrevistados:

  - Entrevistado 2: "*[...] Sí que habría que poner alguna barra de progreso o soniditos, por ejemplo con el envío de los mensajes, poner algún parpadeo porque no nos hemos enterado de la recepción de mensajes.*"

  - Ingenieros: "*¿Qué cosas mejorarías?*"

  - Entrevistado 3: "*La parte de los mensajes, que no sabíamos cómo abrirlo, y poner un aviso sonoro.*"

  En un principio, durante la fase de desarrollo, se contempló la posibilidad de introducir una notificación adicional con cada nuevo mensaje recibido, ya fuera una vibración, un parpadeo del *LED* del teléfono o un sonido. Sin embargo, esta idea se desechó dado el mayor consumo

de batería que supondría. No obstante, ante los problemas detectados, la solución final pasó por conservar la notificación de nuevos mensajes por medio de la barra de estado, así como añadir una breve vibración al mismo tiempo que se despliega en pantalla un mensaje a modo de aviso y que desaparecerá tras unos instantes.

Por otro lado, la pregunta de la encuesta número 16, hace notar que los retos de geolocalización no contaron con el funcionamiento deseado. Tan sólo un equipo pudo superar el único reto de geolocalización que se planteó y además tras un tiempo prolongado de estancia en el lugar objetivo. Por ello, éste será uno de los principales puntos sobre los cuales trabajar, no sólo optimizando la programación de esta funcionalidad, sino también variando la configuración de consulta de la posición mediante el *GPS* (la configuración por defecto era de dos minutos, demasiado tiempo de espera para un equipo que desea superar un reto de este tipo, por lo que ese período de ejecución deberá disminuirse en próximos experimentos).

Por último, se debe destacar la más que aceptable fiabilidad de la aplicación para tratarse de la primera prueba realizada en un escenario real. Tan sólo un equipo indicó la aparición durante la *gymkhana* de una ventana lanzada por el sistema operativo indicando que la aplicación estaba tardando en responder. Esta misma incidencia, como se estudiará en el apartado dedicado al segundo experimento, se repitió entonces de manera mucho más frecuente, encontrándose en ese momento el verdadero foco del problema y por tanto subsanándolo.

- **Originalidad del proyecto e información general sobre diversos aspectos tecnológicos.**

  Desde la pregunta 28 y hasta la 33, sin tratarse de preguntas estrictamente relativas al presente Proyecto Fin de Carrera, resulta llamativo el casi absoluto desconocimiento de los voluntarios, todos ellos estudiantes de ingenierías, de aspectos tecnológicos presentados en el experimento (*Android* y computación ubicua), por lo que incluso se puede atribuir a estas sesiones de pruebas un carácter divulgativo entre los alumnos de la *Escuela Técnica Superior de Ingeniería de Telecomunicación*.

- **Valoración de la experiencia como participante en la *gymkhana*.**

  En cuanto a la experiencia del usuario como mero participante, se obtiene una valoración muy positiva, tanto en los resultados de las últimas preguntas de la encuesta (especialmente en la pregunta 39, donde la totalidad de los voluntarios, con mayor o menor grado de incertidumbre, afirman estar dispuestos a participar en próximos experimentos), como en los comentarios realizados durante las entrevistas finales:

- Ingenieros: "*¿Qué te ha parecido la gymkhana en general?*"

- Entrevistado 1: "*Bien, me ha parecido bastante fácil en cuanto a las pruebas*".

- Entrevistado 2: "*Pues me ha parecido muy bien, muy interesante. Un poco rápida pero está bien tener por primera vez el Android entre las manos.*"

- Entrevistado 3: "*Me ha parecido bastante bien. Me ha parecido que ha sido sencillo de usar y hubiera sido mejor que la gymkhana hubiera tenido muchas más preguntas.*"

- Entrevistado 4: "*Me ha parecido bien, bastante elaborado, los menús bastante intuitivos, y es un poco corta, pero para probar está bien.*"

Como se puede entender, valorándose positivamente la experiencia, destaca que los voluntarios señalen la facilidad de resolución de los retos planteados así como la corta duración de la prueba experimental, lo que indica que en efecto les hubiera gustado disfrutar del servicio durante mucho más tiempo.

5.3. Segundo Experimento

Los datos del experimento son los siguientes:

- **Objetivos primordiales del experimento:**
  - Comprobar nuevamente el funcionamiento correcto, fiable y libre de errores de la aplicación *Android* y del sistema en general, pero en esta ocasión con un mayor número de equipos participantes (teléfonos móviles) y miembros de cada equipo.
  - Comprobar nuevamente la facilidad de uso de la interfaz gráfica de usuario para personas ajenas al proceso de diseño e implementación de la aplicación, especialmente tras las mejoras introducidas en respuesta a los problemas surgidos en este respecto durante el primer experimento.
  - Recopilar nuevamente la mayor cantidad de información posible que pueda resultar de utilidad para perfeccionar el sistema.
- **Fecha:** 25 de marzo de 2010.
- **Lugar de la citación:** Campus de Fuenlabrada de la *Universidad Rey Juan Carlos*. Laboratorios 004 y 005 del edificio Laboratorio II.
- **Hora de la citación:** 17:00.

- **Personas reunidas:** acuden a la cita un total de veintidós voluntarios para la realización del experimento, y dos ingenieros para la organanización, control y monitorización de la *gymkhana*. Los voluntarios se distribuyen en seis equipos: cuatro equipos de cuatro miembros y dos equipos de tres.

- **Perfil de los voluntarios:** estudiantes de ingenierías dependientes de la *Escuela Técnica Superior de Ingeniería de Telecomunicación* y estudiantes de ingenierías en informática.

- **Recursos materiales:** se cuenta con un total de seis terminales móviles *Android* (concretamente *HTC Magic* (*G2*) de *Vodafone*). Uno de los teléfonos cuenta con tarifa plana de datos y los cinco restantes accederán a la red *UMTS* con tarjetas *SIM* de prepago de *Vodafone* con un saldo de 15 euros cada una.

- **Pruebas de la *gymkhana*:** se plantean un total de dieciséis pruebas de dificultad media en cuanto a su resolución, combinándose las posibilidades que ofrece la aplicación: pruebas textuales, pruebas fotográficas, pruebas de geolocalización, uso de mapas, envío de mensajes, etc.

El orden de la sesión se presenta a continuación:

- **Hora 17:15 -** Tras la llegada de todos los citados, comienzo de una breve presentación del experimento realizado y justificación de su necesidad para el proyecto. Indicaciones sobre el manejo del terminal móvil entregado a cada grupo (pantalla táctil, botones, teclado virtual, etc.), explicación de las diversas posibilidades que ofrece la aplicación y pautas a seguir en caso de error en la misma.

- **Hora 17:35 -** Reparto de teléfonos móviles y recogida de *DNIs*. Resolución de problemas de configuración de la aplicación (*login* en *LibreGeoSocial*, arranque de la aplicación para *gymkhanas*, selección de *gymkhana* y selección de equipo).

- **Hora 17:50 -** Comienzo de la *gymkhana* propuesta (núcleo del experimento) por todo el recinto del Campus de Fuenlabrada de la Universidad Rey Juan Carlos. Al mismo tiempo que los equipos resuelven las pruebas, los organizadores controlan las acciones de los equipos (ubicación, respuestas a retos, mensajes) y contemplan la hipotética aparición de posibles errores en el registro del servidor.

- **Hora 19:20 -** Aviso a los equipos de la finalización de la *gymkhana* mediante el servicio de mensajería interno del sistema. Regreso de los equipos al laboratorio. Recogida de telefónos móviles y devolución de

DNIs. Comienzo en ese mismo momento de la cumplimentación por parte de cada voluntario de la encuesta de *Moodle*.

- **Hora 19:40 -** Comienzo de una merienda a modo de gratificación a los voluntarios por su buena disposición para la participación en el experimento y charla distendida con todos ellos sobre el experimento para intercambio de impresiones.

- **Hora 20:20 -** Despedida y agradecimientos a los voluntarios. Fin de la sesión.

5.3.1. Resultados

La encuesta planteada en esta ocasión (recogida junto a los resultados en el apéndice A, sección A.2) fue mucho más breve (quince preguntas más tres adicionales a responder únicamente por los voluntarios asistentes también en el primer experimento) y con preguntas mucho más genéricas que las del primer experimento puesto que en aquél se atesoró una gran cantidad de información útil de cara a futuras pruebas. Las preguntas formuladas fueron mucho más generales y centrándose fundamentalmente en dos aspectos: uso de la aplicación por un usuario (y posibles problemas) y valoración general de la experiencia. Por otro lado, dado que en esta ocasión se esperaba una mayor población muestral a la hora de obtener resultados estadísticos a partir de la encuesta, se amplió el rango de posibles respuestas en aquellas preguntas a responder de manera predefinida, es decir, pasándose a valorar los factores expuestos de 0 a 10.

Tras este experimento, se obtuvieron fundamentalmente dos conclusiones. La primera de ellas tiene que ver con el servicio de mensajería del sistema para *gymkhanas*. En esta ocasión, no se indicó explícitamente a los equipos participantes que activasen al inicio de la *gymkhana* la opción del *auto-check* para comprobar periódicamente la recepción de nuevos mensajes. Esto provocó que la gran mayoría de equipos no tuviera comunicación con el *Manager* dado que tampoco comprobaban manualmente la recepción de mensajes nuevos. Por esto mismo se decidió que en adelante, la configuración de la aplicación activase por defecto ese chequeo automático y periódico de manera transparente al usuario, dada la importancia con que cuenta la posibilidad de que organizadores y participantes mantengan contacto permanente, y pese a que la intención original era que los propios usuarios se encargasen de manipular adecuadamente la activación y desactivación de esa opción, así como de comprobar manualmente el buzón de entrada de mensajes todo ello con tal de no incrementar considerablemente el consumo de batería. Pese a todo, tal y como se ha podido comprobar experimentalmente, la activación de este *auto-check* no supone un incremento preocupante en el consumo de batería,

más teniendo en cuenta que la aplicación *Android* hace uso de periféricos con un consumo mucho mayor, como es el caso de la cámara y del *GPS*. Por otro lado, cabe destacar que las modificaciones realizadas en la notificación de los mensajes entrantes ha sido considerada de manera positiva por parte de aquellos voluntarios que ya participaron en el primer experimento.

Pero el segundo aspecto destacado es que la prueba estuvo marcada por la continua aparición de cuelgues, lentitud y cierres inesperados de la aplicación. Obviamente, esto supone un incidente inaceptable, más teniendo en cuenta el tipo de servicio de telecomunicaciones que se está ofreciendo, donde el dinamismo y la rapidez son requisitos fundamentales. Tal y como los voluntarios explicaron y también respondieron en la encuesta, estos cuelgues y cierres se producían coincidiendo con la entrada en ejecución y consulta del *GPS*, y especialmente cada vez que el equipo salía de un edificio tras resolver en su interior un reto. Trabajando en la búsqueda del origen de estos comportamientos indeseables, la primera diferencia con respecto al primer experimento residía en que la consulta del *GPS* había sido configurada cada 15 segundos o bien cada 5 metros de movimiento, frente a los 2 minutos / 20 metros de aquella primera prueba. Si bien en pruebas preliminares el ingeniero no detectó ningún problema al respecto, ni en el emulador ni con el móvil de desarrollo, en los días posteriores al experimento se persiguió la reproducción de esos mismos errores y bajo las mismas hipótesis de trabajo (lugar, edificios, retos, configuración del móvil, etc.). Tal y como se pudo comprobar experimentalmente, el uso del *GPS* a la frecuencia establecida de 15 segundos / 5 metros, durante períodos de tiempo prolongados, originaba el problema indicado dentro de *LibreGeoSocial*.

El siguiente paso consistió en estudiar la implementación del *LocationListener* llevada a cabo en *LibreGeoSocial*, servicio utilizado en la aplicación para *gymkhanas*. Finalmente se encontró que el hecho de enviar con cada consulta del *GPS* una petición *HTTP* para actualizar en el servidor de la red social la geolocalización del usuario, unido a operaciones posteriores destinadas únicamente a *LibreGeoSocial* y sin utilidad dentro de la *gymkhana*, suponían una ralentización excesiva del sistema, hasta el punto de acumularse invocaciones al método *onLocationChanged(Location location)* del *LocationListener* debidas siempre a la consulta del *GPS*. Por esto mismo, se decidió implementar un nuevo *LocationListener* mucho más liviano y de uso exclusivo únicamente en la aplicación para *gymkhanas*, de manera que el servicio implementado en *LibreGeoSocial* fuese parado en el inicio, y rearrancado al salir de la aplicación de *gymkhanas*. Pese a todo, en el nuevo *LocationListener* el envío hacia el servidor de la posición obtenida por el *GPS* para que el *Manager* monitorice la ubicación de los equipos en cada momento, seguía conllevando cierta ralentización en el resto de la aplicación, por lo que se decidió realizar el envío de esa petición *HTTP* en un nuevo hilo de ejecución de manera que el procesamiento del *onLocationChanged(Location location)*

finalizase en cada invocación lo antes posible. Mediante esta nueva forma de actuar, los cuelgues y cierres de la aplicación son totalmente eliminados, con independencia de la configuración del *GPS*, habiendo sido comprobado su correcto funcionamiento incluso cuando se ordena una consulta continua de las coordenadas geográficas devueltas por el sistema de posicionamiento.

Por último, se debe obtener una última conclusión tras el análisis de las respuestas a la encuesta. Esta conclusión es que pese a las imperfecciones aparecidas en esta prueba, el servicio de *gymkhanas* presenta un gran atractivo para los usuarios, como bien demuestra que pese a esos problemas, en la pregunta número 10, un 81.82 % de los encuestados (18 de los 22 totales) haya afirmado en mayor o menor grado de rotundidad, estar en disposición de repetir la experiencia. Por su parte, en la pregunta número 11, un 59.09 % de los encuestados (13 de los 22 totales) reconoce haber alcanzado un grado de diversión igual o mayor a 7 sobre una escala de 0 a 10. Estos datos demuestran sin duda que una vez depurados los problemas surgidos dentro de las pruebas realizadas, y preparando *gymkhanas* con contenidos amenos, interesantes y enriquecedores, se consiga que el presente Proyecto Fin de Carrera alcance con éxito todos los objetivos marcados en un principio.

5.4. Tercer Experimento

Los datos del experimento, realizado a modo de demostración dentro de las actividades organizadas en la festividad del *Patrón de la Escuela Técnica Superior de Ingeniería de Telecomunicación* de la *Universidad Rey Juan Carlos* y una vez solucionados todos los problemas encontrados en los experimentos anteriores, son los siguientes:

- **Objetivos primordiales del experimento:**
  - Realizar una demostración exitosa ante profesores y alumnos de la *Escuela Técnica Superior de Ingeniería de Telecomunicación* en el día de la festividad de su Patrón.
  - Comprobar nuevamente el funcionamiento correcto, fiable y libre de errores de la aplicación *Android* y del sistema en general, con especial énfasis en los problemas presentados en los experimentos anteriores.
  - Comprobar nuevamente la facilidad de uso de la interfaz gráfica de usuario para personas ajenas al proceso de diseño e implementación de la aplicación, especialmente tras las mejoras introducidas en respuesta a los problemas surgidos en este respecto durante el primer y segundo experimento.

  - Recopilar nuevamente la mayor cantidad de información posible que pueda resultar de utilidad para perfeccionar el sistema.

- **Fecha:** 16 de abril de 2010, coincidiendo con la festividad del *Patrón de la Escuela Técnica Superior de Ingeniería de Telecomunicación.*
- **Lugar de la citación:** Campus de Fuenlabrada de la *Universidad Rey Juan Carlos.* Laboratorios 004 y 005 del edificio Laboratorio II.
- **Hora de la citación:** 10:15.
- **Personas reunidas:** acuden a la cita un total de dieciocho voluntarios para la realización del experimento, además de cinco profesores de la *Escuela Técnica Superior de Ingeniería de Telecomunicación* de la *Universidad Rey Juan Carlos* interesados en conocer de primera mano el proyecto aquí desarrollado, así como dos ingenieros para la organización, control y monitorización de la *gymkhana.* Los voluntarios se distribuyen en cinco equipos: un equipo de cinco miembros, dos equipos de cuatro, un equipo de tres y un equipo de dos.
- **Perfil de los voluntarios:** estudiantes de ingenierías dependientes de la *Escuela Técnica Superior de Ingeniería de Telecomunicación.*
- **Recursos materiales:** se cuenta con un total de cinco terminales móviles *Android* (concretamente cuatro *HTC Magic* (*G2*) de *Vodafone* y un *HTC Dream* (*G1*)). Los cinco teléfonos móviles accederán a la red *UMTS* con tarjetas *SIM* de prepago de *Vodafone* con un saldo de 12 euros cada una.
- **Pruebas de la *gymkhana*:** se plantean un total de veintiuna pruebas de dificultad media en cuanto a su resolución, combinándose las posibilidades que ofrece la aplicación: pruebas textuales, pruebas fotográficas, pruebas de geolocalización, uso de mapas, envío de mensajes, etc.

El orden de la sesión se presenta a continuación:

- **Hora 10:15 -** Tras la llegada de todos los citados, reparto de teléfonos móviles y tarjetas a modo de instrucciones de uso de la aplicación y del terminal *Android* (pantalla táctil, botones, teclado virtual, etc.).
- **Hora 10:20 -** Comienzo de la *gymkhana* propuesta (núcleo del experimento) por todo el recinto del Campus de Fuenlabrada de la *Universidad Rey Juan Carlos.* Al mismo tiempo que los equipos resuelven las pruebas, los organizadores controlan las acciones de los equipos (ubicación, respuestas a retos, mensajes), contemplan la hipotética aparición de posibles errores en el registro del servidor y documentan gráficamente por medio de fotografías y vídeos la actividad organizada.

- **Hora 11:45 -** Aviso a los equipos de la finalización de la *gymkhana* mediante el servicio de mensajería interno del sistema. Regreso de los equipos al laboratorio. Recogida de telefónos móviles. Comienzo en ese mismo momento de la cumplimentación por parte de cada voluntario de la encuesta de *Moodle*.

- **Hora 12:00 -** Consulta y comunicación de la clasificación de cada equipo en la *gymkhana*. Despedida y agradecimientos a los voluntarios. Fin de la sesión.

5.4.1. Resultados

La encuesta planteada en esta ocasión (recogida junto a los resultados en el apéndice A, sección A.3) estuvo compuesta por un total de diecisiete preguntas más una adicional a responder únicamente por los voluntarios asistentes también en el primer experimento y/o segundo experimento. La mayor parte de las preguntas ya fueron presentadas en la encuesta del segundo experimento, buscando con ello la obtención de un resultado objetivo con el que poder realizar una comparación directa y evaluar el grado de mejora del sistema de *gymkhanas* respecto a ese segundo experimento. Es decir, las preguntas se centraron igualmente en el uso de la aplicación por un usuario (y posibles problemas) y la valoración general de la experiencia, así como la evaluación de las mejoras introducidas respecto a los experimentos anteriores. E igualmente, dado que en esta ocasión se esperaba una población muestral a la hora de obtener resultados estadísticos a partir de la encuesta, similar a la del segundo experimento, se mantuvo el rango de posibles respuestas en aquellas preguntas a responder de manera predefinida, es decir, valorándose igualmente los factores expuestos de 0 a 10.

Analizando los resultados obtenidos en la encuesta[2], se tiene que este último experimento a modo de demostración fue todo un éxito. En las preguntas génericas 1 a 7 sobre la valoración general del sistema, facilidad de uso y *GUI* de la aplicación *Android*, etc., se obtuvieron resultados muy positivos y satisfactorios. De hecho, en la mayoría de las preguntas se puede atisbar la forma de una función de distribución gaussiana, centrada en la mayoría de los casos en una puntuación de 8-9 puntos, obteniéndose incluso calificaciones de 10 puntos.

En cuanto a la experiencia del usuario, la demostración fue acogida con un gran éxito, como bien se demuestra en las respuestas a las preguntas 9 (valorando el 93.75 % de los encuestados el uso de nuevas tecnologías para la realización de *gymkhanas* como bueno o muy bueno), 10 (dos de los en-

[2]En adelante, téngase en cuenta que de los dieciocho voluntarios asistentes al experimento, respondieron a la encuesta un total de dieciséis

cuestados valoran la experiencia a nivel de usuario en la *gymkhana* con un 7; tres de ellos con un 8; ocho de ellos con un 9; tres de ellos con un 10; es decir, el 100 % de los encuestados valoran la experiencia con una nota igual o superior a 7, y el 68.75 % con un 9-10), 11 (el 50 % de los encuestados afirma sin ningún tipo de duda que repetiría la experiencia con un sistema de *gymkhanas* como el presentado aquí, y el 50 % restante afirma que probablemente repetiría también la experiencia, no habiendo ningún encuestado que responda con una negativa), 12 (siete de los dieciséis encuestados califica su grado de diversión con un 10, lo que supone el 43.75 %, mientras que cinco encuestados más otorgan a su diversión un 9 y otros dos encuestados con un 8, contando con una única respuesta de 7 y otra respuesta más con un 6). De hecho, ya supone un éxito el número de participantes en los experimentos que han repetido experiencia a lo largo de las tres pruebas realizadas, como bien se muestra en la pregunta 17, donde se tiene que un 62.50 % de los asistentes (diez voluntarios de los dieciséis totales en la demostración) ya habían estado presentes en el primer experimento y/o en el segundo.

Otro indicativo más del grado de madurez del sistema de *gymkhanas* aquí presentado, y más concretamente de la aplicación *Android* empleada por los participantes, se encuentra en las respuestas ofrecidas a la pregunta número 15, donde los encuestados no destacan ningún punto débil, o bien, se centran en aspectos ajenos a la propia aplicación y más relativos al *hardware* del teléfono (los problemas comentados relativos a la pulsación de las teclas en el teclado virtual no guardan relación con la aplicación sino con la pantalla táctil y su tamaño, la detección de las pulsaciones desde el sistema operativo, etc.). E igualmente, dos de los voluntarios que respondieron a la última pregunta, califican las mejoras introducidas en la fiabilidad de la aplicación, ejecución del *GPS*, etc., con un 10; cinco de ellos lo valoran con un 9; dos más con un 8; uno con un 7, y otro más con un 6. Es decir, que el 63.63 % de los voluntarios que respondieron a esta última pregunta, calificaron las mejoras introducidas con un sobresaliente (9-10).

No obstante, aún existen dos aspectos destacados por los usuarios y que pueden ser mejorados fácilmente. Uno de ellos es el hecho de que a menudo, los usuarios, tras pulsar varias veces consecutivas el botón *hardware* del terminal destinado a ir hacia una pantalla anterior, finalizaban la ejecución de la aplicación para la *gymkhana*, teniendo que arrancarla de nuevo. Este aspecto se ha solucionado de manera que cada vez que se produzca esta circunstancia, se muestre al usuario un *AlertDialog* del *API* de *Android* en el que se inste a dicho usuario a indicar si de verdad está interesado en abandonar la aplicación o si simplemente se trata de un error en el manejo de la misma. El otro aspecto reside en que el equipo que participó en el experimento con el teléfono *HTC Dream* (*G1*), al intentar introducir las respuestas textuales a través del teclado *QWERTY* físico, tras girar el móvil éste no permitía el cambio de la pantalla de la posición vertical a la horizontal, produciéndose

con ello un cuelgue de la aplicación. Esto se debe a que en la declaración de las *Activity* de la aplicación en el *AndroidManifest.xml*, tan sólo se permitió su uso con la pantalla en vertical (*portrait*). El problema se resuelve con tan sólo retirar esa prohibición de manera que la aplicación se pueda emplear en cualquier orientación del móvil (horizontal o vertical).

Igualmente se intentó evaluar cuál sería el número ideal de participantes en la *gymkhana* por cada uno de los equipos posibles. En base a las respuestas ofrecidas por los voluntarios en la pregunta número 13, se puede decir que ese número ideal de integrantes de cada equipo es de 4-5 personas. Un menor número de integrantes podría llevar a una mayor dificultad para la superación de ciertas pruebas, así como posiblemente un menor grado de diversión. Mientras que con grupos mucho más poblados, el contacto con el terminal móvil sería mucho menor por lo que algún miembro del equipo podría llegar a sentirse desplazado e incómodo.

Por último, otro aspecto destacado es que tras activar por defecto el *auto-check* de nuevos mensajes recibidos de modo que el usuario de la aplicación no se tuviera que preocupar por ello, hizo posible que en esta ocasión todos los equipos y el *Manager* de la *gymkhana* pudieran mantener una comunicación fluida, siendo el servicio de mensajería utilizado en algunos casos para comentarios entre los equipos que hicieron la experiencia mucho más divertida. Y al mismo tiempo, las pruebas de geolocalización así como los problemas de fiabilidad de la aplicación fueron finalmente solventados gracias a las medidas expuestas en el experimento número dos en base a los resultados obtenidos en el mismo. De hecho, catorce de los dieciséis encuestados valoraron el funcionamiento de la aplicación ante las pruebas de geolocalización con una nota igual o superior a 8 (el 87.50 % de los encuestados, y calificándolo con un 10 el 31.25 % de los encuestados, es decir, cinco de ellos).

Así pues, tras los resultados cosechados en este último experimento a modo de demostración, y teniendo en cuenta el alto grado de satisfacción de todos los asistentes, así como la diversión de todos ellos durante la *gymkhana* móvil *Android*, se puede decir que el presente Proyecto Fin de Carrera queda desplegado, operativo y suficientemente maduro como para poderse emplear en cualquier otro escenario.

Capítulo 6

Conclusiones y Futuras Líneas de Trabajo

La recompensa del trabajo bien hecho es la oportunidad de hacer más trabajo bien hecho.

Jonas Edward Salk

6.1. Conclusiones

En el presente Proyecto Fin de Carrera se ha logrado satisfacer de manera exitosa los objetivos marcados en un principio. En base al servicio de telecomunicaciones aquí desarrollado y desplegado, se puede afirmar que en efecto se mejora la experiencia vivida por el participante de la *gymkhana*, frente a su participación en una *gymkhana* tradicional. Y además, la organización de la *gymkhana* se facilita y simplifica sobremanera, necesitándose para esta labor una menor cantidad de recursos y tiempo. Por lo tanto, en base al trabajo desempeñado en este proyecto, se pueden señalar las siguientes como principales conclusiones del mismo:

- La originalidad e innovación del proyecto queda patente tanto por el hecho de trabajarse con tecnologías punteras en el campo de las telecomunicaciones, como por tratarse de un proyecto sin antecedentes explotando y explorando la potencia y posibilidades de los modernos teléfonos móviles denominados *smartphones*. Además, su aplicabilidad se puede dar en múltiples y diversos campos, como pueden ser en aspectos de turismo (recorridos turísticos a través de ciudades), de aprendizaje (*e-learning* y *m-learning*, con *gymkhanas* con fines didácticos y/o divulgativos), para la promoción de la *Universidad Rey Juan Carlos* y de la

Ingeniería de Telecomunicación ante futuros estudiantes universitarios dando a conocer un determinado campus en las diversas recepciones organizadas para este tipo de alumnado, o simplemente para ocio en sustitución de las *gymkhanas* tradicionales o complementando otras actividades con cierta relación como pueda ser la práctica del *geocaching*.

- El proyecto aquí presentado cuenta con una alta correlación con el tipo de proyectos que se pueden desarrollar en el mundo empresarial. La primera razón de esto se encuentra en que en el presente Proyecto Fin de Carrera se han cubierto todas las fases por las que debe atravesar un proyecto empresarial, desde su diseño e implementación, hasta su posterior despliegue, pruebas en escenario real, perfeccionamiento y resolución de problemas. Esto también se debe a la combinación y uso dentro de un mismo proyecto de múltiples tecnologías todas ellas punteras, así como de distintas áreas de conocimiento tales como los sistemas operativos, las redes y protocolos de comunicaciones, las bases de datos, la *web*, la telefonía móvil, etc.

  Igualmente se debe tener en cuenta en este aspecto el trabajo realizado durante el despliegue del sistema, buscando en todo momento que el servicio desarrollado fuera íntegramente funcional y operativo en cualquier momento y escenario. Además, también se debe tener presente la toma de contacto en este proyecto con otros tipos de tecnologías como puedan ser *Subversion* (para el control de versiones en un proyecto de *software*), *Moodle* (para la elaboración de las encuestas de los experimentos) o la edición de vídeos (para la promoción y demostración *a posteriori* de los experimentos realizados en el proyecto).

- En consonancia con la cercanía del presente proyecto a uno propio del mundo empresarial, se debe tener en cuenta la importancia que en todo momento ha tenido la componente humana. Esto se refiere desde el propio hecho de haber trabajado junto al grupo de desarrolladores del proyecto *LibreGeoSocial*, hasta la búsqueda, comunicación y tratamiento con los voluntarios asistentes a los experimentos. Y por supuesto, el pensamiento continuo durante el desarrollo de la aplicación *Android* en el usuario final y en que su uso fuera lo más sencillo e intuitivo posible, pero sin renunciar nunca a la potencia que se quería otorgar al sistema.

- Una conclusión meramente técnica hace referencia al cambio radical que debe experimentar el ingeniero en su manera de pensar y de operar cuando se trabaja en el desarrollo para dispositivos limitados en recursos. Dado el trabajo sobre terminales de telefonía móvil con fuertes limitaciones no sólo en la capacidad de procesamiento o de almacenamiento en memoria, sino también en la batería, en las conexiones de red y su ancho de banda o en las dimensiones de la pantalla del móvil,

los parámetros a tener en cuenta durante el desarrollo cambian completamente respecto a por ejemplo, una aplicación de escritorio para ordenadores personales. Con esto, el desarrollo para estos dispositivos debe estar optimizado y muy meditado, pensando además siempre en el usuario final dado que estará utilizando un tipo de aplicación con la que a día de hoy no está familiarizado debido a la aparición reciente en el mercado de estos *smartphones*.

6.2. Futuras Líneas de Trabajo

La lista de posibles trabajos futuros que complementen y mejoren el resultado final de este Proyecto Fin de carrera son inagotables. Entre los más destacados se encuentran los siguientes:

- Ofrecer soporte para la participación en la *gymkhana* con varios teléfonos móviles por cada equipo, siendo la situación ideal en un futuro hipótetico aquella en la que cada miembro de un equipo participase con su propio teléfono *Android*, lo que a su vez podría dar origen a nuevos retos de colaboración entre los integrantes del equipo, separándose geográficamente entre sí en un determinado momento pero pudiendo conocer la posición de todos ellos a través del mapa virtual de la aplicación *Android*, etc.

  Sin embargo, ante esta línea de trabajo se presentan múltiples problemas cuya resolución necesitaría de un estudio en profundidad. Estos problemas tienen que ver con la sincronización en tiempo real de los distintos teléfonos *Android* participantes dentro de un mismo equipo ante por ejemplo, el envío de cada respuesta a un reto y paso a la siguiente prueba, el estudio de la posibilidad de que cualquier miembro del equipo pudiera realizar el envío de una respuesta o si fuera más conveniente declarar un teléfono a modo de capitán del equipo y siendo ese el único con ciertas capacidades a modo de privilegios, etc.

  Además, la resolución de todos estos problemas por medio de la técnica del *polling* ya empleada en el servicio de mensajería del proyecto podría implicar ahora un consumo de batería demasiado alto, por lo que sería conveniente estudiar posibles técnicas basadas en el soporte de *Android* para el *PUSH* de correos electrónicos, implementando por tanto un sistema automático para el envío de correos a los teléfonos móviles (conectándose desde la máquina *Linux* en la que reside el servidor del sistema con una *MTA* vía *SMTP*, cosa en principio de más fácil resolución y sin coste adicional frente a la implementación de un sistema para el envío automático de *SMSs*) para su posterior intercepción en el terminal y tratamiento en consecuencia.

- Aumentar y enriquecer la variedad de los tipos de retos ya desarrollados. Un elemento fundamental en este aspecto sería la introducción de retos para cuya superación fuera necesario el uso del módulo de realidad aumentada desarrollado en *LibreGeoSocial*, lo que sin duda aumentaría la potencia del sistema de *gymkhanas* sobremanera. E igualmente, introducir retos relacionados con el manejo de ficheros de audio, vídeos o alarmas ante determinados eventos como pueda ser la cercanía a un determinado amigo, lugar, recurso virtual, etc. (en todos los casos ya se cuenta con soporte para su uso en *LibreGeoSocial*), retos de orientación combinando tanto el uso del *GPS* como el de la brújula del teléfono, así como pruebas colaborativas entre distintos equipos con tal de conseguir un fin común, etc.

- Continuar optimizando el *sotware* desarrollado hasta el momento y buscar alternativas que ofreciendo una misma funcionalidad puedan conllevar un menor consumo de batería dado que éste es uno de los principales problemas que se presentan en el desarrollo de aplicaciones destinadas a *smartphones*, tal y como se ha estudiado a lo largo de la memoria, así como continuar estudiando posibles cambios en la interfaz gráfica de usuario que permitan aumentar la sencillez y facilidad de uso de la aplicación de usuario.

- Por último, dos aspectos de menor entidad pero no por ello con menos interés, como es la mejora en la automatización de la validación de las respuestas a retos textuales mediante la definición de una heurística basada en algoritmos para el cálculo de distancias entre cadenas de caracteres como pueda ser la *Distancia de Levenshtein* y sus derivados. Por otro lado, el desarrollo de una versión de la aplicación *Android* destinada al *Manager* de la *gymkhana* por medio de la cual pudiera realizar las acciones propias del control y monitorización del transcurso de la actividad con plena movilidad con tan sólo portar el pequeño terminal de telefonía móvil, pero además, de manera mucho más cómoda que por medio del acceso a través del navegador del teléfono a la aplicación *web* ya desarrollada.

Apéndice A

Resultados de las Encuestas de los Experimentos

En este apéndice se presentan las encuestas realizadas a los voluntarios de los experimentos programados para probar el sistema de organización y realización de *gymkhanas* junto con sus correspondientes resultados.

A.1. Encuesta del Primer Experimento

A continuación se presentan los resultados obtenidos en la encuesta de la primera prueba del sistema de *gymkhanas*.

Respuestas enviadas: 8
Preguntas: 40

1. ¥alore la facilidad de uso del terminal (reconocimiento de las pulsaciones en la pantalla táctil, botones del teléfono móvil, etc.).

- Malo: 0
- Regular: 4 (50.00 %)
- Bueno: 3 (37.50 %)
- Muy Bueno: 1 (12.50 %)

2. ¥alore la calidad de las instrucciones e indicaciones realizadas sobre el manejo del terminal móvil.

- Malo: 0
- Regular: 3 (37.50 %)
- Bueno: 3 (37.50 %)
- Muy Bueno: 2 (25.00 %)

3. ¿Las instrucciones e indicaciones fueron necesarias, o el manejo del terminal es suficientemente intuitivo?

- Muy Necesarias: 0
- Necesarias: 1 (12.50 %)
- Intuitivo: 7 (87.50 %)
- Muy Intuitivo: 0

4. Valore la calidad de la conexión de datos del terminal móvil (si hubo desconexiones de la red, si se experimentó lentitud en la conexión, etc.).

- Malo: 1 (12.50 %)
- Regular: 2 (25.00 %)
- Bueno: 3 (37.50 %)
- Muy Bueno: 2 (25.00 %)

5. ¿La información o ayuda sobre el uso de la aplicación es suficiente y la adecuada o resulta excesiva?

- Innecesaria: 0
- Excesiva: 0
- Necesaria pero Insuficiente: 0
- Necesaria y Suficiente: 8 (100.00 %)

6. Valore la interfaz gráfica de usuario (GUI) en cuanto a su sencillez y facilidad de manejo.

- Malo: 0
- Regular: 0
- Bueno: 7 (87.50 %)
- Muy Bueno: 1 (12.50 %)

7. La interfaz resulta fácil de usar, intuitiva y agradable.

- Malo: 0
- Regular: 1 (12.50 %)
- Bueno: 6 (75.00 %)
- Muy bueno: 1 (12.50 %)

8. Las pantallas de la GUI, resultan llamativas, originales y atractivas.

- Malo: 0
- Regular: 1 (12.50 %)
- Bueno: 6 (75.00 %)
- Muy Bueno: 1 (12.50 %)

9. En cada pantalla de la GUI, ¿el texto y demás elementos presentados son insuficientes, adecuados o excesivos?

- Insuficiente: 0
- Excesivo: 0
- Adecuado: 8 (100.00 %)
- Muy Bueno: 0

10. ¿En cada pantalla de la GUI queda siempre resaltado lo importante?

- Malo: 0
- Regular: 1 (12.50 %)
- Bueno: 7 (87.50 %)
- Muy Bueno: 0

11. Valore la calidad estética de los elementos: títulos, menús, ventanas, avisos emergentes, iconos, botones, barras de estado, etc.

- Malo: 0
- Regular: 3 (37.50 %)
- Bueno: 4 (50.00 %)
- Muy Bueno: 1 (12.50 %)

12. Valore el uso en cada momento del elemento adecuado dependiendo de la información presentada (menús, avisos, iconos, etc.).

- Malo: 0
- Regular: 3 (37.50 %)
- Bueno: 5 (62.50 %)
- Muy Bueno: 0

13. Valore la calidad de la visualización de la información en pantalla en base a los tamaños de letra, colores elegidos, etc. en la presentación de retos, pistas, alertas, avisos emergentes.

- Malo: 0
- Regular: 2 (25.00 %)
- Bueno: 5 (62.50 %)
- Muy Bueno: 1 (12.50 %)

14. En las pruebas cuya respuesta es en forma de texto, valore la detección de dichas respuestas como correctas o incorrectas (si cree que ha introducido una respuesta correcta y que le fue rechazada, etc.).

- Malo: 1 (12.50 %)
- Regular: 0
- Bueno: 3 (37.50 %)
- Muy Bueno: 4 (50.00 %)

15. Valore la GUI realizada para las pruebas fotográficas (la manera elegida para el lanzamiento de la cámara, la selección de foto a enviar, etc.)

- Malo: 1 (12.50 %)
- Regular: 0
- Bueno: 5 (62.50 %)
- Muy Bueno: 2 (25.00 %)

16. Valore la calidad de las pruebas de geolocalización, en base a la precisión a la hora de detectar la llegada al lugar indicado, el tiempo que la aplicación tarda en indicar la consecución del reto y paso a la siguiente prueba, etc.

- Malo: 5 (62.50 %)
- Regular: 3 (37.50 %)
- Bueno: 0
- Muy Bueno: 0

17. Valore la velocidad de ejecución (interacción) del servicio/aplicación respecto a las acciones del usuario.

- Malo: 0
- Regular: 3 (37.50 %)
- Bueno: 5 (62.50 %)
- Muy Bueno: 0

18. ¿En algún momento se ha echado en falta la presentación de una barra de progreso para evitar la posible sensación de un cuelgue de la aplicación?

- No: 5 (62.50 %)
- Sí: 3 (37.50 %)

19. En caso de haber respondido afirmativamente a la pregunta anterior, indique bajo qué acciones u operaciones.

-
-
- Cuando se da a enviar la respuesta creo que haría falta una barra
-
-
- Tras envíar la respuesta a una prueba, la aplicacion ha perdido la conexion y tras varios reenvios ha logrado funcionar.
-
-

20. ¿Ante ciertas acciones, ha echado de menos alertas por medio de sonidos, leds, vibración, etc.?

- No: 2 (25.00 %)
- Sí: 6 (75.00 %)

21. En caso de haber respondido afirmativamente a la pregunta anterior, indique bajo qué acciones u operaciones.

- Mensajería
- Nos habéis dicho que nos habéis mandado mensajes pero no sabíamos cómo abrirlos ni nos hemos dado cuenta hasta que hemos terminado la prueba
- Cuando recibes un mensaje ya que el icono de mensajes es pequeño y con la luz del sol no se ve bien
- Al recibir los mensajes
-
-
- al enviar una respuesta correcta o con fallo. Cuando aparece una nueva pregunta.
- Mensajes recibidos

22. Durante la gymkhana, ¿ha tenido la sensación de que la batería se ha descargado mucho y rápido?

- No: 2 (25.00 %)
- Sí: 0
- Lo mismo que cualquier otra aplicación o videojuego: 1 (12.50 %)
- No me he fijado: 5 (62.50 %)

23. ¿Ha activado el auto-check de los mensajes internos de la aplicación de la gymkhana?

- No: 1 (12.50 %)
- Sí: 7 (87.50 %)
- Durante un tiempo y lo he desactivado: 0

24. ¿Recibió los mensajes internos de la aplicación de la gymkhana inmediatamente? Explique si activó el auto-check de los mensajes entrantes, si consultaba manualmente de manera periódica, si no se dio cuenta de ello, etc.

- Me di cuenta al terminar la prueba y no pude abrirlo (¿quizá no he realizado la operación adecuada?)
- No sabemos porque no nos hemos dado cuenta de los mensajes hasta el final de la gymkama
- No lo sé
- No lo sé
- Hola Jorge!!! que tal???
- No me di cuenta
- no me di cuenta, para eso hubiese puesto un sonido una vobracion con el dibijo del mensaje parpadeando
- Los recibimos, pero nos dimos cuenta de camino al laboratorio, tras terminar.

25. Durante la ejecución, ¿se produjo algún error en la aplicación, cierre inesperado de la misma, posibles bugs, etc.?

- No: 7 (87.50 %)
- Sí: 1 (12.50 %)

26. En caso de haber respondido afirmativamente a la pregunta anterior, indique bajo qué acciones u operaciones.

-
-
-
-
-
-
-
- Antes de llegar al laboratorio, ha salido un mensaje de que la aplicación tardaba en responder.

27. Valore la fiabilidad de la ejecución del servicio/aplicación.

- Malo: 0
- Regular: 1 (12.50 %)
- Bueno: 5 (62.50 %)
- Muy Bueno: 2 (25.00 %)

28. ¿Conocía la tecnología Android antes de la participación en esta gymkhana de prueba?

- Lo desconocía totalmente: 1 (12.50 %)
- Algo he oído y/o leído: 7 (87.50 %)
- A nivel de usuario: 0
- A nivel de ingeniería/desarrollo: 0

29. ¿Le interesan los temas relacionados con la computación ubicua?

- Nada: 0
- Poco: 5 (62.50 %)
- Bastante: 3 (37.50 %)
- Muchísimo: 0

30. ¿Estaría interesado en asistir a seminarios/talleres sobre Android y temas relacionados?

- Nada: 1 (12.50 %)
- Poco: 3 (37.50 %)
- Bastante: 4 (50.00 %)
- Muchísimo: 0

31. Tras la participación en la gymkhana, ¿ha aumentado su grado de interés por Android y la computación ubicua en general?

- Nada: 0
- Poco: 4 (50.00 %)
- Bastante: 4 (50.00 %)
- Muchísimo: 0

32. ¿Utiliza la conexión de datos de su teléfono móvil para conectarse a Internet, ver su correo electrónico, etc.?

- No tengo un móvil que me lo permita: 1 (12.50 %)
- No porque el servicio es muy caro: 6 (75.00 %)
- Alguna vez: 0
- Frecuentemente: 0
- Todos los días: 1 (12.50 %)

33. En caso de haber respondido afirmativamente a la pregunta anterior, indique cuál era el teléfono móvil o sistema operativo utilizado.

- iPhone: 1 (33.33 %)
- Android: 0
- Symbian: 1 (33.33 %)
- Windows Mobile: 0
- J2ME: 0
- Otros: 1 (33.33 %)

34. Valoración media de la experiencia a nivel de usuario que participa en la gymkhana.

- Malo: 0
- Regular: 0
- Bueno: 5 (62.50 %)
- Muy Bueno: 3 (37.50 %)

35. Valoración media de la aplicación como servicio innovador frente a la organización de gymkhanas de las maneras más tradicionales.

- Malo: 0
- Regular: 0
- Bueno: 2 (25.00 %)
- Muy Bueno: 6 (75.00 %)

36. Valoración media de la calidad de la aplicación/servicio.

- Malo: 0
- Regular: 1 (12.50 %)
- Bueno: 6 (75.00 %)
- Muy Bueno: 1 (12.50 %)

37. Además de Android, los terminales móviles utilizados incluyen pantalla táctil, cámara, Bluetooth, Wi-Fi, UMTS, GPS, brújula o acelerómetros. ¿Cómo se podría explotar para que la gymkhana sea mucho más atractiva, divertida y novedosa para el usuario?

- Pruebas de recorrido de distancia en tiempo determinado, pruebas de localización conjuntas a fotografía con ayuda de la brújula.
-
-
-
-
-
- Tras una prueba de geolocalización, usar la brújula para realizar un recorrido mediante direcciones y distancias.
  Ej: Tras geolocalizar en una posición, ir 10 metros al oeste y luego 15 al norte, para encontrar alguna información.

38. ¿Cree que podría ser una nueva forma de hacer turismo, al mismo tiempo que se aprende haciendo algo de deporte?

- Malo: 0
- Regular: 0
- Bueno: 7 (87.50 %)
- Muy Bueno: 1 (12.50 %)

39. Si tuviera oportunidad, ¿repetiría la experiencia junto a sus compañeros de clase, amigos, etc.?

- Seguro que no: 0
- No lo creo: 0
- Probablemente sí: 3 (37.50 %)
- Seguro que sí: 5 (62.50 %)

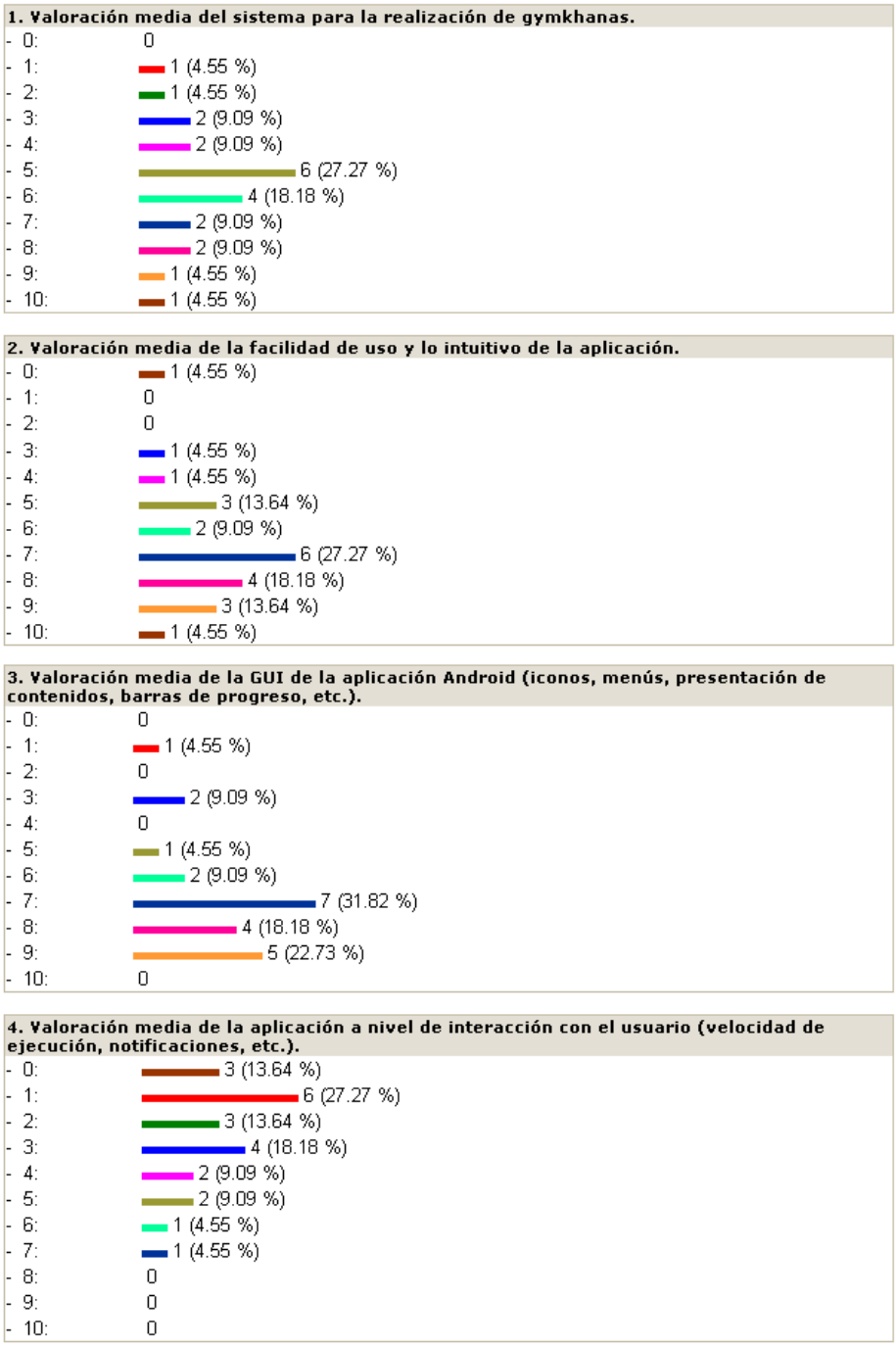
40. Por último, deje aquí cualquier comentario o sugerencia que desee realizarnos, posibles problemas que se han producido durante la ejecución y que no están contemplados en la encuesta, posibles matizaciones a respuestas dadas en las preguntas anteriores...

- Falta información sobre algunos aspectos de uso del móvil (mínimos). Durante la prueba de localización, el bloqueo de la pantalla dificultaba ligeramente el uso del GPS.
- la localización debería ser más rápida y eficaz, ya que no detectaba nuestra posición tras un buen rato parados.
-
-
-
-
-
-

A.2. Encuesta del Segundo Experimento

A continuación se presentan los resultados obtenidos en la encuesta de la segunda prueba del sistema de *gymkhanas*.

Respuestas enviadas: 22
Preguntas: 18



5. Valoración media de la fiabilidad de la aplicación (cierres inesperados, cuelgues, posibles bugs, etc.).

- 0: 8 (36.36 %)
- 1: 8 (36.36 %)
- 2: 3 (13.64 %)
- 3: 3 (13.64 %)
- 4: 0
- 5: 0
- 6: 0
- 7: 0
- 8: 0
- 9: 0
- 10: 0

6. Valoración media de las indicaciones e instrucciones de manejo del terminal móvil y de la aplicación.

- 0: 0
- 1: 2 (9.09 %)
- 2: 0
- 3: 2 (9.09 %)
- 4: 0
- 5: 4 (18.18 %)
- 6: 5 (22.73 %)
- 7: 4 (18.18 %)
- 8: 1 (4.55 %)
- 9: 3 (13.64 %)
- 10: 1 (4.55 %)

7. ¿Ha participado con anterioridad en gymkhanas tradicionales (en lugar de nuevas tecnologías, validando las respuestas ante un árbitro de la gymkhana, lectura de retos y pistas en papeles/tarjetas, consulta de mapas tangibles en lugar de GoogleMaps, etc.)?

- No: 6 (27.27 %)
- Sí: 16 (72.73 %)

8. ¿Cree que el uso de las nuevas tecnologías para estas gymkhanas frente a las tradicionales, mejora la experiencia a nivel de usuario, especialmente si se siguieran explorando nuevos tipos de pruebas por medio de módulos de realidad aumentada, etc.?

- Malo: 0
- Regular: 2 (9.09 %)
- Bueno: 14 (63.64 %)
- Muy Bueno: 6 (27.27 %)

9. Valoración media de la experiencia vivida a nivel de usuario que participa en la gymkhana.

- 0: 0
- 1: 0
- 2: 1 (4.55 %)
- 3: 2 (9.09 %)
- 4: 0
- 5: 5 (22.73 %)
- 6: 3 (13.64 %)
- 7: 7 (31.82 %)
- 8: 1 (4.55 %)
- 9: 2 (9.09 %)
- 10: 1 (4.55 %)

10. Repetiría la experiencia de participar en una gymkhana que contase con el mismo sistema que la organizada aquí.

- Seguro que no: 1 (4.55 %)
- No lo creo: 3 (13.64 %)
- Probablemente sí: 13 (59.09 %)
- Seguro que sí: 5 (22.73 %)

11. ¿Se ha divertido?

- 0: 0
- 1: 0
- 2: 0
- 3: 0
- 4: 1 (4.55 %)
- 5: 5 (22.73 %)
- 6: 3 (13.64 %)
- 7: 3 (13.64 %)
- 8: 4 (18.18 %)
- 9: 2 (9.09 %)
- 10: 4 (18.18 %)

12. Señale al menos un punto fuerte del sistema probado.

- facilidad de uso
- La facilidad de uso
- La facilidad de uso
- Es más divertido y atractivo a nivel usuario por las características del sistema
- Innovador
- Pues que la gyncana en si tiene otro punto de vista, mas participacion entre los competidores a "tiempo real"
- la facilidad en el manejo de la aplicacion
- Sistema de pistas (el sistema)
  Variedad de pruebas
- Si se mejora puede estar bastante bien.
- Sistema de mensajes.
- GUI sencilla y potente.
  Comunicación enter usuarios y con el manager.
- La comunicación con los manager
- La facilidad de uso.
- Las pruebas son divertidas y el sistema es bastante intuitivo
- Permite nuevas pruebas por el uso del gps y demas.
- la experiencia del usuario al utilizar el programa, cuando funciona.
- La interacción del dispositivo con el usuario a la hora de introducir las respuestas.
- La interfaz de usuario me ha parecido bastante buena.
- El contenido variado diverso. Ademas hace uso de el hardware del movil (camara,gps etc)
- Las pruebas muy variadas, el tema de las fotos es muy original.
- El interfaz gráfico
- La disposición de todas las herramientas necesarias en la aplicación (lanzamiento de la cámara, mapa con la localización,...)

13. Señale al menos un punto débil del sistema probado.

- Muchos cuelgues
- Iba muy lento y se colgaba demasiado la aplicación
- Se colgaba constantemente la aplicacion
- El incipiente desarrollo del mismo y los consiguientes fallos del dispositivo.
- Lento
- Que el telefono no iba bien
- las veces que nos ha tocado reiniciar
- Sistema de pistas (las pistas eran demasiado genéricas)
  Inestabilidad (sobre todo en movimiento y ante bajadas de cobertura)
- La fiabilidad
- Demasiados problemas con el GPS activado.
- Al habilitar el GPS el sistema obligaba a forzar el cierre con mucha frecuencia.
- la inestabilidad cuando esta el gps activo
- El sistema se quda boqueado cuando se activa el gps e exteriores.
- El sistema es lento y se cuelga mucho
- Los bugs, que son mas constantes de lo que debiera, ademas de quedarse el movil colgado cada cierto tiempo.
- La aplicación es muy poco fiable, se cuelga constantemente, y muchas veces no arranca
- Los cierrres inesperados. Es poco estable.
- Lentitud, se quedaba colgado muchas veces,se reiniciaba
- Se queda colgado con facilidad
  El sistema de mensajes impide que funcione bien la aplicacion
- El sistema no funciona bien, se cuelga cada dos por tres y tardaba demasiado en cargar y enviar respuestas. Al final se ha reiniciado y nos ha vuelto a salir una prueba del principio.
- La funcionalidad
- La facilidad con que se cuelga, al salir al exterior.

14. Escriba aquí cualquier comentario que desee realizar, sugerencias, posibles mejoras, problemas con la aplicación, etc.

-
-
- Ha habido muchos problemas de cuelgue
- Me parece una idea muy atratctiva. Sólo que el dispositivo se bloqueaba muy fácilmente.
-
- -Se queda bloqueado cada letra que pulsabas, por lo que las pruebas eran muy lentas y por lo tanto aburridas.
  -Como sugerencia mejorar el sistema del telefono
- Que no fuerze tanto el cierre
- Sobre los errores, es preferible un intento de reconexión o similar por parte de la aplicación que por parte de la VM de Android.
  Sobre el sistema de mensajería, se preferirían los mensajes nuevos primero.
-
-
- La localización con GPS debe ser mejorada.
- corregir qla inestabilidad del gps, y que no todos los equipos realicen las pruebas a la vez
-
- que no se cuelgue tanto!!!
-
- Creo que se debería trabajar en la fiablidad (cuelgues, lentitud...)
- Mejorar la estabilidad. La experiencia de usuario se ve muy reducida al cerrarse tanto la aplicación.
- Mas velocidad y mas robustez en la aplicacion
-
- Si el móvil no se hubiese colgado y hubiese ido un poco más rápido habría estado genial
- Creo que es una aplicación demasiado pesada para un terminal movil, y debería suprimir parte de la funcionalidad en pos de la eficiencia.
- Solucionar el que la aplicación se cuelgue al salir al exterior, cuando el móvil intenta usar la señal de GPS.

15. ¿Participó en la primera gymkhana de prueba que fue organizada el 11 de febrero de 2010?

- No: 18 (81.82 %)
- Sí: 4 (18.18 %)

16. Si participó en la primera gymkhana de prueba, indique si ha supuesto una verdadera mejora la introducción de vibraciones y barras de progreso en determinadas acciones de la aplicación, y si esto ha paliado los aparentes problemas de la primera prueba.

- Malo: 0
- Regular: 3 (13.64 %)
- Bueno: 1 (4.55 %)
- Muy Bueno: 0

17. Si participó en la primera gymkhana de prueba, indique si la notificación de la recepción de nuevos mensajes ha mejorado lo suficiente (notificación en barra de estado, notificación en pantalla y vibración).

- Malo: 0
- Regular: 0
- Bueno: 3 (13.64 %)
- Muy Bueno: 1 (4.55 %)

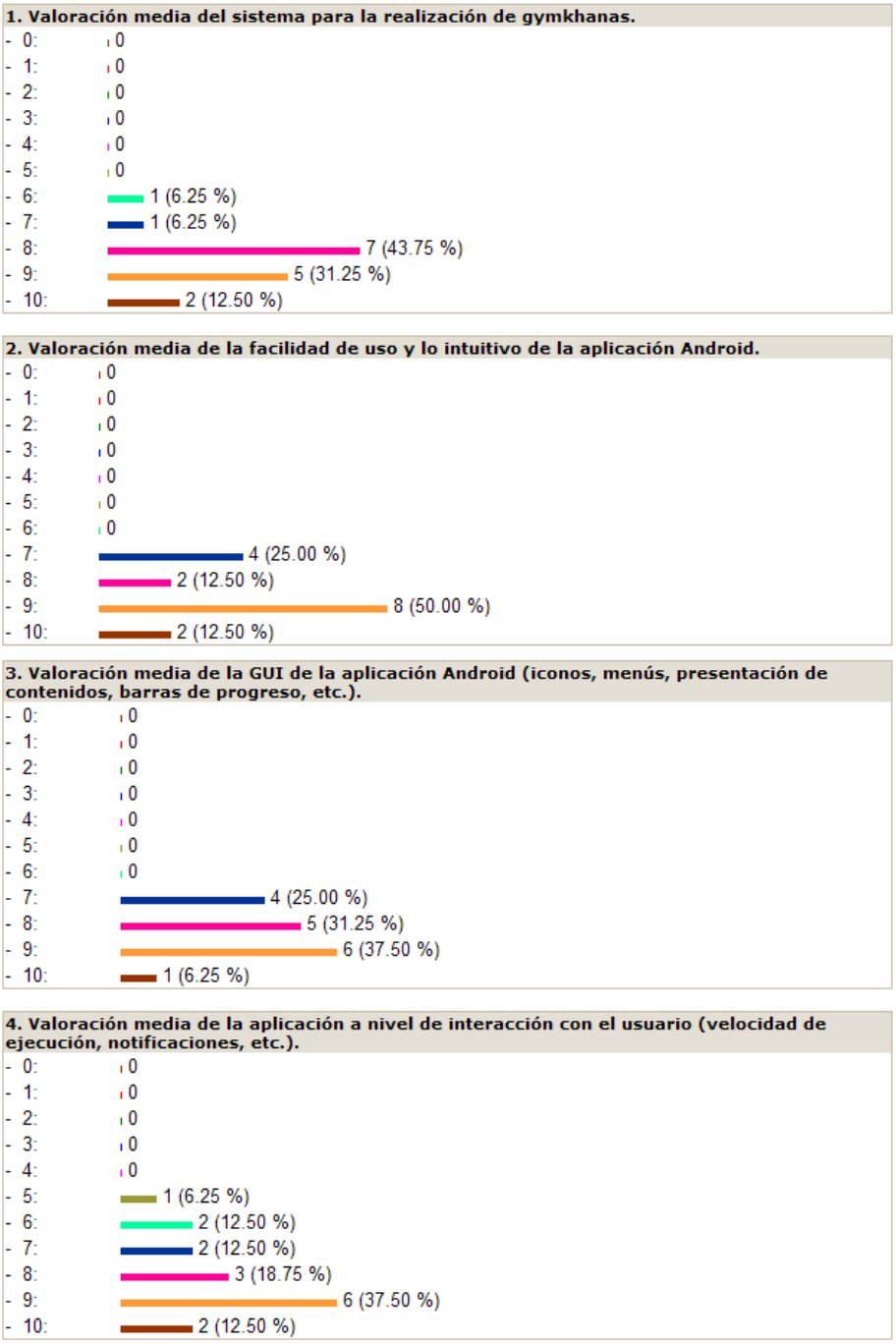
18. Si participó en la primera gymkhana de prueba, indique si considera que la detección de la posición en las pruebas de geolocalización ha mejorado lo suficiente.

- Malo: 1 (4.55 %)
- Regular: 1 (4.55 %)
- Bueno: 1 (4.55 %)
- Muy Bueno: 0

A.3. Encuesta del Tercer Experimento

A continuación se presentan los resultados obtenidos en la encuesta de la tercera prueba del sistema de *gymkhanas*.

Respuestas enviadas: 16
Preguntas: 18



5. Valoración media de la fiabilidad de la aplicación (si se produjeron cierres inesperados, cuelgues, posibles bugs, etc.).

- 0: 0
- 1: 0
- 2: 0
- 3: 1 (6.25 %)
- 4: 2 (12.50 %)
- 5: 1 (6.25 %)
- 6: 1 (6.25 %)
- 7: 1 (6.25 %)
- 8: 1 (6.25 %)
- 9: 5 (31.25 %)
- 10: 4 (25.00 %)

6. Valoración media de las indicaciones e instrucciones de manejo del terminal móvil y de la aplicación.

- 0: 0
- 1: 0
- 2: 0
- 3: 0
- 4: 0
- 5: 0
- 6: 0
- 7: 3 (18.75 %)
- 8: 5 (31.25 %)
- 9: 7 (43.75 %)
- 10: 1 (6.25 %)

7. Valore el funcionamiento-validación de la aplicación ante las pruebas de geolocalización con el GPS.

- 0: 0
- 1: 0
- 2: 0
- 3: 0
- 4: 0
- 5: 1 (6.25 %)
- 6: 0
- 7: 1 (6.25 %)
- 8: 6 (37.50 %)
- 9: 3 (18.75 %)
- 10: 5 (31.25 %)

8. ¿Ha participado con anterioridad en gymkhanas tradicionales (en lugar de nuevas tecnologías, validando todas las respuestas ante un árbitro, lectura de retos y pistas en papeles/tarjetas, consulta de mapas tangibles en vez de GoogleMaps, etc.)?

- No: 6 (37.50 %)
- Sí: 10 (62.50 %)

9. ¿Cree que el uso de las nuevas tecnologías para estas gymkhanas frente a las tradicionales, mejora la experiencia a nivel de usuario, especialmente si se siguieran explorando nuevos tipos de pruebas por medio de módulos de realidad aumentada, etc.?

- Malo: 0
- Regular: 1 (6.25 %)
- Bueno: 7 (43.75 %)
- Muy Bueno: 8 (50.00 %)

10. Valoración media de la experiencia vivida a nivel de usuario que participa en la gymkhana.

- 0: 0
- 1: 0
- 2: 0
- 3: 0
- 4: 0
- 5: 0
- 6: 0
- 7: 2 (12.50 %)
- 8: 3 (18.75 %)
- 9: 8 (50.00 %)
- 10: 3 (18.75 %)

11. Repetiría la experiencia de participar en una gymkhana que contase con el mismo sistema que la organizada aquí.

- Seguro que no.: 0
- No lo creo.: 0
- Probablemente sí.: 8 (50.00 %)
- Seguro que sí.: 8 (50.00 %)

12. ¿Se ha divertido?

- 0: 0
- 1: 0
- 2: 0
- 3: 0
- 4: 0
- 5: 0
- 6: 1 (6.25 %)
- 7: 1 (6.25 %)
- 8: 2 (12.50 %)
- 9: 5 (31.25 %)
- 10: 7 (43.75 %)

13. Si su equipo estaba compuesto por más de cinco personas: ¿cree que sería mejor poder contar con grupos participantes más pequeños o se ha sentido cómodo/a igualmente?

-
-
- mi grupo era de 3 personas
- Me he sentido cómodo con la prueba.
- Se estaba cómodo.
- Hemos sido 5 en el transcurso de la prueba, y en ningún momento me he sentido incómodo.
-
- Éramos 4
-
-
-
- grupos mas pequeños es mejor , porque estas mas en contacto con el movil
- Éramos 4. Me parece un buen número.
-
-
- Bien , eramos 2 en el grupo , aunque el grupo ideal seria segun mi opinion 4 personas :P

14. Señale al menos un punto fuerte del sistema probado.

- pruebas divertidas y rápida interacción con el usuario
- Interfaz de usuario y manejo intuitivo
- Interfaz intuitivo
- Velocidad de la aplicacion
- El sistema es fiable, sencillo e intuitivo
- La intuitividad en el uso, y la velocidad de respuesta.
- el gps
- Todo. Mejora todos los puntos respecto a una gymkhana tradicional.
- La facilidad de uso y la velocidad de la apliacción. Aplicación robusta.
- Muy útil para publicitar nuevos modelos de terminal (el usuario descubre cómo funciona, la usabilidad, lo lleva en la mano, comprueba la calidad de las fotos...)
- Las pruebas que hacen la gymkhana muy divertida.
- la estabilidad
- Originalidad de las pruebas. Interfaz de la aplicacion
- La interacción del usuario con el sistema de geolocalización del programa para hacer las pruebas.
- La originalidad de las pruebas, sobre todo las de las fotos
- Originalidad de las pruebas , buen uso de la camara de fotos

15. Señale al menos un punto débil del sistema probado.

- el teclado es un poco pequeño para dedos normales :P
- Dificultad de manejo con el teclado (a la hora de introducir las respuestas numéricas es complicado, teclas pequeñas)
- al darle a las letras o los botones o tengo los dedos morcillones o es que no coge bien las dichas letras o números
- Hoy ha funcionado muy bien
- No se han detectado puntos débiles de interés.
- No he detectado ningún punto débil durante esta prueba.
- nose me ha parecido todo bien
- Visión general. Si lo que se busca es el mayor desplazamiento de los participantes está perfecta, pero podría mostrarse al principio la localización de todas las pruebas y facilitar la elección de las mismas por el usuario. Para un teleco es poco eficiente el tiempo empleado entre desplazamientos =P
- La aplicación no tiene puntos débiles. Del sistema, el punto débil es la interacción con el teclado táctil a la hora de escribir las respuestas de las pruebas.
- La disponibilidad de GPS no ayuda a que los participantes aprendan, por ejemplo, técnicas de orientación. Yo no me he enterado de que se podían comprar pistas.

  Si escribes una respuesta incorrecta, te lo dice al instante. ¿No es mejor "aceptar" la incorrecta y decir al final cuáles eran buenas y cuáles malas?
- El salir de la gymkhana sin preguntar al darle dos veces al boton de ir atrás
- preguntar si se desea salir de la aplicacion , porque se ha cerrado sin preguntar
- Quizá asegurarse que dando al botón atrás no se salga de la aplicación (o pregunte al usuario, que hay compañeros no demasiado hábiles...)
- Al ir regresando a pantallas anteriores de la aplicación, que se pudiera salir de la misma sin una posible notificación de ello, para evitar posibles salidas accidentales del mismo.
- El sistema se sigue colgando de vez en cuando
- Al escribir mensajes ,en nuestro movil con teclado fisico , girabamos el movil no activaba el modo horizontal y terminaba bloqueandose la aplicacion

16. Escriba aquí cualquier comentario que desee realizar, sugerencias, posibles mejoras, problemas con la aplicación, etc.

- está muy mejorada esta gymkana respecto a la anterior
- ha estado mucho mejor que la última vez, la aplicación solo se nos ha colgado dos veces y la velocidad de recuperacion ha sido mucho mas rapida
- se ha colgado 2 veces, pero se ha recuperado rapido.
- Ninguno
-
- No ha habido ningún problema con la aplicación; y la veo muy completa.
-
- * La experiencia de usuario ha mejorado enormemente respecto a ediciones anteriores.

  * Hay personal del campus al que le molesta la realización de las pruebas.
- La opción de poder visualizar todas las pruebas para planificar un itinerario.
- Si escribes una respuesta incorrecta, te lo dice al instante. ¿No es mejor "aceptar" la incorrecta y decir al final cuáles eran buenas y cuáles malas?
-
- Esta muy bien
- Lo indicado en el punto anterior.
-
- Al abrir el teclado físico el móvil se colgaba el programa y se reiniciaba la prueba
-

17. ¿Participó en alguna de las gymkhanas celebradas con anterioridad (días 11 de febrero y 25 de marzo de 2010)?

- No: 6 (37.50 %)
- Sí: 10 (62.50 %)

18. En caso afirmativo, indique en qué grado las modificaciones introducidas suponen una mejora en la fiabilidad de la aplicación, uso de GPS, etc.

- 0: 0
- 1: 0
- 2: 0
- 3: 0
- 4: 0
- 5: 0
- 6: 1 (6.25 %)
- 7: 1 (6.25 %)
- 8: 2 (12.50 %)
- 9: 5 (31.25 %)
- 10: 2 (12.50 %)

Apéndice B

Presupuesto del Proyecto

En este apéndice se presenta el desglose por partidas del presupuesto para la realización del presente proyecto, contemplándose tanto los costes materiales como el de los recursos humanos implicados en el mismo y necesarios para la finalización del proyecto contando con el cumplimiento de los objetivos marcados en un principio.

B.1. Costes Materiales

En la tabla B.1 se presenta el desglose del coste de los recursos materiales necesarios para la realización del proyecto. Explicando brevemente cada uno de los conceptos, se tiene lo siguiente:

- Ordenador para que el ingeniero lleve a cabo sus labores de desarrollo.
- Máquina virtual en la que desplegar el servidor del servicio de telecomunicaciones presentado.
- Teléfono móvil *HTC Dream* (*G1*) empleado durante la fase de desarrollo para la depuración de errores.
- Teléfonos móviles *HTC Magic* (*G2*) empleados durante los experimentos realizados con voluntarios para las pruebas del sistema.
- Los tres teléfonos *HTC Magic* (*G2*) empleados para el primer experimento contaban con tarifa plana de datos de *Vodafone* con un coste de 12 euros/mes, considerándose que se ha contado con esas tarifas planas de cara al presente proyecto durante un total de tres meses. Por tanto, en la fila de la tabla de costes relativa a este concepto, se debe tener en cuenta que se han considerado nueve unidades resultantes de multiplicar esas tres tarifas planas empleadas durante tres meses, y teniendo

finalmente en cuenta el coste de 12 euros/mes por cada mes de tarifa plana.

- Cinco tarjetas *SIM* de prepago de *Vodafone* a número nuevo, necesarias para los teléfonos *HTC Magic* (*G2*) empleados para el segundo y tercer experimento, pues de lo seis totales disponibles, tan sólo uno de ellos contaba con tarifa plana de datos (en el caso del segundo experimento).
- Recargas de saldo a las tarjetas *SIM* de prepago de *Vodafone* adquiridas para la realización del segundo y tercer experimento.
- Coste nulo en concepto de adquisición de licencias por uso de *software*. Todo el *software* involucrado en el proyecto cuenta con licencias de *software* libre: *Django*, *PostgreSQL*, *Android*, LaTeX 2_ε, etc.
- Impresora para la impresión de pequeños manuales de instrucciones sobre el manejo de la aplicación de usuario *Android*, memoria del proyecto, etc.
- Tóner para la impresora y diverso material de oficina (papel, agendas, bolígrafos, calculadora, etc.).
- Cámara réflex digital y trípode para la grabación de vídeos de los experimentos, demostraciones del funcionamiento del sistema, entrevistas a los voluntarios de los experimentos, etc.
- Alquiler de un local de trabajo debidamente equipado y acondicionado durante un total de diez meses (tiempo dedicado a la realización del proyecto). Para el cálculo del coste por este concepto, téngase en cuenta que se consideran diez unidades (diez meses) a un coste de doscientos euros por unidad (200 euros/mes).

B.2. Costes de los Recursos Humanos

Según el último estudio realizado por el *COIT* (*Colegio Oficial de Ingenieros de Telecomunicación*)[1], el intervalo salarial en que se encuentra la mayor proporción de ingenieros de telecomunicación es la comprendida entre los 18.000 y los 36.000 euros brutos anuales (remuneración propia de cuatro de cada diez encuestados). Aunque el salario más frecuente (moda) es el comprendido entre los 24.000 y 30.000 euros brutos anuales (15 % de los encuestados). Así, se estimará el coste salarial en concepto del autor del

[1]Resultados accesibles en *http://coit.es/descargar.php?idfichero=463* y publicados en abril de 2005 en la página *web* del *COIT*.

Concepto	**Uds.**	**Coste/Ud.**	**Coste Total**
Ordenador Portátil de Gama Media	1	1.100 euros	1.100 euros
Máquina Virtual	1	200 euros	200 euros
Teléfono *HTC Dream (G1) T–Mobile*	1	200 euros	200 euros
Teléfono *HTC Magic (G2) Vodafone*	6	300 euros	1.800 euros
Tarifa Plana Datos *Vodafone*	9	12 euros	108 euros
Tarjeta *SIM* Prepago *Vodafone*	5	15 euros	75 euros
Recarga Saldo *SIM* Prepago *Vodafone*	5	12 euros	60 euros
Licencias de *Software*			0 euros
Impresora Láser	1	200 euros	200 euros
Material de Oficina (tóner, papel, etc.)	1	200 euros	200 euros
Cámara Réflex Digital	1	600 euros	600 euros
Trípode	1	50 euros	50 euros
Local de Trabajo	10	200 euros	2000 euros
Coste Total Recursos Materiales			**6.593 euros**

Tabla B.1: Coste de los recursos materiales necesarios durante el proyecto

proyecto, considerando que se trata de un recién titulado, en 20.000 euros brutos anuales. Teniendo en cuenta que la realización del proyecto se ha llevado a cabo en un total de diez meses, y considerándose también doce pagas anuales, el coste del ingeniero resulta un total de 16.667 euros brutos.

Por otro lado, se estima que el coste del proyecto se ha incrementado un 10 % adicional (sobre la suma del coste material y el coste de personal, es decir, 23.260 euros) en relación a la dirección del proyecto, consultas a expertos, etc. Esto supone una cuantía de 2.326 euros.

Adicionalmente, se deben incluir los costes asociados al administrador de sistemas necesario para la creación, configuración y administración de la máquina virtual en que se ha deplegado el servicio de telecomunicaciones aquí presentado, considerando su cifra en 100 euros.

Por último, el coste de contratación de personas ajenas al proyecto para la participación en los experimentos para poner a prueba el proyecto aquí presentado y obtener un posterior *feedback*, es nulo dado su carácter altruista. Sin embargo, se debe sumar un total de 150 euros adicionales en concepto de los pequeños ágapes ofrecidos a los voluntarios a la finalización de los

experimentos a modo de gratificación.

Concepto	**Coste Total**
Ingeniero	16.667 euros
Dirección y Consultas a Expertos	2.326 euros
Administrador de Sistemas	100 euros
Gratificación a Voluntarios	150 euros
Coste Total Recursos Humanos	**19.243 euros**

Tabla B.2: Coste de los recursos humanos involucrados en el proyecto

B.3. Coste Total

Calculando finalmente el coste total del proyecto como la suma de los costes materiales y del coste asociado a los recursos humanos, el resultado obtenido es de un total de 25.836 euros, como se recoge en la tabla B.3.

Concepto	**Coste Total**
Coste Recursos Materiales	6.593 euros
Coste Recursos Humanos	19.243 euros
Coste Total del Proyecto	**25.836 euros**

Tabla B.3: Coste total de la realización del proyecto

Apéndice C

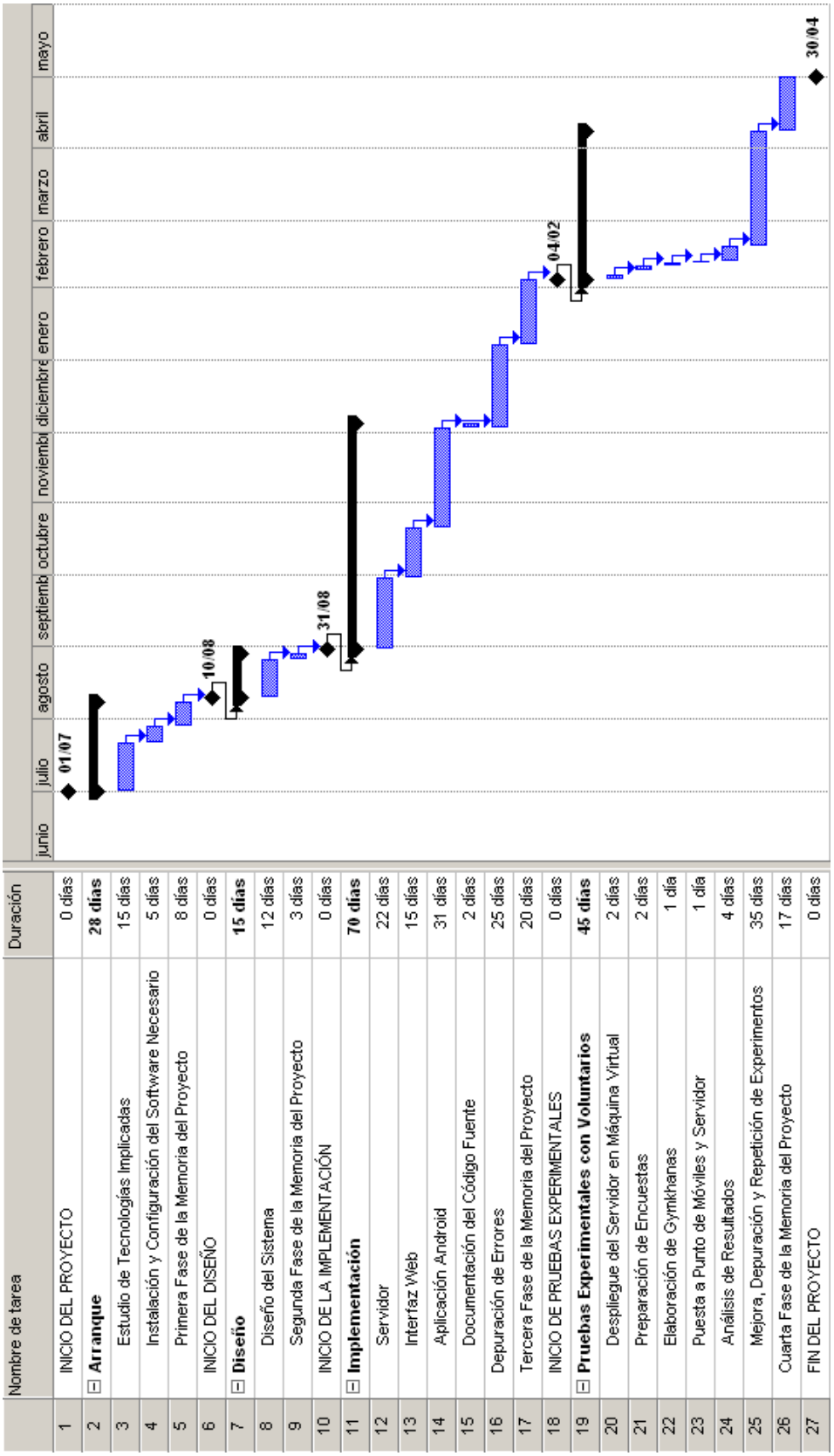
Planificación del Proyecto

En este apéndice se presenta la planificación del proyecto llevada a cabo para su finalización y consecución de objetivos, con el debido desglose de tareas y subtareas.

C.1. Diagrama de Gantt

En la figura C.1 se presenta el Diagrama de Gantt con la planificación llevada a cabo para que el proyecto alcanzara todos sus objetivos. En dicho diagrama se puede observar (además de la relación secuencial de todas las tareas debida al trabajo de un único ingeniero en el proyecto, resultando imposible la ejecución de varias tareas en paralelo) tanto la descomposición del proyecto en diversas tareas y subtareas, con la duración de cada una de ellas así como sus fechas de inicio y finalización, como el establecimiento de los principales hitos del proyecto una vez superada una tarea importante y comenzada la siguiente etapa relevante dentro del mismo.

Cabe destacar que si bien la implementación ha consumido buena parte del tiempo asignado al proyecto, la subtarea destinada a la depuración de errores dentro de dicha fase de implementación, así como la propia fase de pruebas experimentales con usuarios finales ajenos al proyecto, han contado con un peso vital, tal y como debe exigirse en el desarrollo de un servicio de telecomunicaciones destinado al gran público, de manera que no se deje lugar a posibles errores. Resaltar igualmente la importancia de la redacción al cierre de cada gran etapa del documento técnico a modo de memoria del proyecto, recogiendo los principales aspectos de la fase y evitándose así la pérdida de información valiosa durante su transcurso.



	Nombre de tarea	Duración
1	INICIO DEL PROYECTO	0 días
2	**Arranque**	**28 días**
3	Estudio de Tecnologías Implicadas	15 días
4	Instalación y Configuración del Software Necesario	5 días
5	Primera Fase de la Memoria del Proyecto	8 días
6	INICIO DEL DISEÑO	0 días
7	**Diseño**	**15 días**
8	Diseño del Sistema	12 días
9	Segunda Fase de la Memoria del Proyecto	3 días
10	INICIO DE LA IMPLEMENTACIÓN	0 días
11	**Implementación**	**70 días**
12	Servidor	22 días
13	Interfaz Web	15 días
14	Aplicación Android	31 días
15	Documentación del Código Fuente	2 días
16	Depuración de Errores	25 días
17	Tercera Fase de la Memoria del Proyecto	20 días
18	INICIO DE PRUEBAS EXPERIMENTALES	0 días
19	**Pruebas Experimentales con Voluntarios**	**45 días**
20	Despliegue del Servidor en Máquina Virtual	2 días
21	Preparación de Encuestas	2 días
22	Elaboración de Gymkhanas	1 día
23	Puesta a Punto de Móviles y Servidor	1 día
24	Análisis de Resultados	4 días
25	Mejora, Depuración y Repetición de Experimentos	35 días
26	Cuarta Fase de la Memoria del Proyecto	17 días
27	FIN DEL PROYECTO	0 días

Figura C.1: Diagrama de Gantt con la planificación del proyecto

Bibliografía

[DjangoWebSite] *Inf. téc.*, Sitio oficial de *Django*, *http://djangoproject.org*. Última fecha de consulta: febrero 2010.

[PythonWebSite] *Inf. téc.*, Sitio oficial de *Python*, *http://python.org*. Última fecha de consulta: febrero 2010.

[DjangoCommunity] *Inf. téc.*, Comunidades de desarrolladores sobre plataforma *Django*, *http://groups.google.com/group/django-users*, *http://www.djangosnippets.org/*, *http://django.es/*. Última fecha de consulta: febrero 2010.

[PostgreSQLWebSite] *Inf. téc.*, Sitio oficial de *PostgreSQL*, *http://www.postgresql.org/*. Última fecha de consulta: enero 2010.

[AndroidDeveloper] *Inf. téc.*, Sitio oficial de *Android* para desarrolladores (descarga de *SDK*, tutoriales, *API*, etc.), *http://developer.android.com/*. Última fecha de consulta: abril 2010.

[AndroidCommunity] *Inf. téc.*, Foros y comunidades de desarrolladores *Android*, *http://www.anddev.org*, *http://www.android-spa.com/*. Última fecha de consulta: abril 2010.

[EclipseWebSite] *Inf. téc.*, Sitio oficial de *Eclipse*, *http://eclipse.org*. Última fecha de consulta: octubre 2009.

[HTCDream] *Inf. téc.*, Especificaciones técnicas de *HTC Dream*, *http://www.htc.com/www/product/dream/specification.html*. Última fecha de consulta: abril 2010.

[HTCMagic] *Inf. téc.*, Especificaciones técnicas de *HTC Magic*, *http://www.htc.com/www/product/magic/specification.html*. Última fecha de consulta: abril 2010.

[UMLWebSite] *Inf. téc.*, Sitios oficiales de *UML* (*Unified Modeling Language*) y *OMG* (*Object Management Group*), *http://uml.org*, *http://omg.org*. Última fecha de consulta: abril 2010.

[XMLWebSite] *Inf. téc.*, Sitio oficial de *XML*, *http://xml.org*. Última fecha de consulta: noviembre 2009.

[JSONWebSite] *Inf. téc.*, Sitio oficial de *JSON*, *http://json.org*. Última fecha de consulta: marzo 2010.

[OverLIBWebSite] *Inf. téc.*, Sitio oficial de *overLIB*, *http://www.bosrup.com/web/overlib/*. Última fecha de consulta: diciembre 2009.

[PostgreSQLAdmin] *Inf. téc.*, Administración de *PostgreSQL* en *Linux*, *http://wiki.woop.es/PostgreSQL*. Última fecha de consulta: abril 2010.

[NexusOne] *Inf. téc.*, Especificaciones técnicas de *Nexus One Phone*, *http://www.htc.com/www/product/nexusone/specification.html*. Última fecha de consulta: abril 2010.

[ECMA262] *Standard ECMA-262 (ECMA Script Language Specification), online at http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf*, *Inf. téc.*, ECMA International, December 2009. Última fecha de consulta: marzo 2010.

[MashupTutorial] *How To Make Your Own Web Mashup, online at http://programmableweb.com/howto*, *Inf. téc.*, Última fecha de consulta: noviembre 2009.

[RFC791] *RFC791 - Internet Protocol*, *Inf. téc.*, Information Sciences Institute, University of Southern California, September 1981.

[RFC793] *RFC793 - Transmission Control Protocol*, *Inf. téc.*, Information Sciences Institute, University of Southern California, September 1981.

[Andersson:2006] E. Andersson, P. Greenspun, y A. Grumet, *Software Engineering for Internet Applications*, The MIT Press (*online at*

http://philip.greenspun.com/seia/), March 2006, ISBN 9780262511919.

[Angulo-Martínez:2009] I. Angulo-Martínez, *GPS, Compás, Sónar, RFD Control de Motores e Internet : Tecnologías Avanzadas*, Creaciones Copyright, 2009, ISBN 9788496300842.

[W3CXHTML] M. Baker, M. Ishikawa, S. Matsui, P. Stark, T. Wugofski, y T. Yamakami, *XHTML Basic 1.1, http://www.w3.org/TR/xhtml-basic/*, *Inf. téc.*, *W3C*, World Wide Web Consortium, July 2008. Última fecha de consulta: diciembre 2009.

[RFC1738] T. Berners-Lee, L. Masinter, y M. McCahill, *RFC1738 - Uniform Resource Locators (URL)*, *Inf. téc.*, Network Working Group, *IETF* (Internet Engineering Task Force), December 1994.

[W3CCSS21] B. Bos, T. Çelik, I. Hickson, y H. W. Lie, *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, http://www.w3.org/TR/CSS21/*, *Inf. téc.*, *W3C*, World Wide Web Consortium, September 2009. Última fecha de consulta: diciembre 2009.

[W3CXML] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, y F. Yergeau, *Extensible Markup Language (XML) 1.0 W3C Recommendation, http://www.w3.org/TR/REC-xml/*, *Inf. téc.*, *W3C*, World Wide Web Consortium, November 2008. Última fecha de consulta: diciembre 2009.

[Burnette:2008] E. Burnette, *Hello, Android: Introducing Google's Mobile Development Platform*, 1st Ed., Pragmatic Bookshelf, December 2008, ISBN 9781934356173.

[RFC4627] D. Crockford, *RFC4627 - The application/json Media Type for JavaScript Object Notation (JSON)*, *Inf. téc.*, Network Working Group, *IETF* (Internet Engineering Task Force), July 2006.

[Miguel-Castaño:1999] A. de Miguel-Castaño, M. Piattini, y E. Marcos, *Diseño de Bases de Datos Relacionales*, Ra-Ma, 1999, ISBN 8478973850.

[Eguíluz-Pérez:2007] J. Eguíluz-Pérez, *Introducción a AJAX*, *online at http://www.librosweb.es/ajax*, June 2008.

[RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, y T. Berners-Lee, *RFC2616 - Hypertext Transfer Protocol – HTTP/1.1*, *Inf. téc.*, Network Working Group, *IETF* (Internet Engineering Task Force), June 1999.

[Flanagan:2007] D. Flanagan, *JavaScript*, Anaya Multimedia, 2007, ISBN 9788441522022.

[Gramlich:2008] N. Gramlich, *Android Programming*, *online at http://andbook.anddev.org*, 2008.

[Holovaty:2006] A. Holovaty y J. Kaplan-Moss, *The Django Book. Web Framework for Python*, *online at www.djangobook.com*, 2006, ISBN 9781590597255.

[Holovaty:2007] A. Holovaty y J. Kaplan-Moss, *The Definitive Guide to Django: Web Development Done Right*, Apress, December 2007, ISBN 9781590597255.

[Kaaranen:2006] H. Kaaranen, *UMTS Networks Architecture, Mobility and Services*, 2nd Ed., John Wiley & Sons, 2006, ISBN 9780470011034.

[RFC2965] D. Kristol y L. Montulli, *RFC2965 - HTTP State Management Mechanism*, *Inf. téc.*, Network Working Group, *IETF* (Internet Engineering Task Force), October 2000.

[Lutz:2006] M. Lutz, *Programming Python*, 3rd Ed., O'Reilly, 2006, ISBN 9780596009250.

[Meier:2008] R. Meier, *Professional Android Application Development*, Wrox, November 2008, ISBN 9780470344712.

[Márquez-García:1996] F. M. Márquez-García, *UNIX. Programación Avanzada*, 2nd Ed., Ra-Ma, 1996, ISBN 8478972390.

[Naughton:1996] P. Naughton, *Manual de Java*, Osborne/McGraw-Hill, 1996, ISBN 8448106938.

[Oaks:1999] S. Oaks y H. Wong, *Java Threads*, 2nd Ed., O'Reilly, January 1999, ISBN 1565924185.

[Pimpler:2009] E. Pimpler, *Mashup Mania with Google Maps*, GeoSpatial Training Services, LLC (*online at http://geochalkboard.files.wordpress.com/2009/01/google-maps-pdf-article-v51.pdf*), January 2009.

[W3CHTML] D. Raggett, A. L. Hors, y I. Jacobs, *HTML 4.01 Specification, http://www.w3.org/TR/html4/, Inf. téc., W3C*, World Wide Web Consortium, December 1999. Última fecha de consulta: diciembre 2009.

[Richardson:2007] L. Richardson y S. Ruby, *RESTful Web Services*, O'Reilly, 2007, ISBN 9780596529260.

[Román-López:2009] R. Román-López, "Localización de Dispositivos Móviles para Redes Sociales Dinámicas.", en *Proyecto Fin de Carrera*, Escuela Técnica Superior de Ingeniería de Telecomunicación, Universidad Rey Juan Carlos, Madrid, Spain, July 2009.

[Selfa:2006] D. Selfa, M. Carrillo, y M. D. R. Boone, "A Database and Web Application Based on MVC Architecture." *Electronics, Communications and Computers, 2006. CONIELECOMP 2006. 16th International Conference on*, 48–48, 2006.

[Sinnott:1984] R. W. Sinnott, "Virtues of the Haversine." *Sky and Telescope*, 68 (2), 159, 1984.

[Stallings:2004] W. Stallings, *Data and Computer Communications*, 5th Ed., Pearson Prentice Hall, 2004, ISBN 0131833111.

[Tanenbaum:1995] A. S. Tanenbaum, *Distributed Operating Systems*, Prentice-Hall International, 1995, ISBN 0131439340.

[Tanenbaum:2003] A. S. Tanenbaum, *Computer Networks*, 4th Ed., Pearson Education International, 2003, ISBN 0130384887.

[Vlist:2007] E. van der Vlist, *Programación Web 2.0*, Anaya Multimedia, 2007, ISBN 9788441522527.

[Weiss:2000] M. A. Weiss, *Estructuras de Datos en Java*, Addison Wesley, 2000, ISBN 8478290354.

[Yujian:2007] L. Yujian y L. Bo, "A Normalized Levenshtein Distance Metric." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29 (6), 1091–1095, 2007.

[Zakas:2006] N. C. Zakas, *Ajax*, Anaya Multimedia, 2006, ISBN 8441520771.

Glosario

API Un Interfaz para Programación de Aplicaciones (*Application Programming Interface*, *API*) es un conjunto de clases que el desarrollador de *software* podrá emplear a modo de librerías o módulos, sin tener que invertir tiempo en implementar los métodos, funciones o procedimientos que se utilizan habitualmente en programación y que ya son ofrecidos por el fabricante.

AJAX Acrónimo de *Asynchronous JavaScript and XML*. No configura una tecnología en sí misma, sino que se trata de la unión de varias (*XHTML* y *CSS*; *DOM*; *XML*, *XSLT* y *JSON*; *XMLHttpRequest*; *JavaScript*). Esto permite contar con una nueva técnica de desarrollo *web* para crear aplicaciones interactivas, semejantes a las propias aplicaciones de escritorio. Estas nuevas aplicaciones se ejecutan en el cliente *web* del usuario (el navegador), al mismo tiempo que se mantiene una comunicación asíncrona con el servidor en segundo plano (normalmente intercambio de datos vía *XML* o *JSON*, nunca páginas *HTML* completas) sin que el usuario lo perciba. De este modo, se consiguen realizar cambios (actualizaciones) sobre la página que está visitando el usuario en ese momento, sin necesidad de recargarla nuevamente y por completo, lo que supone un aumento de la interactividad, de la velocidad y de los usos de la aplicación.

Android *Android* es una plataforma *software* que incluye un sistema operativo destinado a dispositivos móviles y cuyo *kernel* está basado en los sistemas *Linux*. Inicialmente fue desarrollado por *Google* tras adquirir en julio de 2005 una pequeña empresa de California llamada *Android, Inc.*, dedicada al desarrollo de *software* para teléfonos móviles.

Aplicación Cliente-Servidor Arquitectura de comunicaciones en la que confluyen un conjunto de aplicaciones basadas en dos categorías que llevan a cabo funciones distintas (un cliente que solicita servicios y un servidor que los ofrece), pero que al mismo tiempo pueden desarrollar actividades tanto de manera conjunta como de forma independiente.

Base de Datos Partiendo de que los datos son hechos conocidos que pueden registrarse y que cuentan con un significado implícito (susceptible de ser modelado), una base de datos será una colección coherente de datos con significado inherente. Es decir, se trata de un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior procesado y uso.

Base de Datos Relacional Modelo de bases de datos empleado en la actualidad para modelar problemas reales y administrar dinámicamente una colección de datos relacionados entre sí. El fundamento es por tanto el uso de relaciones, que en forma lógica se consideran conjuntos de datos denominados "tuplas". Simplificando el concepto, se podría pensar en cada relación como si fuese una tabla compuesta por una serie de registros (las filas de la tabla) que representarían las tuplas, y campos (las columnas de la tabla).

Bluetooth Tecnología inalámbrica basada en ondas de radiofrecuencia de corto alcance (2.4 *GHz* de frecuencia, banda *ISM*), dentro del ámbito de las Redes Inalámbricas de Área Personal (*Wireless Personal Area Network*, *WPAN*). Permite la transmisión de voz y datos entre diferentes dispositivos, siendo sus principales objetivos la facilitación de las comunicaciones entre equipos móviles y fijos, la supresión de cables y conectores entre éstos, la fácil creación de pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales tales como ordenadores personales y portátiles, teléfonos móviles, agendas electrónicas, *PDAs*, cámaras digitales, impresoras, etc.

Cliente Se trata de un proceso ejecutándose en una determinada máquina cuya función será solicitar un determinado servicio ofrecido por un servidor. Para ello, deberá establecer un canal de comunicación con dicho servidor para lo cual necesitará conocer su ubicación (nombre de máquina o dirección *IP*, habitualmente), proceder a la realización de la petición de servicio manteniéndose a la espera del resultado de dicho servicio (esperar una respuesta), para finalizar cerrando el canal de comunicación y ofreciendo al usuario el resultado fruto del servicio solicitado, terminándose posteriormente su ejecución.

Cookie El tráfico *web* se realiza sobre el protocolo de comunicaciones *HTTP*. Dicho protocolo no mantiene por sí mismo información de estado sobre la comunicación del cliente con el servidor. Para poder conservar la información entre una página visitada y la siguiente (ejemplos de *login* de usuario, productos almacenados en el carro en un sistema de compra *on-line*), es necesario almacenarla. Los mecanismos que tradicionalmente se han empleado para conservar esa información de estado

han sido la reescritura de *URLs*, el uso de campos ocultos en formularios y las propias *cookies* (mecanismo recogido en la *RFC 2965*). Así, se puede entender una *cookie* como un fragmento de información que el servidor envía al cliente para que éste lo almacene en su disco duro, y vuelva a enviárselo al servidor en una petición *HTTP* posterior.

CSS Acrónimo de *Cascading Style Sheets*, siendo el *W3C* (*World Wide Web Consortium*) el organismo encargado de formular la especificación que servirá como estándar para los agentes de usuario o navegadores. En *HTML* existen elementos y atributos que permiten especificar el estilo de los documentos (tipos y tamaños de fuente, colores de texto, colores de fondo, ancho de los elementos, etc.). Pero el uso de estas características de estilo de *HTML* presenta una serie de problemas como por ejemplo la dificultad para homogeneizar las características estéticas al estar incluida la información de estilo dentro del propio documento. Los ficheros denominados hojas de estilo *CSS* son por tanto un mecanismo para solventar este tipo de problemas, al configurar *CSS* un lenguaje para definir la presentación de un documento estructurado asociado y escrito en *HTML* o *XML* (y por extensión en *XHTML*).

DalvikVM Máquina virtual desarrollada para *Android* y empleada en dicha plataforma *software*, similar a las máquinas virtuales de *Java* (*JVM*, *Java Virtual Machine*) ofrecidas por *Sun Microsystems Inc.*. Incluye las librerías o *APIs* fundacionales de *Java* y por lo tanto, las funcionalidades básicas del lenguaje.

Clase En Programación Orientada a Objetos (*POO*), se corresponde con el modelo o abstracción de un tipo determinado de objetos, que presentan determinadas características o propiedades comunes a todos ellos, y cuya diferencia de valores hará que cada objeto sea una instancia única de la clase.

Django Entorno Integrado de Desarrollo (*Integrated Development Environment*, *IDE*) de aplicaciones *web*. Escrito en lenguaje *Python* y de código abierto, cumple en cierta medida con el patrón de diseño *MVC* (Modelo-Vista-Controlador). En un principio, fue concebido y desarrollado para la gestión de varias páginas orientadas a noticias de la *World Company* de Lawrence, Kansas, y fue liberado al público bajo licencia *BSD* en julio de 2005.

DOM Acrónimo de *Document Object Model*. El organismo responsable de la especificación de *DOM* es el *W3C* (*World Wide Web Consortium*). Se trata de un *API* para el procesado de documentos *XML*. Su funcionamiento se basa en la lectura del documento íntegramente, permitiendo posteriormente recorrer el árbol de elementos creado según la

jerarquía definida entre los mismos en el *XML*. Así, se definen funciones que permiten recorrer el árbol, acceder a elementos y atributos, valores, etc., siendo muy adecuado el uso de *DOM* dadas sus características, para la modificación de documentos *XML*.

Eclipse Entorno de Desarrollo Integrado (*Integrated Development Environment*, *IDE*) de código abierto y multiplataforma para el desarrollo de proyectos *software*, desarrollado actualmente por la *Fundación Eclipse*, organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

e-learning El *e-learning* o *electronic-learning* hace referencia a una metodología para la enseñanza y el aprendizaje haciendo uso de dispositivos electrónicos. Se trata principalmente de un medio electrónico para el aprendizaje a distancia o virtual, un sistema de educación electrónico en el que se integra el uso de las tecnologías de la información y otros elementos pedagógicos (didácticos) para la formación y enseñanza de los estudiantes.

Framework En el desarrollo *software*, se trata de una estructura definida que da soporte a la organización y desarrollo de otro proyecto de *software*, incluyendo para ello normalmente, compiladores, enlazadores, *APIs* y bibliotecas, bases de datos y servidores (poco recomendables cuando el proyecto pasa a modo de producción), etc., que faciliten el desarrollo, unión, prueba, ejecución, etc., de los diferentes componentes del proyecto.

GPS El Sistema de Posicionamiento Global (*Global Positioning System*, *GPS*) es un sistema global de navegación por satélite que permite determinar la posición de un ente (objeto, persona, vehículo, nave), con una precisión de incluso centímetros (cuando se emplea *GPS Diferencial*, *DGPS*) siendo el error habitual de unos pocos metros.

Gymkhana *Gymkhana* o *gymkana* es el nombre genérico con el que se conoce en la actualidad a los juegos a modo de concurso y competición en los que grandes grupos de personas realizan múltiples pruebas de habilidad, desarrollados normalmente en espacios al aire libre.

HTML Acrónimo de *HyperText Markup Language*. Desarrollado inicialmente por Tim Berners-Lee en el *CERN* (Centro Europeo de Investigación Nuclear). Para poder publicar la información y distribuirla globalmente se necesita un lenguaje universalmente entendible por las máquinas. En el *World Wide Web* este lenguaje es *HTML*. Se trata por lo tanto del lenguaje de marcado predominante en la creación de páginas *web*. Se

usa para describir la estructura y el contenido del documento de manera textual, así como para complementar el texto con objetos tales como imágenes, vídeos, sonidos, etc. Permite la publicación de documentos, el acceso a documentos cargándolos directamente en el navegador (o simplemente haciendo *click* en un hiperenlace existente en otro documento), rellenar formularios con información que es enviada al servidor para que la procese y realice cierta tarea con ella, y el incluir objetos multimedia (vídeos, imágenes, sonidos, animaciones, etc.) en los documentos.

HTTP Acrónimo de *HiperText Transfer Protocol*. Definido en la *RFC 2616* (*HTTP/1.1*), *HTTP* es un protocolo transaccional de comunicaciones a nivel de aplicación (puerto 80 por defecto, sobre el protocolo de nivel de transporte *TCP*) de petición-respuesta que se utiliza para construir sistemas distribuidos y colaborativos de información hipermedia. Se trata de un protocolo de comunicaciones sin estado, y utilizado desde 1990 por *WWW* (*World Wide Web*). Su funcionamiento básico será la conexión de un cliente a un servidor al que envía su petición y a la cual el servidor envía una respuesta. El formato de los mensajes de petición y respuesta queda especificado por el propio protocolo *HTTP*, pudiendo aparecer entre cliente y servidor diversos intermediadores tales como un *proxy* (modifica petición y respuesta), una pasarela (traduce entre distintos protocolos) o un túnel (reenvía la petición para atravesar un *firewall*).

IDE Un Entorno de Desarrollo Integrado (*Integrated Development Environment*, *IDE*) ofrece al programador un conjunto de herramientas orientadas a facilitar el trabajo, tales como un compilador o un depurador.

Java Lenguaje de Programación Orientada a Objetos (*POO*) desarrollado por *Sun Microsystems, Inc.* a principios de los años 90, teniendo entre sus principales objetivos la portabilidad entre plataformas independientemente de la arquitectura *hardware* y/o *software*, cosa que se consigue gracias a la compilación de las aplicaciones en *bytecodes* que posteriormente son interpretados por una Máquina Virtual de Java (*Java Virtual Machine*, *JVM*), siendo también posible su compilación a código nativo para la ejecución. Cuenta con un modelo de objetos sencillo y elimina herramientas de bajo nivel que suelen inducir a errores en el desarrollador, tales como la manipulación de punteros y el acceso a memoria.

JavaScript Lenguaje de *scripting* basado en objetos, utilizado principalmente integrado en un navegador *web* (existiendo también *JavaScript* para aplicaciones servidor) permitiendo el desarrollo de interfaces de usuario mejoradas y páginas *web* dinámicas. El código *JavaScript* es

enviado por el servidor incrustado en una página *HTML* solicitada por el cliente, o bien en un fichero aparte como resultado de una petición *web*. El cliente ejecutará dicho código, consiguiendo efectos de gran interés según el objetivo marcado, como puede ser la verificación de la información introducida por el usuario en un formulario sin necesidad de transmitir ningún dato por la red.

JDK El *Java Development Kit* es un grupo de herramientas para el desarrollo de *software* (aplicaciones *Java*) provisto por *Sun Microsystems, Inc.*. Incluye las herramientas necesarias para compilar, ejecutar, documentar y depurar aplicaciones y *applets* de *Java*.

JSON Acrónimo de *JavaScript Object Notation*. Recogido en la *RFC 4627*. Es un lenguaje de marcado que ofrece un formato ligero para el intercambio de datos a través de una red de comunicaciones, representando estructuras de datos. *JSON* es un subconjunto de la notación literal de objetos de *JavaScript* que no requiere el uso de *XML*. Su simplicidad ha llevado a la generalización de su uso como alternativa a *XML* en *AJAX* (para el cual fue específicamente pensado), especialmente debido a la sencillez de su parseo o análisis semántico por medio de *JavaScript*.

JVM Una Máquina Virtual de Java (*Java Virtual Machine*, *JVM*) es un programa nativo (es decir, ejecutable en una plataforma específica) capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (los *bytecodes* de *Java*), generado por el compilador del lenguaje *Java*. Estos *bytecodes* interpretables por una *JVM* hacen posible la portabilidad multiplataforma de las aplicaciones desarrolladas en lenguaje *Java*, pues tan sólo será necesario contar con la *JVM* adecuada para la plataforma sobre la cual ejecutar la aplicación.

Lenguaje de Marcado Un lenguaje de marcado o de marcas codifica un documento que, junto con el texto, incorpora etiquetas (las marcas) que contienen información adicional acerca de la estructura del texto o su presentación. El lenguaje de marcado más extendido es *HTML*, pilar fundamental del *World Wide Web*, siendo también populares y de uso extendido otros como por ejemplo *XML*.

LibreGeoSocial Proyecto desarrollado en el grupo de trabajo *LibreSoft* perteneciente al departamento *GSYC* (*Grupo de Sistemas y Comunicaciones*) de la *Escuela Técnica Superior de Ingeniería de Telecomunicación* (*Universidad Rey Juan Carlos*). También se denomina bajo este mismo nombre al servidor de este proyecto, desarrollado en *Python* y *Django* ofreciendo soporte para la creación de redes sociales móviles, incorporando la geolocalización de todos sus nodos.

LibreGeoSocialApp Dentro del proyecto *LibreGeoSocial*, se trata de la aplicación *Android* que explota la funcionalidad de la red social móvil del mismo nombre. Utiliza dispositivos físicos como el *GPS* para obtener la posición de un determinado nodo en todo momento.

Mashup Combinación de contenidos (y funcionalidad) de varios sitios en una aplicación *web* para dar origen a un nuevo contenido, funcionalidad o aplicación mucho más rica y potente. Normalmente el contenido del *mashup* proviene de sitios *web* de terceros a través del protocolo *HTTP* y ejecutando en el servidor *APIs* públicas ofrecidas por esos terceros. Siendo por lo tanto el concepto fundamental, la creación de aplicaciones mediante composición de aplicaciones *web*.

Middleware Capa de *software* (normalmente distribuida) que se suele situar entre la capa de aplicaciones y las capas inferiores (sistema operativo, e incluso, red). El *middleware* proporciona un modelo de programación (un *API*) conveniente para los programadores de aplicaciones distribuidas (facilitando su programación y manejo de aplicaciones), abstrayendo o enmascarando la complejidad y heterogeneidad de plataformas (sistemas operativos, lenguajes de programación, redes de comunicación subyacentes, etc.).

MTA Acrónimo de *Mail Transport Agent* o *Message Transport Agent*. Un agente de transferencia de correo es un programa que transfiere los correo electrónicos (o mensajes, en general) entre distintas máquinas de la red.

MVC El patrón de diseño o arquitectura *MVC* (Modelo-Vista-Controlador) separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, obteniéndose una importante serie de ventajas al estar estos tres componentes desacoplados. Así pues, el modelo mantiene los datos y los métodos de acceso a dichos datos. La vista hace referencia a la representación gráfica de los datos de un modelo. Y el controlador se encarga de la gestión de los eventos que se produzcan en la aplicación. Se trata de una arquitectura frecuentemente empleada en aplicaciones *web*.

m-learning El *m-learning* o *mobile-learning* hace referencia a una metodología para la enseñanza y el aprendizaje haciendo uso de dispositivos móviles tales como teléfonos móviles, agendas electrónicas, tablets *PC*, etc. Se trata de una evolución más que complementa las modalidades educativas que incorporan las nuevas tecnologías de telecomunicación, siendo algunos de los términos más destacados los de la enseñanza apoyada en el ordenador, la tele-educación, la enseñanza basada en *web* o el aprendizaje electrónico (*e-learning*).

Objeto En Programación Orientada a Objetos (*POO*), instancia única de una clase que mantiene la estructura y comportamiento definidos por dicha clase. Se diferenciará de sus homónimos a partir de los distintos valores que tomen sus atributos o propiedades.

PostgreSQL Sistema de Gestión de Bases de Datos (*SGBD*) relacionales orientadas a objetos, distribuido bajo una licencia de *software* libre *BSD*. Con un potente motor de bases de datos, ofrece prestaciones y funcionalidades equivalentes a muchos *SGBD* comerciales y siendo más completo que *MySQL*.

Plug-in Pequeña aplicación que a modo de complemento de otra aplicación de mayor entidad, le aporta una nueva funcionalidad y por lo general, muy específica. Esta aplicación será ejecutada por la aplicación principal, interactuando ambas por medio de una *API*. Son famosos los *plug-in* para navegadores *web* (por ejemplo *Flash* de *Adobe*, usado normalmente para la visualización de vídeos) o para los *IDE* como *Eclipse*, para ofrecer soportes a nuevos lenguajes de programación, sistemas, etc. (por ejemplo para dar soporte a *Java*, *CVS*, *UML*, *Android*, etc.), no soportados en su versión de serie.

***POO* o Programación Orientada a Objetos** Paradigma de programación que define los programas en términos de clases de objetos, siendo los objetos entidades que combinan estado (los datos), comportamiento (métodos, procedimientos o funciones) e identidad (propiedad del objeto que lo distingue del resto de objetos de su misma clase). La *POO* expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para llevar a cabo una tarea, con lo que se consigue programar de manera más modular y fácil, siendo el resultado final más sencillo de mantener y reutilizar.

Python Creado en Holanda por Guido van Rossum a finales de los años 80 y distribuido bajo una licencia de código abierto (*Python Software Foundation License*) compatible con la licencia *GPL*, *Python* es un lenguaje de programación interpretado, ofreciendo por esto mismo las ventajas de los lenguajes propios de *scripting*, como puede ser un rápido desarrollo, además de buscar una mayor facilidad tanto de lectura del propio código, como en el diseño.

QWERTY El teclado *QWERTY* es la distribución de teclado más común. Diseñado y patentado por Christopher Scholes en 1868 y vendido a Remington en 1873, su nombre proviene de las primeras seis letras de su fila superior de teclas. Igualmente, existen diversas variantes del teclado *QWERTY*, como por ejemplo en Alemania, donde se intercambian la tecla "Y" y la tecla "Z", con lo que se convierte en el teclado *QWERTZ*;

en Francia y Bélgica se introducen más cambios, teniéndose el teclado *AZERTY*; o añadiéndose en España la tecla "Ñ".

Red Social Estructura social representable en forma de uno o varios grafos en los que cada nodo representa un actor (ya sea una persona individual o un grupo), y las aristas representan las diferentes relaciones entre ellos (amistad, amor, compañerismo, admiración, relaciones profesionales, etc.). Llevado al entorno de *Internet*, se trata de aplicaciones *web* que permiten relaciones sociales entre personas (usuarios de la red social) por medio de una serie de nodos estáticos y que cuentan con un gran auge en la actualidad, siendo los ejemplos más representativos a nivel internacional *Facebook*, *MySpace* o *LinkedIn*.

Red Social Dinámica Red social en la que desempeña un papel vital la posición geográfica (tupla formada por latitud y longitud, ya sea fija, como ocurriría en el caso de un museo o de un centro comercial, o variable como por ejemplo en el caso de un usuario en movimiento) de los distintos nodos de la red. La red social se convierte en un gestor de contenidos geolocalizados que pueden estar relacionados unos con otros de la manera clásica propia de las redes sociales tradicionales, o mediante vínculos que pueden surgir dinámicamente, como puedan ser relaciones instantáneas y efímeras en torno a un interés en un momento dado.

REST Acrónimo de *REpresentational State Transfer*. Se trata de un modelo de arquitectura para sistemas distribuidos que se encuentra ampliamente extendido en la *web* estática (ficheros y directorios) y parcialmente extendido en la *web* programática (aplicaciones y servicios *web*), basándose en un conjunto de normas como son: cada recurso debe tener una *URL*, siendo un recurso cualquier tipo de información que se desea hacer visible en la *web* (documentos, imágenes, servicios, perfiles de personas, etc.); los recursos tienen hiperenlaces a otros recursos; con un conjunto limitado y estándar de métodos (*GET*, *PUT*, *POST*, *DELETE*); con múltiples representaciones por recurso que muestren el estado actual de dicho recurso en un formato dado; y teniendo siempre una comunicación sin estado (una petición del cliente al servidor debe contener toda la información necesaria para entender la misma).

SAX Acrónimo de *Standard API for XML*. Se trata de un *API* para el procesado de documentos *XML*. Procesa el documento según está disponible. Esto hace que sea útil para realizar algún procesamiento al mismo tiempo que se está leyendo el documento *XML* (como por ejemplo, buscar ciertas etiquetas ignorando las restantes). Es rápido y sencillo al no necesitar cargar el documento entero en memoria para formar y analizar el árbol de elementos como ocurre en *DOM*, pero a su vez,

resulta complicado realizar modificaciones sobre la estructura del documento.

SDK Un *Software Development Kit* (*SDK*) es un conjunto de herramientas destinadas a hacer más sencillo al programador el desarrollo de aplicaciones. Entre las herramientas que normalmente se incluyen destaca el compilador, el enlazador, el depurador y el conjunto de librerías y su documentación, acerca del lenguaje de programación.

Servidor Un servidor es un proceso ejecutándose en una determinada máquina de la red. Su cometido será gestionar diversos servicios así como el acceso a los recursos capaces de ofrecer dichos servicios. Para hacer posible esta gestión, el servidor estará en continua espera de una petición de servicio proveniente de un cliente. En el momento en que esa petición se produzca, el servidor comenzará a realizar las acciones necesarias para atender dicha petición, volviendo al estado de espera una vez que se haya terminado de ofrecer el servicio solicitado.

***SGBD* o Sistema de Gestión de Bases de Datos** En inglés *DataBase Management System*, *DBMS*. Tipo de *software* muy específico que facilita la definición, construcción y manipulación de bases de datos, sirviendo a modo de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

SMTP Acrónimo de *Simple Mail Transfer Protocol*. Protocolo de comunicaciones empleado para el envío de correo electrónico ya sea entre *MTAs* o desde un cliente de correo (*UA*, *User Agent*) a una *MTA*.

Sistema Operativo *Software* de sistema que controla de manera eficiente todos los recursos de una computadora y que establece la base sobre la que puede escribirse el *software* de aplicación. Actúa por tanto de intermediario entre el usuario de un computador y su *hardware*, siendo sus principales objetivos el control y ejecución de programas, el almacenamiento de datos de usuarios y la facilitación del uso del sistema, todo ello mediante una gestión y administración eficiente del *hardware* (recursos) disponible.

Subversion *Software* libre bajo licencia *Apache/BSD* conocido normalmente como "*svn*" por ser éste el nombre de la herramienta utilizada en línea de comandos. Se trata de un *software* para el control de versiones en el desarrollo de proyectos *software* similar a otros sistemas tan populares como *CVS* (*Concurrent Versions System*).

UML Lenguaje Unificado de Modelado (*Unified Modeling Language*, *UML*). Contando con un gran apoyo por parte del *OMG* (*Object Management Group*), es el lenguaje de modelado de sistemas *software* más conocido y utilizado en la actualidad. Permite de manera gráfica, visualizar, especificar, construir y documentar un sistema *software*.

UMTS Acrónimo de *Universal Mobile Telecommunications System*. Tecnología sucesora de *GSM* (*Groupe Spécial Mobile*, segunda generación de telefonía móvil, surgida para ofrecer servicios de voz en comunicaciones móviles y teniendo limitación para la transmisión de otros servicios) y de *GPRS* (*General Packet Radio Service*, denominada *2.5G*, extensión de *GPS* para la transmisión de datos por paquetes pero con bajas velocidades de transmisión), que supone una de las tecnologías utilizadas por los móviles de tercera generación (*3G*). *UMTS* surge precisamente para cubrir las nuevas necesidades de los usuarios y las carencias de las generaciones anteriores de telefonía móvil, como es el proporcionar altas tasas binarias para poder ofrecer al usuario final otros servicios a alta calidad (capacidades multimedia, variedad de servicios muy extensa, velocidad de acceso a *Internet* elevada, transmisión de audio y vídeo en tiempo real, y transmisión de voz con calidad equiparable a la de las redes de telefonía fija).

Web 2.0 El término *Web 2.0* (también llamada *web social*) fue empleado por primera vez en el año 2004 por Tim O'Reilly para referirse a una nueva etapa en la historia del desarrollo de tecnología *web* basada en comunidades de usuarios y una gama especial de servicios, como las redes sociales, los *blogs* o los *wikis*, que fomentan la colaboración y el intercambio ágil y eficaz de información entre los usuarios de una comunidad o red social. Desde un punto de vista meramente técnico, la *Web 2.0* supone la evolución del *web* de colección de sitios a plataforma informática completa. Proporciona aplicaciones *web* a los usuarios finales, muchas de las cuales podrán sustituir a otras aplicaciones de escritorio. Y explota los llamados efectos de red, por ejemplo con las redes sociales (dada la arquitectura de participación).

Wi-Fi Acrónimo de *Wireless Fidelity*. Conjunto de estándares para redes locales inalámbricas destinadas a la transmisión de datos por medio de ondas de radio, basado en las especificaciones *IEEE 802.11*, y utilizado también para el acceso a *Internet*. También se trata de una marca de la *Wi-Fi Alliance*, organización comercial que adopta, prueba y certifica que los equipos de los fabricantes cumplen con los mencionados estándares recogidos en *IEEE 802.11*.

XHTML Acrónimo de *eXtensible HyperText Markup Language*. *XHTML* es una redefinición de *HTML 4* como aplicación *XML*. Pese a tener distinto nombre, conserva la mayoría de los elementos y atributos de *HTML*. Una de las diferencias fundamentales con *HTML* es la ausencia de elementos y atributos relacionados con estilo (fuentes, tamaños, colores, etc.), que ya en *HTML 4* están desaconsejados, y en la existencia de unas normas más estrictas en la colocación de etiquetas de elementos. Así, en *XHTML* se refuerza la separación contenido/presentación, todas las herramientas disponibles para trabajar con *XML* se pueden emplear con *XHTML* al ser éste un tipo de documento *XML* y existen unas reglas estrictas acerca del formato que debe tener el documento.

XSLT Acrónimo de *eXtensible Stylesheet Language Transformations*. Estándar del *W3C* (*World Wide Web Consortium*) para la transformación de un documento *XML* a cualquier formato textual: *HTML*, *WML*, texto plano, otro documento *XML*, etc., empleando una sintaxis *XML* y basándose en reglas de transformación (reglas de plantilla/*template*) que se aplican sucesivamente, junto con expresiones *XPath* en atributos para evaluación de expresiones. Es decir, esas reglas de plantilla junto al documento fuente a transformar, alimentan a un procesador de *XSLT* que realiza las transformaciones deseadas volcando el resultado final en un archivo de salida o directamente en un dispositivo de presentación como puede ser el monitor del usuario (como sería el caso de una página *web*).

XML Acrónimo de *eXtensible Markup Language*. Es un metalenguaje extensible de marcado o de etiquetas, desarrollado por el *W3C* (*World Wide Web Consortium*). Se trata realmente de un dialecto simplificado y adaptado del *SGML* (*Standard Generalized Markup Language*) y permite definir la gramática de lenguajes específicos, pretendiendo ser razonablemente simple. Por tanto, *XML* no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades, siendo algunos ejemplos de lenguajes que se sirven de *XML* en la actualidad: *XHTML*, *MathML*, *RSS*, *SMIL*, *SVG*.

XMLHttpRequest Interfaz empleada para el envío de peticiones *HTTP* a servidores *web*. Para la transferencia de datos se emplea cualquier codificación basada en texto, como puedan ser *HTML*, *XML*, *JSON*, o incluso texto plano y codificaciones propietarias. El uso más popular de esta interfaz es proporcionar contenido dinámicamente mediante actualizaciones asíncronas en páginas *web*, gracias a tecnologías creadas para tal propósito y partiendo de la base de *XMLHttpRequest*, tal y como es el caso de *AJAX*.