UNIVERSIDAD
REY JUAN CARLOS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

INGENIERÍA DE TELECOMUNICACIÓN

Curso Académico 2014/2015

Proyecto Fin de Carrera

Diseño e Implementación de una Aplicación Web para Móviles: Gymkhanas de Nueva Generación

Autor : Beatriz Aldama Melero

Tutor : Dr. Gregorio Robles Martínez

Proyecto Fin de Carrera

Diseño e Implementación de una Aplicación Web para Móviles: Gymkhanas de Nueva Generación

Autor : Beatriz Aldama Melero

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2015, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2015

Dedicado a

mis padres y a mi hermana.

II

Agradecimientos

En primer lugar quiero dar las gracias por todo a mi familia. En especial a mis padres y a mi hermana por estar siempre ahí que os he necesitado, apoyarme en todo momento y confiar en que podía con ello. Por animarme y darme fuerzas para seguir cuando suspendía algún examen, y alegraros tanto o más que yo cuando aprobaba. A mis abuelas, que siempre se han alegrado de mis logros, y aunque una de ellas ya no esté con nosotros, sé que también la haría mucha ilusión verme terminar. No hay palabras suficientes que expresen mi gratitud hacia vosotros. Gracias por recorrer este camino conmigo, os quiero.

También quiero dar las gracias a mi tutor y profesor, Gregorio Robles, por darme la oportunidad de realizar este proyecto y contagiarme sus ganas de seguir con las gymkhanas móviles. Y a todos los demás profesores que durante todos estos años han contribuido en mi formación.

A todos mis amigos y compañeros con los que he pasado muy buenos momentos a lo largo de esta etapa. Mis amigas María y Soraya por ser pacientes cuando no tenía tiempo libre suficiente. A Fran, mi mejor amigo y gran compañero de estudios, con el que siempre he podido contar. Y a Yana, esa cabecita loca.

Gracias!

Resumen

Desde 2010 se contaba con una aplicación de gymkhanas educativas para móviles que sólo podía ser utilizada con dispositivos Android. Pero debido al auge de diferentes sistemas operativos, dicha aplicación se ha quedado un poco obsoleta y, por tanto, se ha limitado el uso de este canal de aprendizaje. La única forma en la que todos los usuarios pudieran disfrutar de este sistema, sería utilizar una aplicación común, Internet.

Por esto, en el presente Proyecto Fin de Carrera se ha procedido a actualizar dicho sistema, implementando una aplicación web que sea fácilmente accesible desde Internet utilizando cualquier dispositivo móvil. De manera que se pueda ampliar el número de usuarios que puedan disfrutar de este canal de aprendizaje. Para ello se han utilizado las tecnologías de desarrollo más actuales tales como HTML5, jQuery y CSS3 para implementar una interfaz web del cliente que se conectará con el servidor a través de HTTP mediante AJAX.

Acrónimos.

AJAX	**A**synchronous **J**avaScript **A**nd **X**ML
API	**A**pplication **P**rogramming **I**nterface o **I**nterfaz de **P**rogramación de **A**plicaciones
CDN	**C**ontent **D**elivery **N**etwork
CSS	**C**ascading **s**tyle **S**heets o Hoja de estilo en cascada
DOM	**D**ocument **O**bject **M**odel o **M**odelo de **O**bjetos del **D**ocumento
GPS	**G**lobal **P**ositioning **S**ystem
HTML	**H**yper**T**ext **M**arkup **L**anguage o **L**enguaje de **M**arcas de **H**iper**T**exto
JSON	**J**ava**S**cript **O**bject **N**otation
MVC	**M**odel-**V**iew-**C**ontroller o **M**odelo-**V**ista-**C**ontrolador
REST	**RE**presentational **S**tate **T**ransfer
URL	**U**niform **R**esource **L**ocator o **L**ocalizador de **R**ecursos **U**niforme
Wi-Fi	**W**ireless **F**idelity

Índice general

Índice de figuras

Capítulo 1

Introducción

En este primer capítulo se va presentar el motivo de este Proyecto Fin de Carrera pero, para ello, primero hay que recordar un trabajo anterior del que este ha partido. Además se comentará brevemente cómo está estructurada la presente memoria.

1.1. Trabajos anteriores.

El presente Proyecto Fin de Carrea consiste en la actualización de un antiguo proyecto realizado por Jorge Fernández González en 2010. Su proyecto fin de carrera *"Diseño, Implementación y Despliegue de un Servicio de Telecomunicación para M-Learning en Móviles Android: Gymkhanas de Nueva Generación"*[1] consistía, tal y como su nombre indica, en realizar la versión móvil de uno de los juegos más populares de la sociedad como son las *yincanas* o, también denominadas, *gymkhanas*, cuya finalidad es la de superar el máximo número posible de pruebas fomentando el trabajo en equipo. Uno de los objetivos principales de dicho proyecto fue conseguir un nuevo canal de aprendizaje a través de los servicios de telecomunicaciones que ofrecen los móviles, *mobile learning* o *m-learning*. De esta manera, surgen varias ventajas en cuanto a la organización y realización del juego, tales como:

- La reducción del tiempo de organización ya que solamente hay que registrar las pruebas en una base de datos.
- El aumento sobre el control de los equipos ya que el organizador de la gymkhana tiene a su disposición un servicio web donde se muestra la ubicación de cada equipo y todas las

respuestas obtenidas de cada reto.

- La eliminación de desplazamientos innecesarios ya que la aplicación cuenta con un servicio de mensajería con el que poder comunicarse con el resto de equipos o el organizador, en caso de que haya alguna duda frente a una prueba.

- La modificación de las pruebas a lo largo del transcurso de la gymkhana según crea conveniente el mánager del evento.

- La reducción de recursos materiales ya que se suprimen las tarjetas con los enunciados y sólo es necesario un dispositivo móvil que soporte la aplicación.

- La reducción de recursos humanos ya que al haber más control sobre el juego no es necesario que en cada reto haya un juez que valide las respuestas de los equipos.

1.2. Motivación.

Actualmente vivimos en una sociedad en la que el uso de dispositivos móviles, como smartphones o tablets, está a la orden de día. Y cada uno de estos dispositivos tiene su propio sistema operativo con sus consecuentes versiones y aplicaciones soportadas. Por eso a la hora de desarrollar una aplicación, un diseñador debe saber adaptarla a cada versión de los sistemas operativos. Actualmente, los más comunes son Android, iOS, Windows Phone y BlackBerry. Aunque hay alguno más, ya son demasiados para tener que desarrollar versiones diferentes de una misma aplicación.

La primera solución sería migrar la aplicación móvil a una aplicación web ya que todos los sistemas operativos tienen navegadores web para poder acceder a internet. Pero aun así, al igual que existen diversos sistemas operativos, los navegadores también tienen sus propias versiones. Por lo tanto, un diseñador, a la hora de implementar una aplicación web, debe tener en cuenta para qué tipo de navegador quiere desarrollarla.

En base a esto, es necesario un lenguaje que sea soportado por todos los navegadores. Es aquí cuando nace HTML5, un lenguaje de marcado para la elaboración de páginas web HTML, que soportan todos los micronavegadores más actuales.

Por otro lado, se tenía una aplicación Android para móviles que ofrecía un nuevo canal de aprendizaje mediante la realización de uno de los juegos más populares de la sociedad, las gymkhanas. Pero debido a la gran diversidad de sistemas operativos en la actualidad, dicho canal se ha quedado un poco restringido. Con lo cual se ha decidido evolucionar dicho sistema para que no sólo se pueda utilizar con dispositivos Android, sino en cualquier sistema operativo. Por consiguiente, el presente Proyecto Fin de Carrera es la migración de esa aplicación móvil a una aplicación multiplataforma, con el fin de que pueda utilizarse en cualquier ordenador o dispositivo móvil.

1.3. Estructura de la Memoria.

Con la intención de facilitar la lectura, se ha decidido explicar brevemente cada uno de los capítulos de la presente memoria.

- **Capítulo 1: Introducción.**

  En este capitulo se presentan las motivaciones en la realización del presente Proyecto Fin de Carrera así como trabajos anteriores. Además se muestra brevemente la estructura de la memoria.

- **Capítulo 2: Objetivos.**

  En este capitulo, como su nombre indica, se presentan los objetivos finales que tiene la realización de este proyecto. Y también un comentario sobre el tiempo llevado a cabo en el desarrollo.

- **Capítulo 3: Estado del Arte.**

  En este capitulo, se comenta de manera breve el estado de las tecnologías utilizadas en la implementación del nuevo sistema.

- **Capítulo 4: Diseño del Sistema.**

  En este capitulo, se presenta la estructura general del sistema del que se tomó partida, y la nueva estructura desarrollada.

- **Capítulo 5: Diseño e implementación del Cliente.**

  Este capitulo es el núcleo de la memoria, donde se explica todo el desarrollo llevado a

cabo para la implementación de la nueva aplicación web.

- **Capítulo 6: Resultados.**

  En este capitulo se presentan las pruebas realizadas una vez que el sistema se ha desarrollado, con el fin de que comprobar el buen funcionamiento y corregir posibles errores.

- **Capítulo 7: Conclusiones.**

  En este capitulo se hace revisión de todo lo implementado, es decir, si se han cumplido los objetivos acordados. También se ofrecen futuras líneas de trabajo para evolucionar el sistema desarrollado. Y, por último, se ofrece una valoración personal sobre el proyecto donde se comenta todo lo aprendido.

- **Bibliografía.**

  En este apartado se presenta una bibliografía con las principales referencias consultadas.

Capítulo 2

Objetivos

El objetivo principal de este proyecto es poder utilizar una aplicación de gymkhanas educativas para móviles desarrollada en Android bajo cualquier plataforma. Para ello, se necesita implementarla para una plataforma común a todos los dispositivos, los navegadores web. De esta manera, todos los usuarios que deseen utilizarla no tienen que preocuparse de si su móvil u ordenador la soporta. Por ende, se tiene como meta la migración, en su totalidad, de la aplicación Android a una aplicación web, para así conservar su finalidad educativa.

Uno de los objetivos que tenía el antiguo proyecto de Jorge Fernández era que el administrador de las gymkhanas no se tuviera que desplazar para controlar a los equipos y resolver dudas, sin embargo sí que puede necesitar desplazarse para configurar los retos de cada gymkhana. Es cierto que ya dispone de una interfaz web que se puede acceder desde cualquier dispositivo, pero su visualización resulta incómoda en pantallas pequeñas como las que tienen las tablets o *smartphones*. Por esto, se ha establecido otro objetivo para adaptar la interfaz gráfica de la aplicación web del administrador a cualquier tipo de dispositivo y tamaño de pantalla. De esta manera se dota de cierta movilidad al administrador a la hora de configurar el evento.

Otra meta es que más usuarios puedan disfrutar del canal de aprendizaje, *mobile learning* o *m-learning*, ya que según el enfoque de la gymkhana, se pueden aprender diferentes temas. Por lo que si se consigue el objetivo principal de este Proyecto Fin de Carrera, esta meta también se conseguirá. Además se desea que la nueva aplicación resulte de fácil uso para cualquier usuario.

Y finalmente también se busca una de las mejoras propuestas por Jorge Fernández. En con-

creto, la antigua aplicación Android consume mucha batería como la mayoría de aplicaciones para *smartphones*. Por lo que se va a intentar que la nueva aplicación web a desarrollar no consuma tantos recursos de los dispositivos móviles.

2.1. Planificación temporal

En este apartado se va a comentar el tiempo que se ha tardado en desarrollar el presente Proyecto Fin de Carrera.

En febrero del 2014 y durante dos días, tras elegir qué proyecto hacer, se realizó la lectura de trabajos relacionados y anteriores para obtener una idea de cómo estaba implementado y estructurado el sistema del que se ha tomado partida. Tras esto, el 4 de abril se realizó el primer experimento para tener un primer contacto con la tecnología y ver cómo se desarrolla un ejemplo de gymkhana con móviles. Después, durante una semana de junio del 2014, se estudió de manera global el diseño y la implementación del servidor, y el diseño del cliente.

En julio del 2014 se comienza realmente a desarrollar el proyecto. Como la máquina donde se aloja el servidor estaba caída, se empezó por aprender las tecnologías a utilizar tales como HTML5, jQuery y CSS, durante al menos un mes. Una vez que se adquirieron los suficientes conocimientos, se comenzaron a configurar los documentos HTML que darían lugar a las pantallas web de la aplicación del cliente. La información que mostraban no era la correcta ya que no se podía acceder al servidor para obtener los datos correctos, pero al menos funcionarían a modo de plantilla para que, una vez que el servidor estuviera accesible, los cambios fueran mínimos. Esto se desarrolló, aproximadamente, durante otro mes.

A finales de septiembre, la máquina se logró poner en pie y, por tanto, se pudo lanzar y manipular el servidor para crear la nueva aplicación web. Durante un mes y medio, a medida que se iba cambiando el servidor, se iban terminando los documentos HTML previamente diseñados. Una vez que se logró tener una versión de la nueva aplicación web lo más estable posible, se realizó el segundo experimento donde se obtuvieron errores y se propusieron algunas mejoras. A lo largo del siguiente mes se estuvo corrigiendo o solventando ciertos errores sacados tras la segunda prueba y mejorando ciertas partes del sistema.

La redacción de la memoria de este proyecto ha durado un mes y, a medida, que se iba redactando se fueron cambiando algunos detalles que no tenían gran peso en el funcionamiento de la aplicación.

A modo de conclusión y de manera aproximada se estima que la duración total para la realización de este proyecto ha sido de 590 horas.

Capítulo 3

Estado del arte

En este capítulo se va a comentar el estado actual de las tecnologías que se han utilizado para desarrollar el proyecto.

3.1. Cliente

3.1.1. HTML5

HTML5 es la quinta versión del lenguaje de marcado para la elaboración de páginas web HTML que se publicó en octubre de 2014. Ésta incluye nuevos elementos, atributos, APIs y mejoras sobre versiones anteriores. Es una versión que todos los navegadores soportan, tanto los antiguos como los nuevos, incluyendo los navegadores de dispositivos móviles.

Una de las mejoras, sobre todo para los desarrolladores de páginas web, es que el código se ha hecho mucho más accesible y limpio gracias a la semántica y a ARIA (*Accessible Rich Internet Application*). Esto se debe a la incorporación de nuevos elementos [1] que son más descriptivos sobre su finalidad como <*header*>, <*footer*>, <*nav*>, <*section*> o <*article*>, entre otros muchos más.

También se pueden añadir atributos personalizados sin afectar a la visualización de los elementos para darles estilo posteriormente o poderlos manejar más fácilmente desde JavaScript.

[1] Para consultar las nuevas etiquetas: http://www.w3.org/TR/html-markup/elements.html

Entre las APIs de JavaScript más importantes están:

- API de Geolocalización para obtener la posición geográfica del usuario, siempre y cuando éste de permiso.
- API Drag&Drop para arrastrar o mover elementos dentro de la página web.
- API Local Storage: para almacenar localmente más información sin tener que utilizar cookies. Además que persiste en el sistema hasta que sea expresamente eliminada.
- API Application Caché para trabajar Off-Line: los recursos que sean necesarios se almacenan en la caché para que se pueda trabajar sin conexión a internet y a mayor velocidad ya que al estar guardados en local se cargan más rápido. A parte, se reduciría la carga del servidor ya que se actualizarían los recursos que han cambiado.
- API WebWorkers para ejecutar hilos JavaScript en paralelo y no afecten al rendimiento de la página, a diferencia de versiones anteriores en las que se debía esperar a que terminara la ejecución para poder operar en la web.
- API Server-Sent Events para recibir actualizaciones automáticas del servidor una vez establecida conexión con el cliente.
- API Multimedia para manejar audio y vídeo, accediendo al micrófono o cámara del dispositivo.

También hay otras APIs para hacer gráficos 2D, comprobar el estado de la batería, WebSockets para tener comunicación bidireccional en tiempo real entre páginas...

3.1.2. CSS3

CSS es un lenguaje de hojas de estilo para controlar o definir el aspecto de documentos HTML o XML. El objetivo de estas hojas es la de separar el contenido de la presentación ya sea en un documento aparte o en el mismo HTML. De esta manera, el código se hace más limpio y accesible para poder mantenerlo más fácilmente. La hoja de estilo CSS se crea una vez que ya está escrito el documento HTML. En ella se especifica una serie de reglas que definidas por unos selectores aplican una serie de estilos a los elementos correspondientes. Los estilos que se

pueden dar son sobre el color, el tamaño, la posición, la alineación de los textos o su fuente, los márgenes, etc...

A lo largo de la historia se han creado diferentes versiones de CSS. La primera fue CSS1 publicada en 1996 que ofrece funcionalidades de fuente, alineación de componentes, márgenes, presentación de listas... La segunda fue CSS2 publicada en 1998 que ofrece funcionalidades como el posicionamiento de las capas o estilos de tablas. Y la tercera y última es CSS3, aún en desarrollo, que ofrece mayor control sobre el estilo al separar en módulos bordes, fondos, colores, textos, interfaces, selectores por atributos, animaciones, transformaciones...

3.1.3. jQuery

jQuery es una librería de JavaScript creada por John Resig cuya filosofía es *"find something, manipulate it"*, que traducido del inglés significa "*encuentra algo, manipúlalo*". Es un software libre y de código abierto que puede ser usado en cualquier tipo de plataforma, libre o privada, con solo añadir en el documento HTML el script JavaScript que contiene el código jQuery. Esta biblioteca facilita el desarrollo de aplicaciones web tanto en tiempo como en espacio ya que ofrece una serie de funcionalidades en JavaScript que son compatibles con todos los navegadores web. De esta manera se reduce la cantidad de código y se programa a mayor velocidad dado que no hay discriminación entre los distintos navegadores, tarea que era muy tediosa hasta ahora. Además en el lado del cliente, el script se almacenará en caché la primera vez que se use permitiéndolo cargar desde el local más rápidamente, la siguiente vez que se utilice.

Su característica principal es que, gracias a su sencilla manipulación del árbol DOM, no necesitamos recargar la página web cada vez que deseemos cambiar un elemento, manipular las hojas de estilo, lanzar un evento o realizar una petición AJAX. jQuery interactúa con la página con una serie de funciones que permiten seleccionar elementos del DOM y poder manipularlos cambiado su estilo y dándoles efectos o animaciones. Además permite añadir o eliminar elementos, atributos y clases en tiempo real.

jQuery User Interface

jQuery UI es un complemento de la biblioteca de jQuery que permite interactuar con las interfaces de usuario añadiendo un conjunto de efectos, plug-ins y widgets para el desarrollo de

aplicaciones web enriquecidas. Su uso es igual que el de jQuery, basta con añadir el correspondiente script en el documento justo después de haber incluido la librería de jQuery.

Consta de cuatro módulos, que describiremos brevemente a continuación:

- **Núcleo**: que contiene las funcionalidades básicas para el resto de módulos.
- **Interacciones**: para añadir ciertos comportamientos a los elementos como poder redimensionarlo o arrastrarlo, ordenar una lista, seleccionar varios elementos con el ratón...
- **Widgets**: para añadir controles a los elementos como poder autocompletar mientras el usuario está escribiendo, crear ventanas de diálogo, menús anidados, una barra de progreso, mostrar calendarios...
- **Efectos**: para añadir transiciones animadas como hacer que un elemento se desvanezca, se deslice, cambie de tamaño o destaque su fondo, entre otros muchos más efectos.

Además desde la página web oficial de jQuery UI[2] podemos personalizar nuestro script eligiendo los componentes que necesitamos.

jQuery Mobile

jQuery Mobile es un plug-in de la biblioteca de jQuery diseñado para la creación de aplicaciones web en dispositivos táctiles compatibles con todos los smartphones y tablets, o sistemas operativos. Para poder utilizarlo, hay que actuar como con jQuery UI, primero indicar el script que contenga la librería jQuery y después el script correspondiente de jQuery Mobile.

Su característica principal es que aprovecha los nuevos atributos personalizados de HTML5. Por ejemplo, un documento HTML consta de tres apartados *div* que contienen los atributos *data-role* para indicar si es *header*, *footer* o *content* (cabecera, pie de página o contenido, respectivamente). Además, gracias al atributo *page*, se pueden incluir varias páginas dentro de un mismo documento HTML y que sólo se vea una. Se podrán acceder a esas otras páginas haciendo referencia a ellas cuando se declaren links.

[2] https://jqueryui.com/

3.1.4. Navegadores

Un navegador es un software que nos permite acceder a Internet, visualizar páginas web e interactuar con ellas. Pero también nos permite acceder a la información proporcionada por servidores web privados o archivos que estén almacenados en el ordenador.

El primer navegador fue creado en 1990 por Tim Berners-Lee, director del W3C (World Wide Web Consortium). Pero el primero en publicarse fue Erwise en 1992, y a partir de aquí se han ido desarrollando el resto de navegadores, desde Mosaic hasta Chrome (Google) pasando por Internet Explorer (Microsoft), Opera (Opera Software), Firefox (Mozilla) y Safari (Apple).

Cada uno de ellos tiene varias versiones, pero lo que nos interesa son cuáles de ellas son soportadas por HTML5 en su mayoría. Google Chrome soporta este nuevo lenguaje desde su versión 10, Firefox lo soporta desde la 28, Internet Explorer desde la 8, Opera desde 12.10, y Safari desde 5.1. Aunque, como es lógico, las últimas versiones de estos navegadores son las que mejor soportan HTML5. Existen páginas web que muestran el nivel de HTML5 que soporta el navegador que se está usando[3]. También nos interesa qué versiones soportan jQuery[4]: Internet Explorer desde la versión 6, Chrome, Firefox y Opera desde sus versiones más actuales, y Safari desde la 5.1. Además, está la página www.caniuse.com auspiciada por la W3C que te dice el soporte de cada funcionalidad para muchos navegadores.

Micronavegadores

Un micronavegador es un navegador web diseñado para ser utilizado por dispositivos móviles, como smartphones o tablets. Están optimizados para mostrar archivos de tamaño reducido en dispositivos de pantallas reducidas que no tienen mucha capacidad ni mucho ancho de banda.

Entre los micronavegadores más comunes están:

- Navegador Web por defecto para Android de Google, BlackBerry y Kindle.
- Internet Explorer Mobile de Microsoft.
- Safari para iOS de Apple.

[3] https://html5test.com/
[4] http://jquery.com/browser-support/

– Opera Mobile u Opera Mini de Opera Software ASA.

– Google Chrome.

– Firefox Mobile.

3.2. Servidor

Una vez comentadas las tecnologías utilizadas para implementar tanto la interfaz web del cliente como la del servidor, se pasa a explicar las que se han utilizado para implementar la lógica del servidor.

3.2.1. LibreGeoSocial

LibreGeoSocial [5] es un proyecto de código abierto desarrollado dentro del grupo LibreSoft[6] del grupo GSYC (Grupo de Sistemas y Comunicaciones, Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universidad Rey Juan Carlos) que consiste en una red social móvil con una interfaz de realidad aumentada.

Consta de dos componentes principales en su estructura:

- LibreGeoSocial es un framework desarrolado en Python y Django que facilita la creación de redes sociales móviles, incorporando la geolocalización.
- LibreGeoSocialApp es una aplicación Android que explota la funcionalidad de la anterior utilizando el GPS para localizar geográficamente a los nodos.

Como es un proyecto de código abierto, permite personalizar LibreGeoSocial en diferentes escenarios: turismo, juegos de realidad aumentada, etc...

3.2.2. Python

Python es un lenguaje de programación de alto nivel interpretado, que usa tipado dinámico y es multiparadigma ya que permite a los desarrolladores trabajar con diferentes estilos: programación orientada a objetos, programación imperativa y programación funcional. Creado a

[5] http://libregeosocial.morfeo-project.org
[6] http://www.libresoft.es

finales de los 80 pero publicado en 1991 por Guido van Rossum, posee una licencia de código abierto denominada Python Software Foundation License.

Hay tres grandes versiones de este lenguaje:

- **Python 1** fue la primera versión de python publicada en 1994. Las principales características de esta versión son las herramientas de la programación funcional y el soporte del uso de números complejos.
- **Python 2** fue la segunda gran versión lanzada en 2000. Incluye la generación de listas, idea tomada del lenguaje de programación funcional Haskell, y un sistema de recolección de basura. Otra novedad fue la unificación de los tipos y las clases, con lo que se logró un modelo de programación orientada a objetos.
- **Python 3** es la última versión de este lenguaje publicada en 2008, que rompe la compatibilidad hacia atrás y se encarga de corregir algunos defectos fundamentales en el diseño de lenguaje.

3.2.3. Django

Django es un framework de desarrollo web de código abierto escrito en Python cuyo objetivo es construir aplicaciones web más rápido y con menos código. Sigue la filosofía DRY (*Don't Repeat Yourself*) o *No Te Repitas* que promueve la reducción del código duplicado. De esta manera provee un alto nivel de abstracción de los patrones de desarrollo web más comunes.

Su arquitectura sigue el Modelo Vista Controlador en la que se separan los datos (*Modelo*), de la interfaz web del cliente (*Vista*) y del módulo que se encarga de gestionar los eventos y las comunicaciones (*Controlador*). Entre las bases de datos más utilizadas se encuentran PostgreSQL_Psycopg2, PostgreSQL, MySQL, SQLite3 u Oracle, aunque la más recomendada es PostgresSQL[7].

[7] http://www.postgresql.org/

Capítulo 4

Diseño del Sistema.

En este capítulo se tratará de explicar los aspectos más relevantes del diseño e implementación del presente Proyecto Fin de Carrera. Para ello se recordará el modelo del sistema proveniente del proyecto de Jorge Fernández, y se explicarán las diferencias en la implementación de su actualización.

4.1. Arquitectura general

El antiguo sistema constaba de tres módulos reflejados en la figura 4.1:

- El servidor que provee un servicio web desde donde el cliente (generalmente el administrador de la gymkhana) podrá manipular la base de datos, también alojada en esta máquina.
- Una interfaz de usuario en la que se mostrará la aplicación web destinada principalmente a los organizadores de la gymkhana para que puedan interactuar con la información alojada en la base de datos.
- Una aplicación Android para ser instalada en los dispositivos móviles que soporten dicho sistema operativo. Servirá a modo de interfaz en la cual los participantes de la gymkhana recibirán los enunciados de las pruebas, enviarán sus respuestas, mensajes, etc...

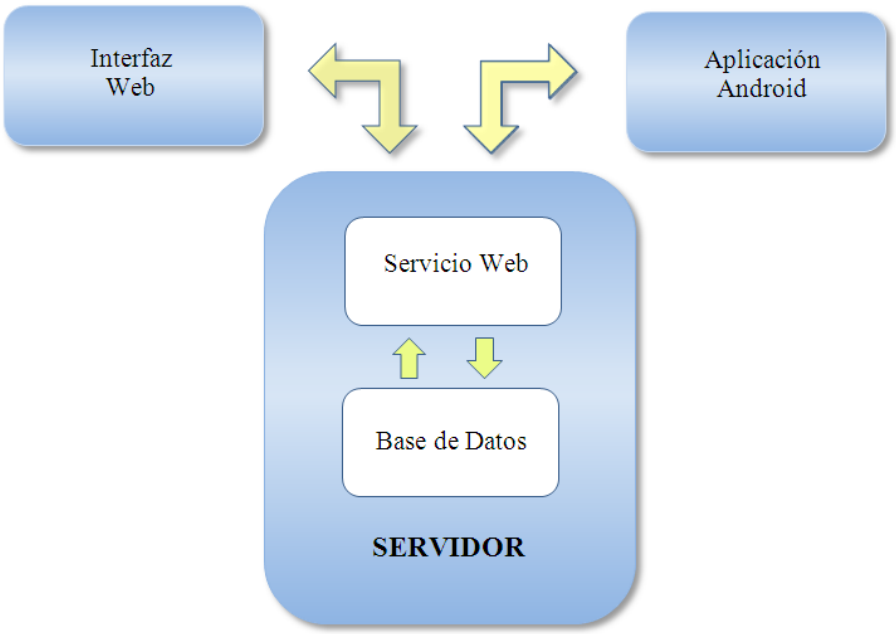


Figura 4.1: Esquema general del antiguo sistema.

Con la actualización implementada, el nuevo sistema constará de dos módulos fundamentales, tal y como aparece en la figura 4.2:

- El antiguo servidor con las nuevas modificaciones para poder manipular la base de datos acorde a la actualización.

- Una interfaz de usuario en la que se mostrarán dos aplicaciones web: una destinada a los organizadores de la gymkhana para que puedan interactuar con la información almacenada en la base de datos, y otra destinada a los participantes donde se presentará el juego.

Como se ha mencionado en el apartado de objetivos, la actualización realizada tiene como objetivo migrar la aplicación Android a una aplicación web, por lo que se tendría que suprimir el antiguo módulo que almacenaba la aplicación Android. Pero en lugar de eso, se ha decidido mantenerla por si en algún momento se desea utilizar.

Tras esta breve visión general de la estructura del sistema, se pasará a explicar cada uno de los módulos mencionados.

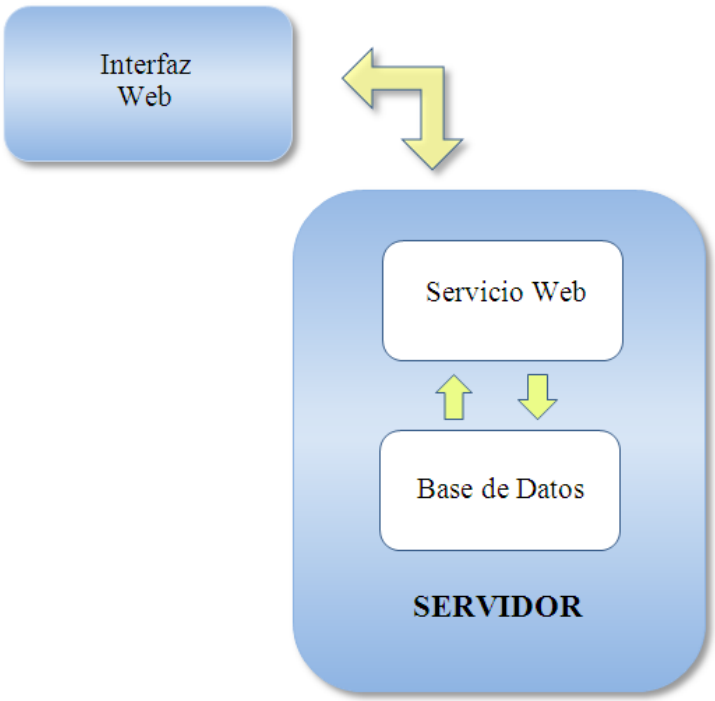


Figura 4.2: Esquema general tras la actualización del sistema.

4.2. Diseño e implementación del servidor

El servidor se implementó en Django básicamente porque es el framework utilizado en el proyecto LibreGeoSocial. De esta manera, el proyecto se podría integrar en él y aprovechar sus funcionalidades. El servidor está constituido fundamentalmente por dos partes: las funciones propias de dicho proceso y un modelo de datos.

4.2.1. Modelo de datos

El modelo de datos seguido toma como referencia las funcionalidades ya desarrolladas en el proyecto LibreGeoSocial, perteneciente a la Universidad Rey Juan Carlos. Como se mencionó en el apartado de Estado del Arte, LibreGeoSocial es una red social móvil clásica pero con la diferencia de que todos sus nodos están geolocalizados. Dicho proyecto sigue el modelo de datos representado en la figura 4.3.

Basándose en este modelo de datos, se añadieron dos nuevas entidades heredando de *Person*: *Manager* para referenciar a los organizadores o gestores de la gymkhana, y *TeamMember*

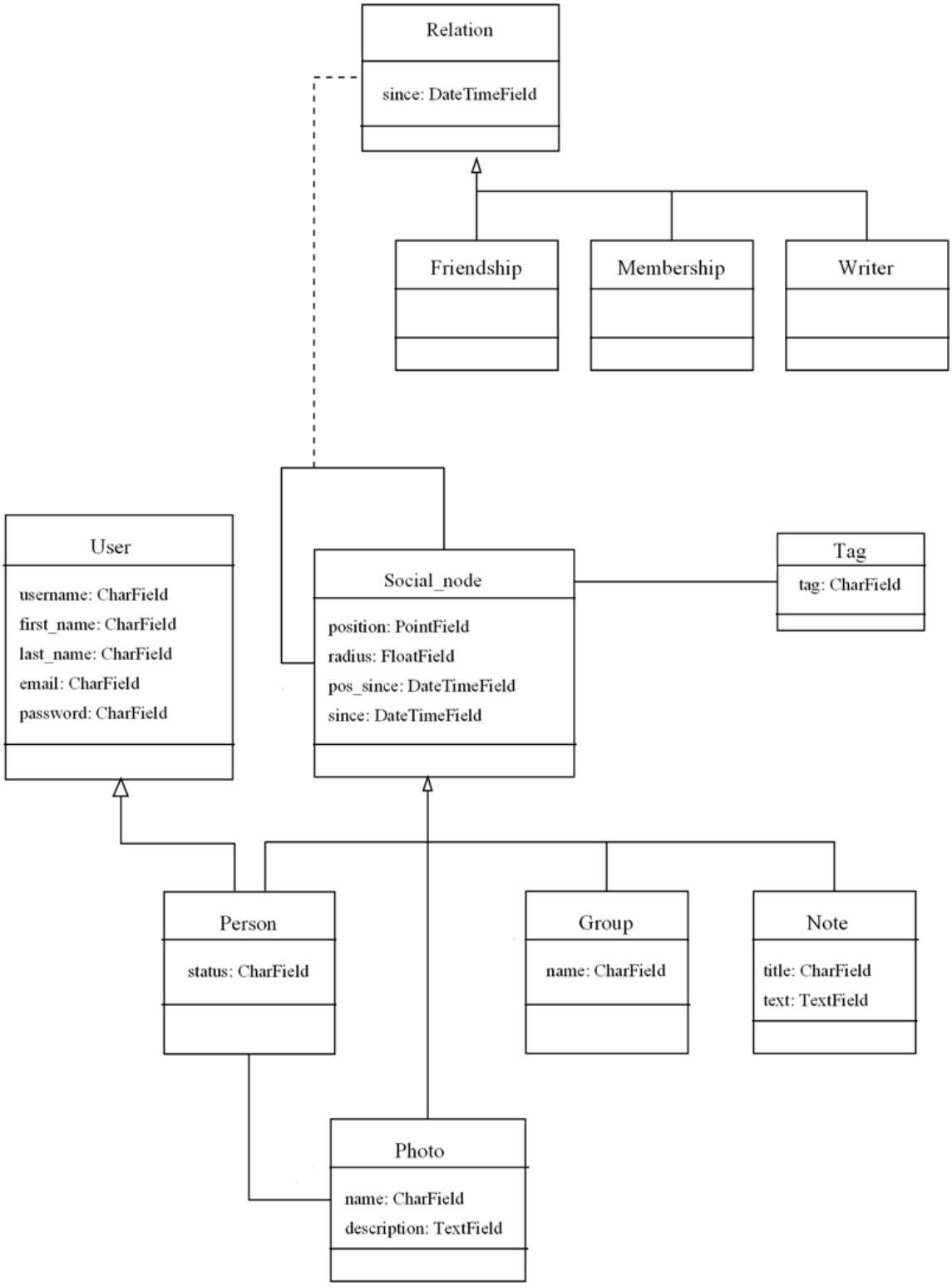


Figura 4.3: Diagrama UML de la base de datos del proyecto LibreGeoSocial.

para referenciar a los participantes de ésta. Los equipos (*Team*) estarán formados por varios participantes y contarán con un marcador (*Scoreboard*) con la puntuación obtenida. Otros elementos que se agregaron fueron: *Response* con las respuestas que envíe cada equipo, y *Message* con los mensajes que se intercambien entre el mánager y los equipos.

De este modo, el modelo de datos final que se utilizará es el representado en la figura 4.4.

Para implementar este modelo de datos se utilizó el gestor de base de datos PostgreSQL por cuatro razones: es el que utiliza LibreGeoSocial, es el que recomienda Django, es el que da soporte a la geolocalización y además es un software libre.

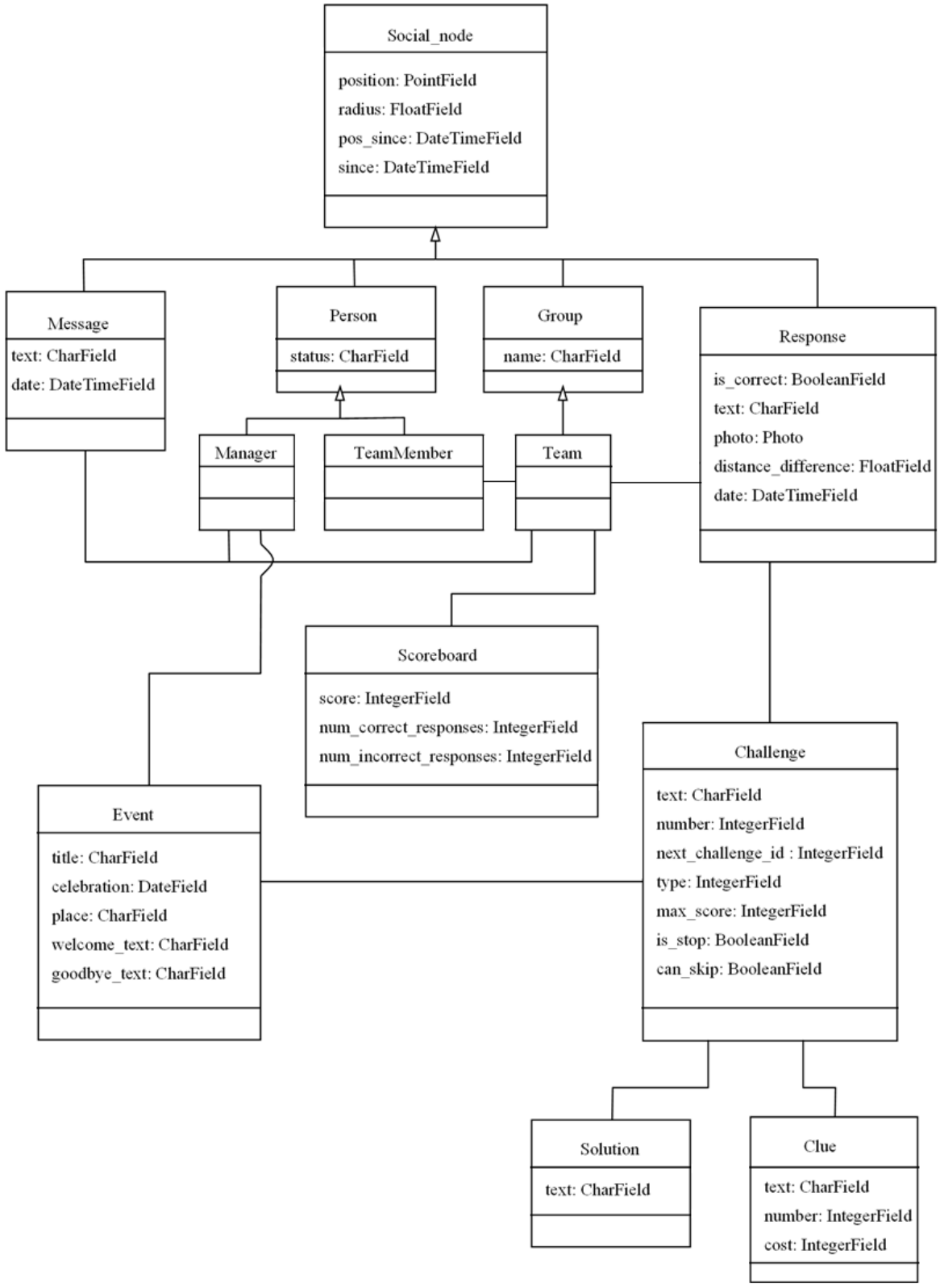


Figura 4.4: Diagrama UML del diseño realizado para la base de datos.

4.2.2. Servicio Web

Integración en LibreGeoSocial

Como se ha mencionado anteriormente, este proyecto es una aplicación integrada dentro de la red social LibreGeoSocial. Por lo tanto, con tan sólo copiar una serie de ficheros en el directorio adecuado, se consiguen utilizar las distintas funcionalidades de LibreGeoSocial para el desarrollo del proyecto.

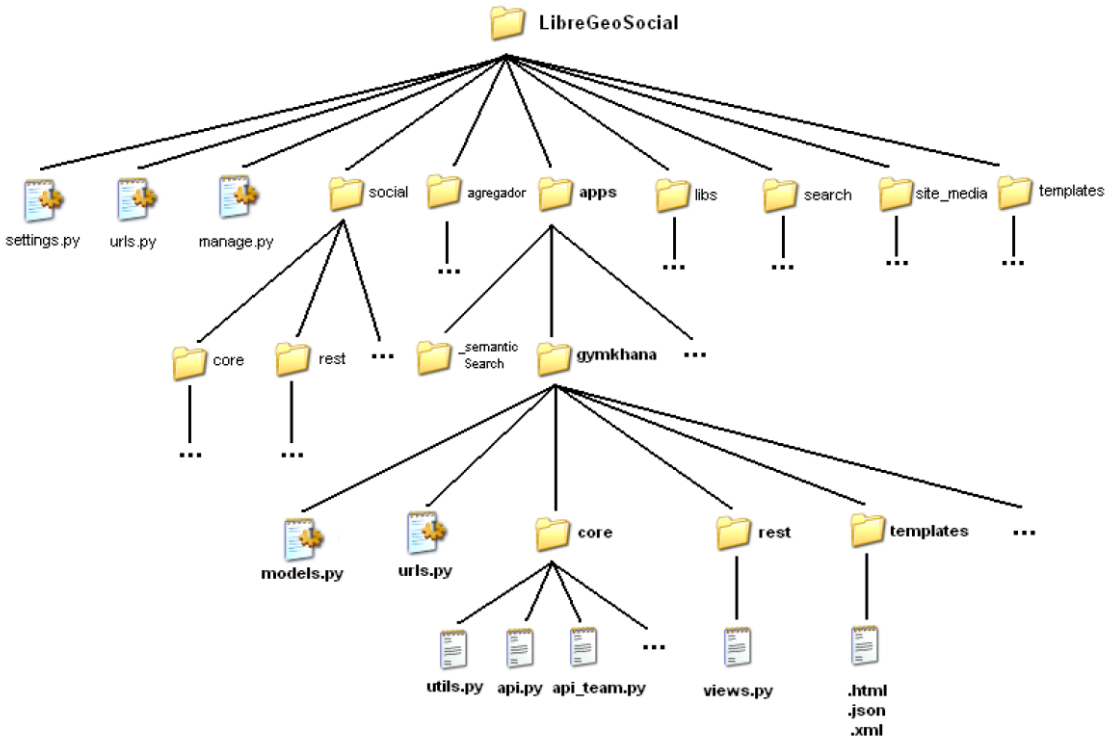


Figura 4.5: Diagrama del árbol de directorios de la integración del proyecto en el servidor de LibreGeoSocial.

Patron de Diseño MVC

Según se comentó en el Capítulo 3 Estado del Arte, el framework de desarrollo utilizado, Django, sigue el Modelo Vista Controlador aunque con ciertas particularidades que llevan a denominarlo Modelo Plantilla Vista (Model-Template-View). Para que estas particularidades no afectasen, se ha seguido el modelo de desarrollo utilizado en LibreGeoSocial. De esta forma se ha desarrollado por una parte una API REST y por otra parte una librería que se denomina

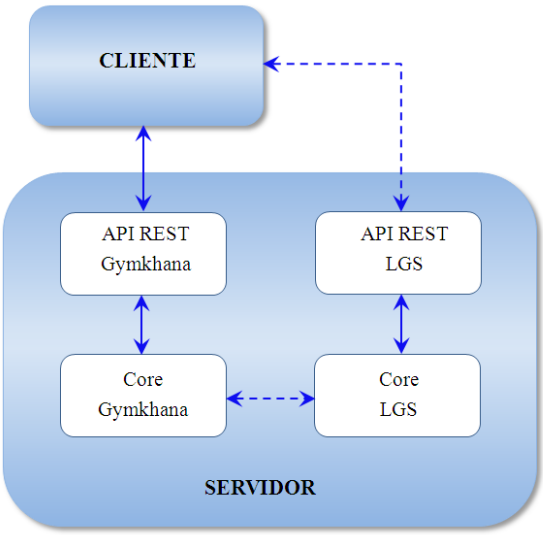


Figura 4.6: Diagrama de interacción entre cliente y servidor.

core.

En el core se realizarán las operaciones fundamentales de procesado, creación, modificaciones y destrucción sobre el modelo de elementos en la base de datos, mientras que el API REST actuará como interfaz de cara al cliente, por lo que quedará encargada de recibir y procesar las peticiones realizadas por el cliente, y tras comprobar que tales peticiones están bien formadas, utilizará el core para las operaciones que sean necesarias. Las respuestas se proporcionarán en forma de documento HTML que será generado a través de plantillas destinadas a ello.

Resumiendo lo anterior, la estructura del servidor quedaría según la figura 4.6 donde se puede observar que el cliente sólo conecta con las APIs, las cuales interaccionan con sus respectivos core según la petición recibida.

Por lo tanto, la arquitectura de este sistema sigue el patrón Modelo Plantilla Vista de la siguiente manera[2]:

- 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 que está definido en el archivo models.py, existente en cualquier proyecto Django.

- La vista estará compuesta por el API REST (fichero views.py de un proyecto Django) y

se encargará de escoger qué datos se mostrarán en la plantilla. Por lo tanto, esta parte se corresponderá con el core de la aplicación, ya que es la que se encarga de la manipulación del modelo de datos.

- La plantilla será la cual indique cómo mostrar los datos por pantalla, es decir, actúa como interfaz con el cliente.

URLs y Arquitectura REST

La manera que tiene Django de asociar el controlador con las diferentes URLs del sistema es mediante un fichero 'urls.py', en el cual se especifican los recursos que pueden ser accedidos del sistema, es decir, cada recurso cuenta con su propia URL. Éste sigue la filosofía REST (REpresentation State Transfer) que es un estilo arquitectural ampliamente extendido en servidores web estáticos, pero no tanto en servidores dinámicos. Entonces, las URLs no mostrarán ningún detalle de la implementación.

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).
- DELETE (permite eliminar un recurso conocida su URL, cambiando el estado y siendo una operación idempotente).

De todos esto métodos, se utilizarán sólo GET para realizar peticiones y POST para transferir datos como los de un formulario HTML.

Capítulo 5

Diseño e Implementación del Cliente.

Una vez estudiado el lado del servidor, se pasará a estudiar el diseño e implementación del cliente. Como ya se comentó anteriormente, esta actualización solamente consta de una interfaz web destinada al usuario, ya sea el organizador de la gymkhana o el participante en ella.

Esta nueva aplicación consiste en una serie de páginas HTML, accesibles desde cualquier navegador web que soporte HTML5 y jQuery, por medio de las cuales el usuario, tras previa autenticación, podrá realizar las acciones permitidas de acuerdo a su status, mánager o Jugador.

- El *mánager* o administrador de la gymkhana podrá manipular la base de datos para crear las gymkhanas.
- El *Jugador* solamente podrá acceder a las actividades propuestas por el mánager y ver sus resultados.

Todas estas páginas web están construidas con HTML5, CSS3, jQuery y Javascript para poder modificar la apariencia o añadir contenido sin tener que pasar por el servidor o tener que recargar la página.

5.1. Estructura de las páginas HTML.

En este apartado se explicará las principales diferencias en cuanto a la estructura de los documentos HTML. En concreto, las diferencias que existen en el código entre el antiguo proyecto y el presente proyecto, es decir, entre los diferentes estándares utilizados[7].

Para empezar, como se está trabajando con HTML5 ya no se tendrá que especificar qué estándar se está utilizando, por lo que se pasará de tener una declaración como la siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

a tener una declaración mucho más simple:

```
<!DOCTYPE html>
```

Toda página HTML tiene dos partes: la cabecera o *head* y el cuerpo o *body*.

Respecto a la cabecera, se han añadido las APIs de jQuery mediante enlaces a su servicio CDN (*Content Delivery Network*). En caso de fallo de conexión a dicho servicio, estas librerías se cargarán desde la versión hospedada en el servidor de la gymkhana para poder seguir utilizando sus funcionalidades, tal y como muestra el siguiente código:

```
<script type="text/javascript">
 if(typeof jQuery=='undefined'){
   document.write(unescape("<script src='jquery.mobile/
      jquery-1.9.1.min.js' type='text/javascript'></scri"+"pt>
      "));
   document.write(unescape("<script src='jquery.mobile/jquery.
      mobile-1.3.2.min.js' type='text/javascript'></scri"+"pt>
      "));
   document.write(unescape("<link rel='stylesheet' href='
      jquery.mobile/jquery.mobile-1.3.2.min.css' /"+">"));
 }
</script>
```

Pero lo cambios más sustanciales se han realizado en la parte del cuerpo. Como se mencionó en el Capítulo 3 Estado del Arte, HTML5 incluye nuevas etiquetas predeterminadas más descriptivas como *header* y *footer* para indicar la cabecera y el pie de página, respectivamente. Por lo que en lugar de crear una sección con la etiqueta *div* e identificador *"header"* o *"footer"*

como se tenía antes:

```
<div id="footer">
 Contenido del pie de página.
</div>
```

Se pasa a tener un código mucho más descriptivo y visual:

```
<footer>
 Contenido del pie de página.
</footer>
```

Lo mismo ocurre para definir secciones cuya temáticas no tienen etiquetas predefinidas que expliquen sus significados, por lo que se crearán etiquetas propias. Para ello, en lugar de especificar un *div* con una clase o identificador:

```
<div class="ranking">
 Contenido específico de la etiqueta.
</div>
```

Se creará una etiqueta propia llamada *ranking* que sustituya al *div* de la misma manera que los casos anteriores.

```
<ranking>
 Contenido específico de la etiqueta.
</ranking>
```

En los siguientes apartados, se explicarán las diferencias entre las interfaces web del administrador y la del jugador, así como las funcionalidades de cada uno.

5.2. Administrador.

En este subapartado se explicará la implementación de las funciones que puede realizar el administrador sobre la base de datos del servidor, y las tecnologías utilizadas para desarrollarlo.

Pero primero se comentará la visualización de la interfaz web mostrada al usuario para poder operar con el servidor.

5.2.1. Interfaz web.

La interfaz web del mánager consta de dos versiones: una para el ordenador y otra para dispositivos móviles. Principalmente está diseñada e implementada para que se pueda acceder a ella desde un ordenador, debido a que resulta mucho más cómodo introducir la información referente a las gymkhanas desde un teclado grande que desde un dispositivo móvil con una pantalla pequeña. La versión para dispositivos móviles se ha desarrollado con el objetivo de que el organizador disponga de mayor movilidad. Es cierto que no necesita desplazarse hacia el lugar de los equipos gracias al servicio de mensajería, pero sí que puede requerir de ella para crear los retos de la gymkhana, o realizar pequeños cambios.

Dado que las pantallas de los dispositivos móviles no son muy grandes, la visualización de toda la página puede resultar incómoda. Por eso es necesario optimizar el espacio, y para ello se ha decidido que la visualización de la aplicación web sea diferente entre los dos tipos de terminales que se van a utilizar. Se podrá realizar dicha distinción detectando el tipo de navegador que se está utilizando gracias a Javascript. Para ello se tendrá que buscar cuál es el navegador utilizado por el usuario, y eso se consigue accediendo a la información del User Agent o Agente de Usuario. Una vez encontrada el tipo de navegador, jQuery se encargará de mostrar la aplicación web correspondiente, suprimir la que no es y aplicar el correspondiente estilo, tal y como aparece en el código 5.1.

```
<script type="text/javascript">

 $(document).ready(function(){

   var dispositivo = navigator.userAgent.toLowerCase();

   if( dispositivo.search(/android|webos|iphone|ipad|ipod|
      blackberry|iemobile|opera mini/i) > -1 ){
```

```
      $('.computer').remove();
      $('.mobile').show();
      $("#linkestilo").attr("href", "/gymkhana/templates/styleM.
         css");

    }else{
      $('.mobile').remove();
      $('.computer').show();
    }
  });

</script>
```

Código 5.1: Código Javascript que diferencia entre dispositivos.

Si se está accediendo desde un dispositivo móvil, pero se quiere visualizar la aplicación como si se tratara de un ordenador, siempre se puede utilizar la opción del navegador web que nos permite mostrar la página como si estuviéramos accediendo desde un ordenador de escritorio.

En las páginas web de la interfaz del administrador se pueden observar dos partes, tal y como se muestra en la figura 5.1: un menú y el contenido de la sección correspondiente (lista de gymkhanas disponibles, creación y visualización de gymkhanas, ranking...). Como se acaba de mencionar, en los dispositivos móviles es necesario optimizar el espacio para una cómoda visualización de la página. Por lo que se ha decidido que en esta versión móvil de la interfaz del administrador, el menú quede en segundo plano y únicamente se muestre cuando el usuario lo desee con sólo pulsar el botón que está redondeado en verde en la figura 5.2. La visualización de la aplicación web para el administrador quedaría como muestran las figuras 5.1 y 5.2.

Una de las principales diferencias en cuanto a la visualización de la aplicación en un ordenador respecto al antiguo proyecto, es que ahora se puede ver el menú desde cualquier punto de la página web, es decir, éste permanecerá estático según el usuario se mueva por la página.

La imagen del icono redondeado en verde de la figura 5.2 funciona a modo de botón, y

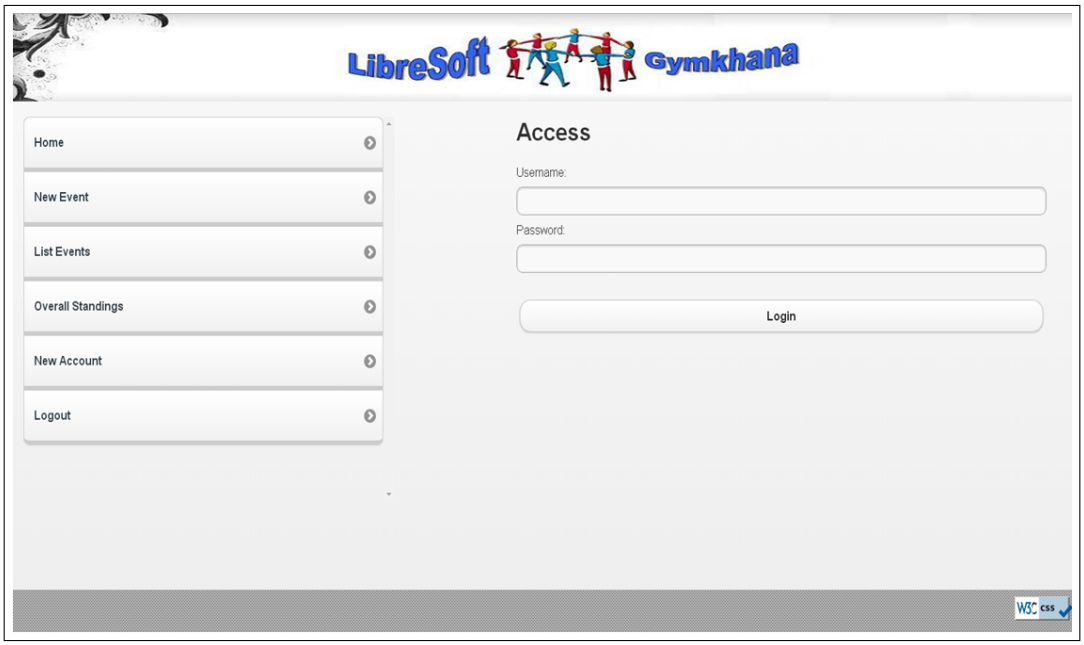


Figura 5.1: Versión web para ordenadores.

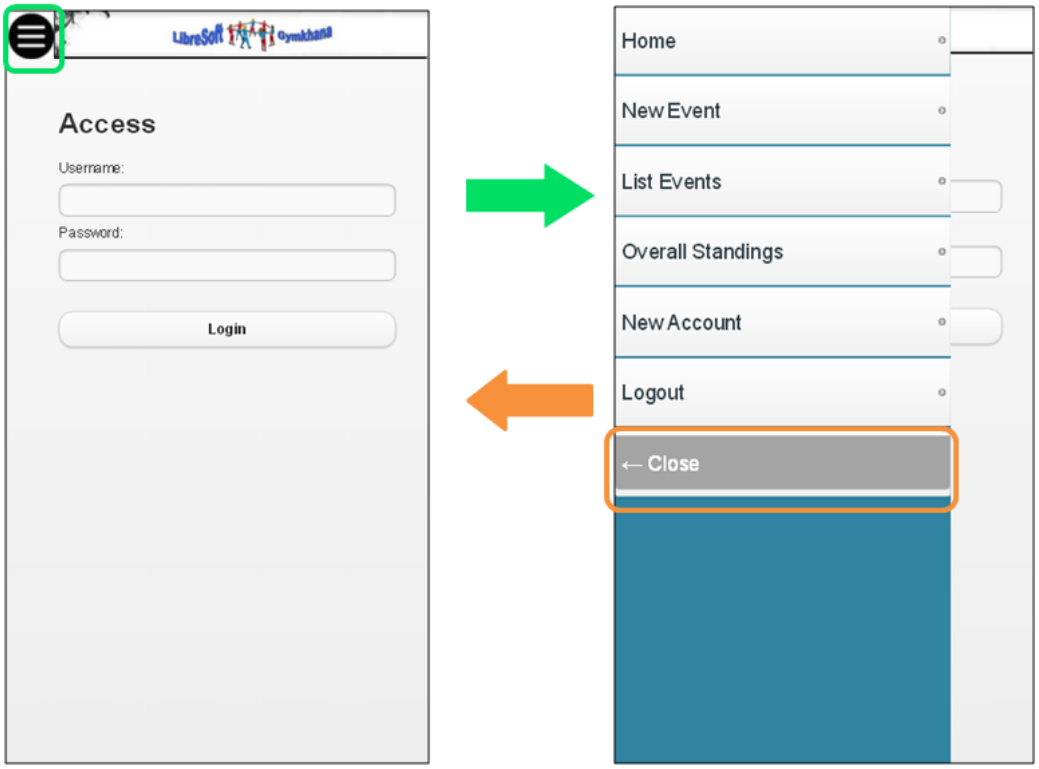


Figura 5.2: Versión web para móviles.

cuando se pulse en él, se deslizará el menú mostrando todas las secciones a las que se puede acceder. Para realizar este movimiento, se ha definido la imagen con una clase *"toggle-slide-left"* a la que se asociará un manejador de eventos para que cuando se haga click en ella, se ejecute una función encargada de mostrar el menú. Y si se quiere ocultarlo, se pulsará en el botón *"Close"* que aparece en la lista de opciones del menú, que al igual que la imagen, se le ha definido una clase *"close-menu"* asociada a un manejador de eventos. El código javascript con el que se consigue dicha operación es el que aparece en el código 5.2:

```
$(".toggle-slide-left").click(function(){
  $("nav").css('left', '0');
});

$(".close-menu").click(function(){
  $("nav").css('left', '-80%');
});
```

Código 5.2: Código Javascript que maneja la visibilidad del menú.

Siendo *"nav"* el elemento del árbol DOM que contiene la lista con los enlaces de las páginas a las que puede acceder el usuario:

```
<nav class="slide-left">

 <ul data-role="listview"  data-inset="true">
   <li><a href="/gymkhana/" data-ajax="false">Home</a></li>
   <li><a href="/gymkhana/event/create/" data-ajax="false">New
       Event</a></li>
   <li><a href="/gymkhana/event/list/" data-ajax="false">List
      Events</a></li>
   {% if event  %}
   <li><a href="/gymkhana/event/{{ event.id }}/show/"
      data-ajax="false">Show Event</a></li>
   <li><a href="/gymkhana/event/{{ event.id }}/monitoring/"
      data-ajax="false">Monitoring</a></li>
```

```
    <li><a href="/gymkhana/event/{{ event.id }}/message/create/
        " data-ajax="false">New Message</a></li>
    {% endif %}
    <li><a href="/gymkhana/overall_standings/show/?many=all"
        data-ajax="false">Overall Standings</a></li>
    <li><a href="/gymkhana/user/create/" data-ajax="false">New
        Account</a></li>
    <li><a href="/gymkhana/user/logout/" data-ajax="false">
        Logout</a></li>
    <li><a class="close-menu">&larr; Close</a></li>
  </ul>

</nav>
```

También hay otras posibilidades para que se muestre/oculte el menú como definir una nueva clase en la hoja de estilo:

```
nav.show-menu{
    left: 0;
}
```

Y en lugar de utilizar la función `.css()` para cambiar el estilo, se podría añadir esta nueva clase al elemento *"nav"* de dos maneras:

- Una forma sería la de usar la función `.toggleClass()` en ambos manejadores de eventos.

  ```
  $("nav").toggleClass("show-menu");
  ```

  Esta función `.toggleClass()` se encarga de añadir o eliminar la clase *"show-menu"* al elemento *"nav"*.

- La otra opción es usar las funciones `.addClass()` y `.removeClass()` para añadir o

eliminar la clase *"show-menu"* al elemento *"nav"* y así poder mostrar u ocultar el menú, respectivamente.

```
$("nav").addClass("show-menu");
$("nav").removeClass("show-menu");
```

Finalmente, se ha decidido usar la función `.css()` porque así se reduce el código de la hoja de estilo y queda más clara la función que se quiere llevar a cabo.

5.2.2. Funcionalidades.

En cuanto a las funciones que puede realizar el administrador respecto a la base de datos, es importante tener en cuenta que para poder manipular tal base, deberá de autenticarse con un usuario que tenga categoría de mánager, por el contrario no podrá realizar ninguna modificación y sólo podrá ver información sobre los eventos. Para ello se mostrará una página web como la de la figura 5.3.

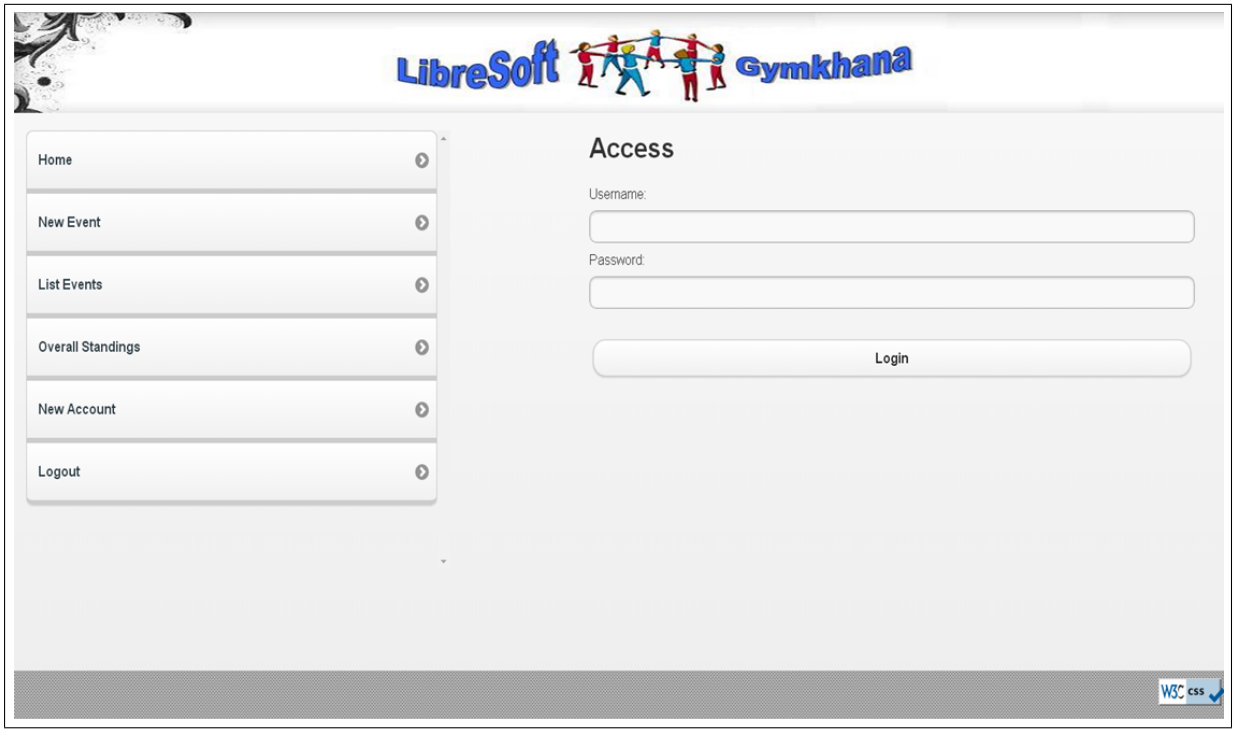


Figura 5.3: Página web para la autenticación de usuarios.

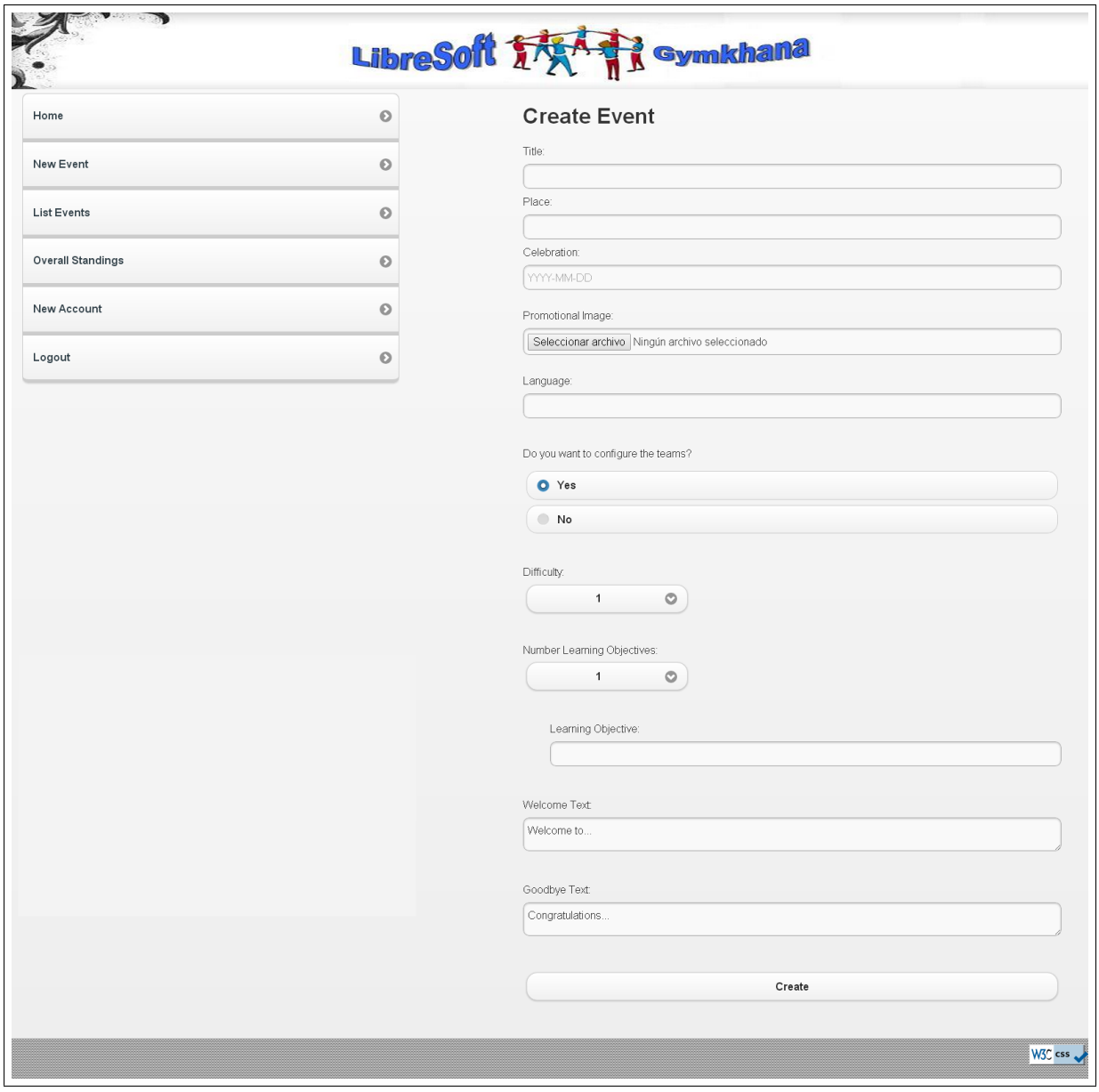


Figura 5.4: Página web para la creación de gymkhanas.

Las acciones que puede realizar el administrador son:

Crear gymkhanas.

Para crear un evento el usuario deberá rellenar un formulario HTML como el de la figura 5.4. En este punto, antes de que se envíe dicho formulario al servidor para que procese la información, se realizarán dos comprobaciones de los datos introducidos, debido a que la primera que se realice no es soportada por todos los navegadores. Ésta será la de comprobar si se han rellenado los campos requeridos gracias al nuevo atributo de la etiqueta *input* de HTML *required*, o de si se han introducido correctamente gracias a otro nuevo atributo *pattern*.

Si se pulsa enviar sin haber rellenado los campos requeridos, el navegador avisará al usuario con un mensaje como el que aparece en la figura 5.5.

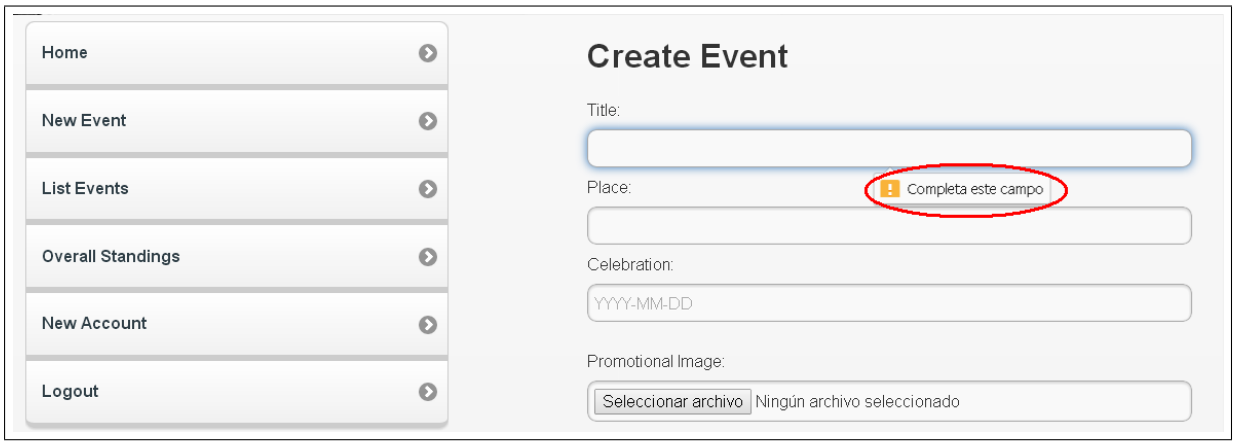


Figura 5.5: Alerta campo sin rellenar.

Atendiendo al otro atributo, *pattern*, si no se introduce el texto con el formato o patrón requerido, el navegador avisará según la figura 5.6.

La segunda comprobación se realizará con Javascript y lanzará una alerta, como la de la figura 5.7, informando del tipo de error encontrado: no se ha rellenado el campo, caracteres no soportados, la forma de la fecha introducida no es correcta...

Esta segunda comprobación es más exhaustiva, se comprueban que los caracteres introducidos sean los correctos ya que algunos de ellos pueden causar errores en el tratamiento de URLs, como los acentuados, la ñ, &...

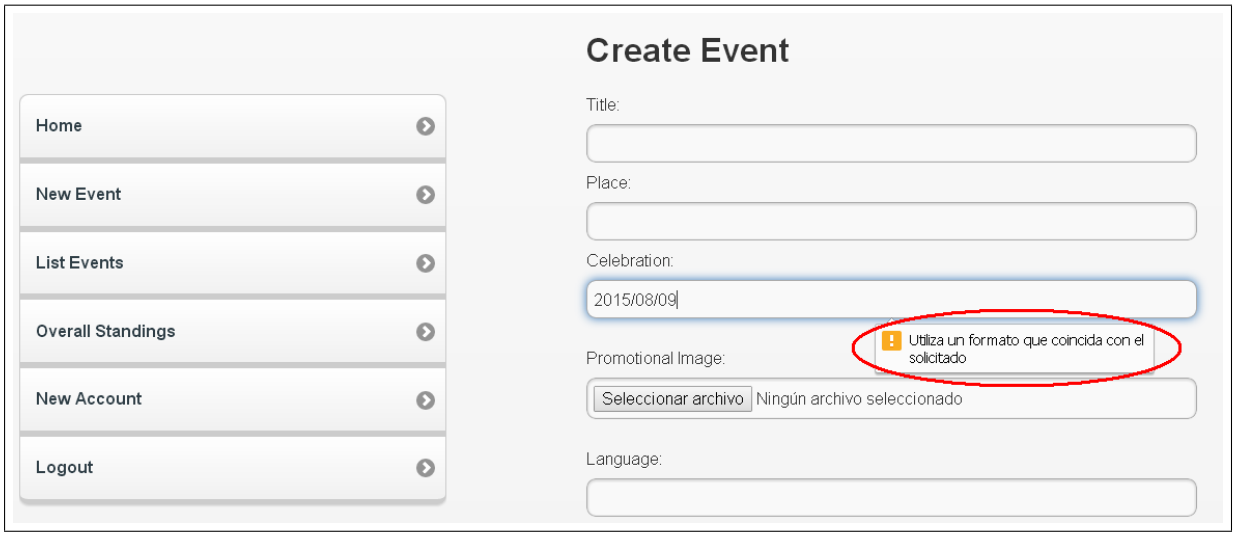


Figura 5.6: Alerta formato introducido es incorrecto.

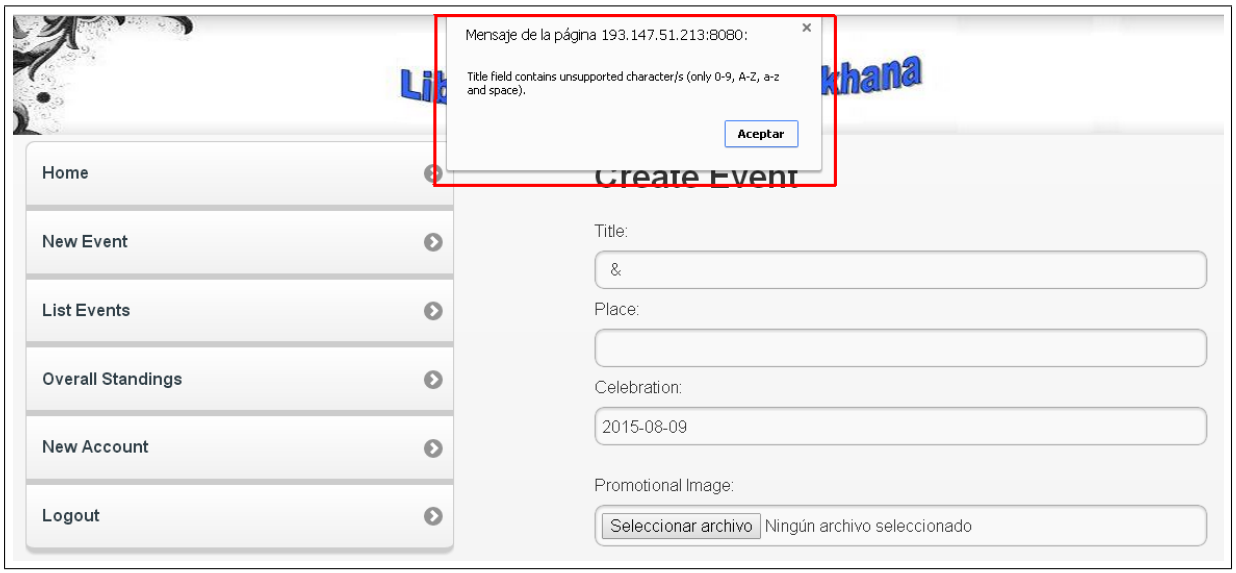


Figura 5.7: Alerta del navegador tras realizar la segunda comprobación del texto introducido.

Una vez que el evento ya ha sido creado correctamente, se pasará a otra página web que mostrará toda la información relativa a la gymkhana, como en la figura 5.8. En esta página aparecerán también otras secciones para insertar retos, visualizar la información de cada reto, añadir equipos a la gymkhana y listar los equipos con su primer reto a realizar.

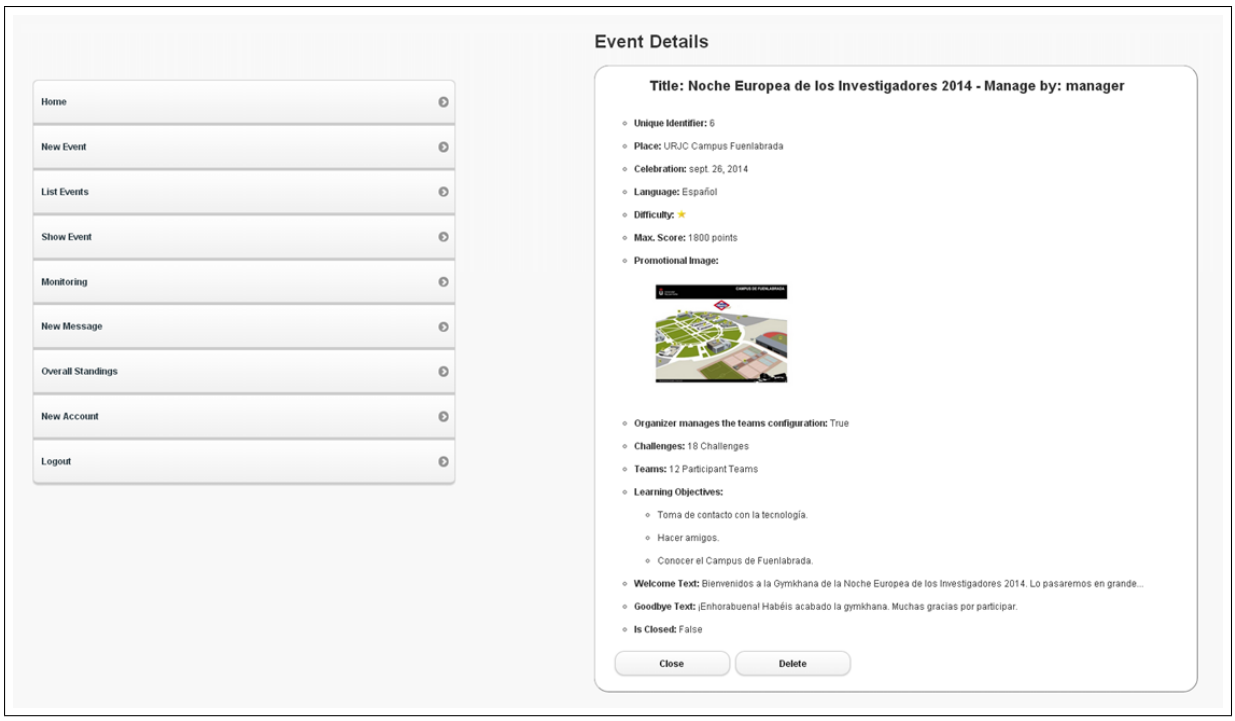


Figura 5.8: Información del evento.

Existen tres tipos de retos que se pueden crear según el tipo de respuesta:

- **Textual**: En la que se formulará una pregunta que los equipos deben responder textualmente.
- **Fotográfica**: En la que la respuesta de los equipos debe ser una fotografía.
- **Localización**: En la que los equipos deben dirigirse a un lugar determinado con la ayuda de un mapa que muestre la posición de destino y del equipo.

Cada uno de estos tipos de respuestas tiene sus propias características, por ejemplo, la respuesta textual puede tener diferentes soluciones pero las de fotografía y geolocalización no, o también que la textual y la de fotografía no necesitan unas coordenadas para resolverse. Por

eso según la opción que elija el usuario se mostrarán los diferentes campos que deben ser rellenados, tal y como se muestra en la figura 5.9. Estos cambios en la página web ser realizarán sin tener que recargar la página gracias a jQuery. Cuando el usuario haga click en una de las opciones, se lanzará internamente un evento controlado por jQuery que se encargará de mostrar los campos correspondientes y ocultar los que no interesan.

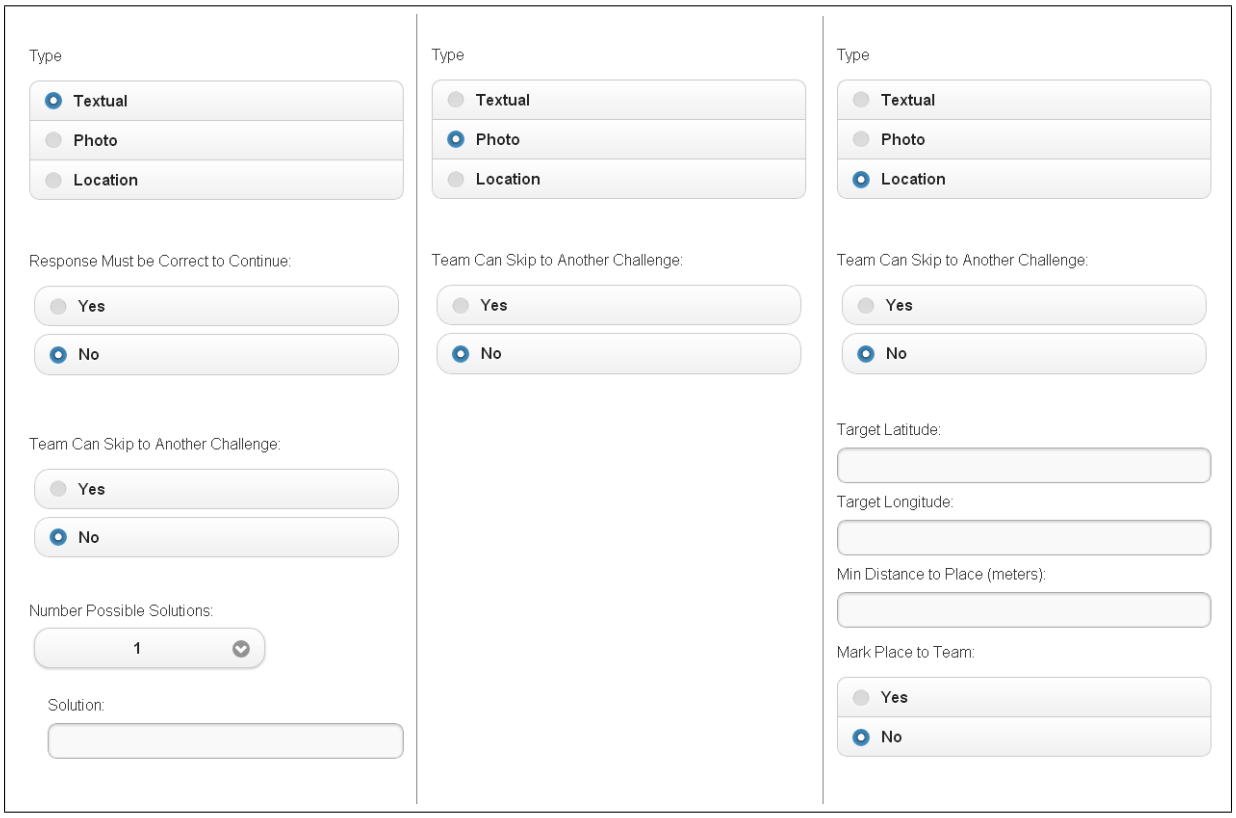


Figura 5.9: Campos de cada tipo de reto.

Concretamente, cuando el usuario presione una de las opciones que aparecen en el apartado *Type* de la figura 5.9, jQuery realizará la función correspondiente de entre las que se muestran en el código 5.3.

```
$(document).ready(function(){

 var correct = $('#correct');
 var solutions = $('#solutions');
 var coords = $('#coords');
```

```
  $("#textual").click(function(){
    coords.hide();
    correct.show();
    solutions.show();
  });
  $("#Ophoto").click(function(){
    coords.hide();
    correct.hide();
    solutions.hide();
  });
  $("#location").click(function(){
    correct.hide();
    solutions.hide();
    coords.show();
  });

});
```

Código 5.3: Código Javascript que cambia los campos mostrados del formulario en cuanto al tipo de respuesta.

También se pueden proporcionar pistas si el organizador de la gymkhana lo cree conveniente. Según el número de pistas que el mánager desee introducir, o si desea proporcionar varios tipos de soluciones para una respuesta textual, con ayuda de Javascript se irán añadiendo al documento HTML los campos de entrada donde insertar el texto. Y todo ello sin tener que pasar por el servidor. Además el organizador de la gymkhana tiene la opción de elegir si el reto puede ser saltado, si debe ser correcto para poder continuar o la posición que ocupa el reto dentro de la gymkhana.

Dado que todos los equipos realizarán las mismas pruebas, el recorrido se ha implementado en forma de lista enlazada como muestra la figura 5.10. De esta manera, cuando un equipo haya terminado todos los retos, el sistema detectará que ya no hay más retos por realizar y le llevará a una página de salida con los resultados obtenidos.

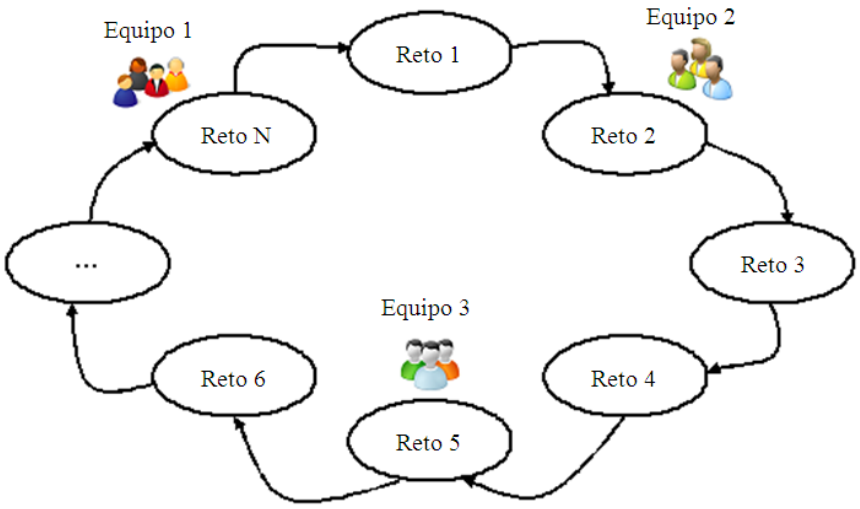


Figura 5.10: *Ejemplo del escenario de una gymkhana recorrida por varios equipos, con las pruebas enlazadas circularmente*[1].

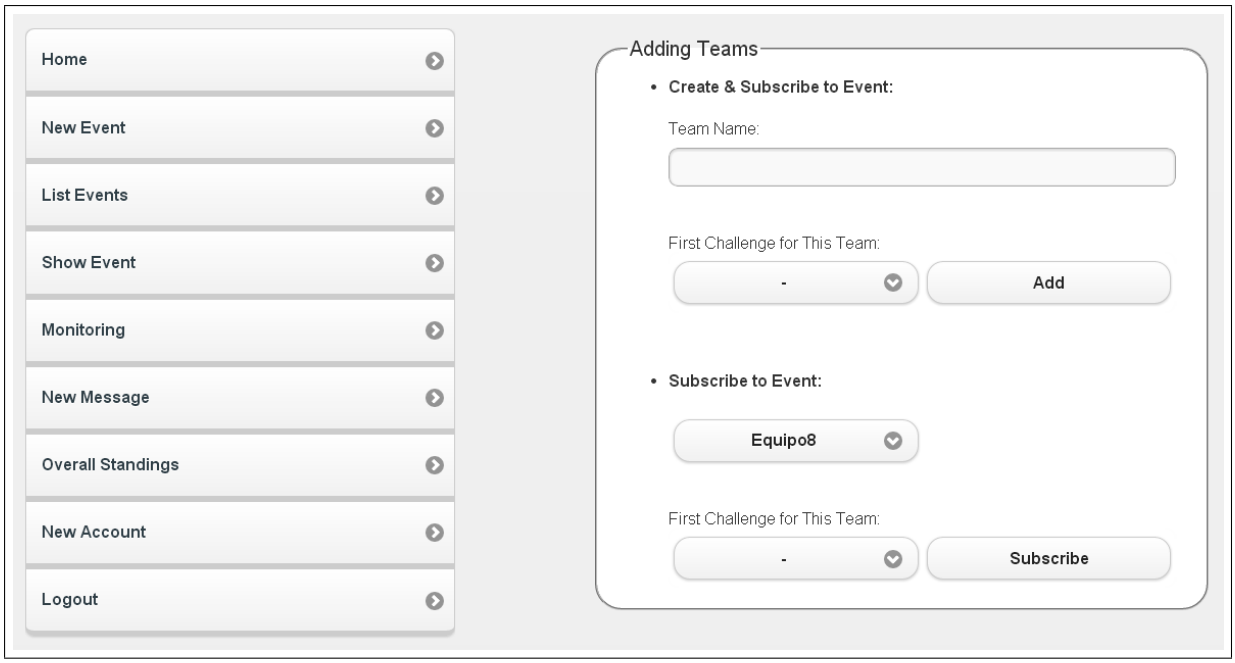


Figura 5.11: Inserción de los equipos.

Finalmente, antes de enviar al servidor el reto creado, se realizarán comprobaciones en los datos introducidos como los del formulario de la creación de gymkhanas comentadas unas hojas antes.

Por otro lado, habrá otra sección, como la que aparece en la figura 5.11, en la que el organizador pueda añadir los equipos que puedan participar y el reto en el que deben comenzar. De esta manera, ningún equipo coincidirá durante la gymkhana.

Monitorización de las gymkhanas.

Una de las ventajas de este sistema respecto al juego tradicional es que el mánager de la gymkhana no necesita desplazarse para controlar el transcurso del evento, ya que dispone de una página web desde la cual puede controlar a los equipos, sus respuestas y los mensajes que éstos envíen. Esta página está estructurada en cuatro secciones:

- Un ranking con la puntuación que consiguen los equipos según si sus respuestas con correctas o incorrectas (figura 5.12).
- Un mapa de GoogleMaps que muestra la localización geográfica de los equipos participantes (figura 5.12).
- Un listado de todas las respuestas que envían los equipos, ya sean correctas o incorrectas. En esta sección, se muestran todos los datos referentes al reto y dos botones por si el mánager desea borrar la respuesta o si desea cambiar el estado de ésta de correcta a incorrecta, o viceversa (figura 5.13).
- Un listado de todos los mensajes intercambiados entre los equipos y el mánager empleando el servicio de mensajería que ofrece la aplicación (figura 5.14).

Para realizar algunas de las secciones anteriores se necesita información alojada en el servidor, para ello se le enviarán peticiones AJAX en segundo plano solicitando información determinada. Cuando se reciba la respuesta del servidor, se procesará dicha información, se comparará con la que ya se dispone y si hay alguna diferencia, se alertará al usuario de tal novedad y se añadirá a la página sin tener que recargarla. De esta manera, el mánager podrá ir viendo las respuestas que los usuarios van enviando o si tiene algún mensaje nuevo que leer.

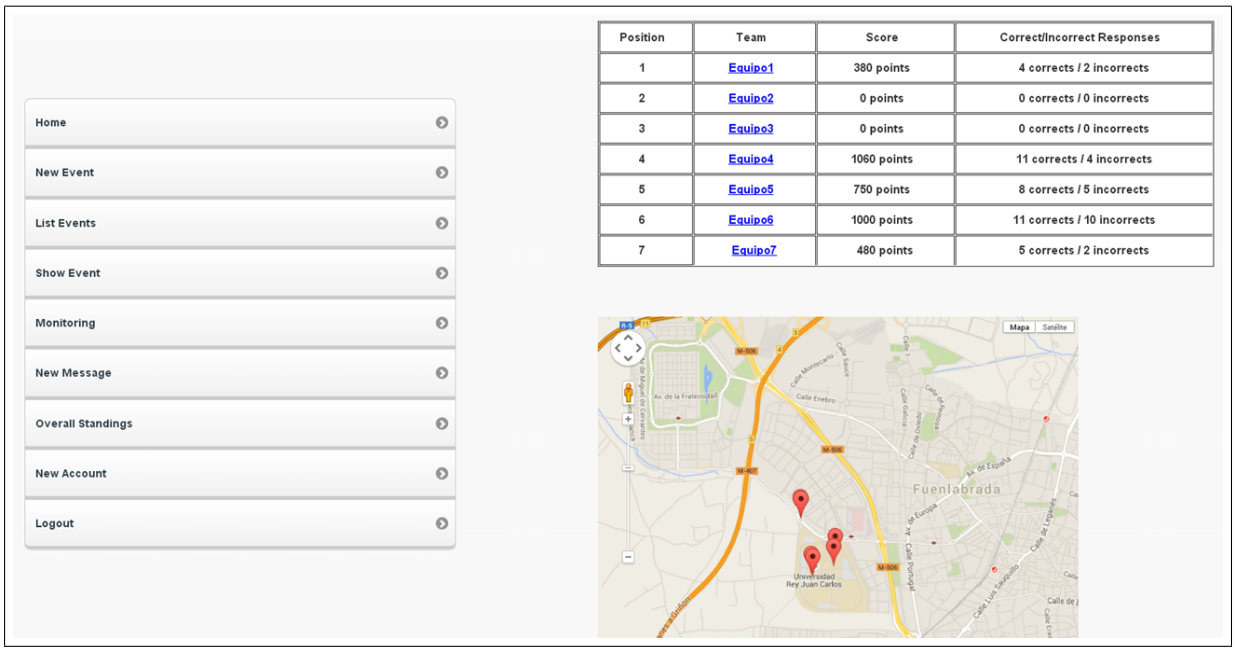


Figura 5.12: Sección de la página web con el ranking y el mapa.

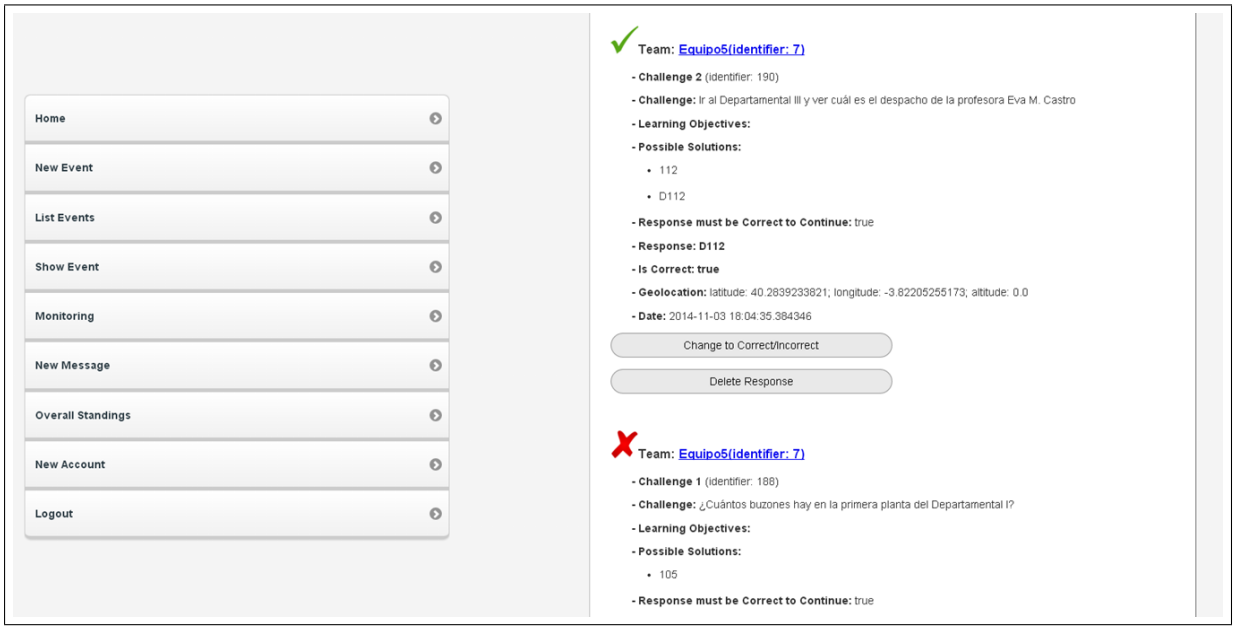


Figura 5.13: Sección de la página web que muestra las respuestas de los equipos.

En la última sección de la página, el mánager puede acceder el servicio de mensajería de dos maneras, tal y como se puede observar en la figura 5.14: una seleccionando el enlace del menú que se llama *"Nuevo Mensaje"* o como respuesta a un mensaje que aparece en el listado de mensajes recibidos por los equipos. Cualquiera de estos enlaces le llevará a otra página web donde podrá seleccionar al destinatario o destinatarios y el cuerpo del mensaje, como aparece en la figura 5.15.

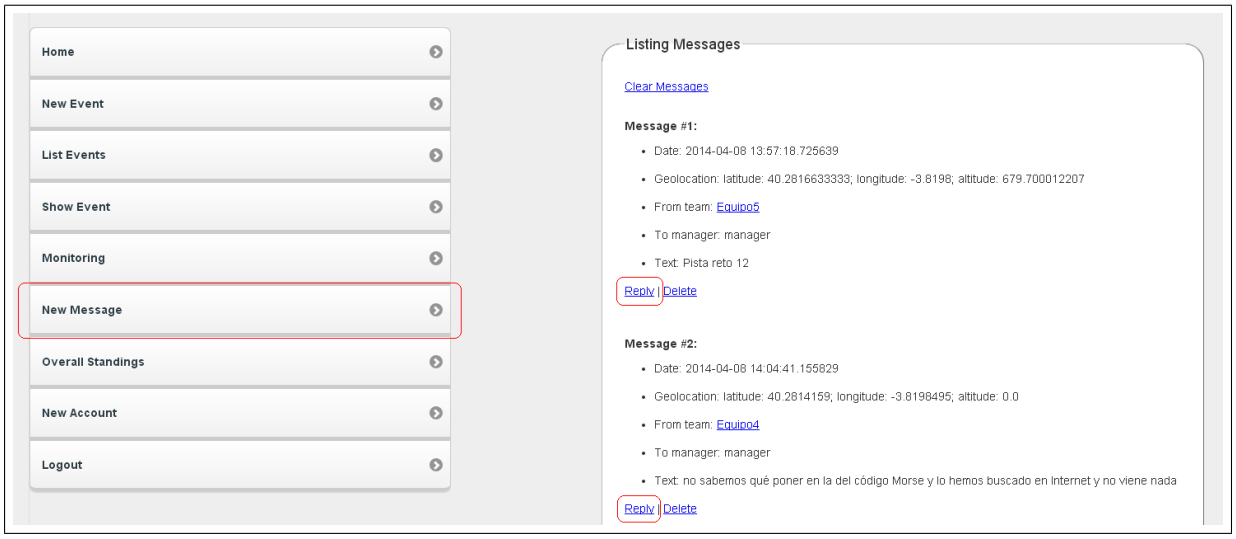


Figura 5.14: Formas de acceder a la página web donde se enviarán los mensajes.

Figura 5.15: Creación de mensajes.

En cuanto a la sección donde aparece el mapa, se han realizado algunas modificaciones ya que antes para mostrarlo se utilizaba la versión 2 de la API JavaScript de Google Maps. Pero esta versión ya ha quedado obsoleta y se necesitaba actualizar el código para utilizar la versión 3 que actualmente es la oficial[12].

La principal diferencia entre ambas APIs es que ahora no es necesario que se incluya una clave para hacer uso de la librería, aunque es recomendable que se utilice. Otras diferencias se encuentran en la forma de crear el mapa, los marcadores y las ventanas de información de cada uno, por ejemplo. Ahora para crear el mapa se utilizará el siguiente código:

```
var latitud = 40.282313;
var longitud = -3.819727;
var zoom = 5;

var mapCenter = new google.maps.LatLng(latitud, longitud);
var mapOptions = {
    zoom: zoom,
    center: mapCenter,
    scaleControl: true,
    overviewMapControlOptions:{opened:true},
    mapTypeId: google.maps.MapTypeId.ROADMAP
};

map = new google.maps.Map(document.getElementById("map"),
   mapOptions );
```

Como se puede ver, primero se establecen las opciones de configuración del mapa tales como el zoom, el centro, el tipo de mapa... Después, se crea el mapa utilizando `google.maps.Map`, al que se le pasan los parámetros recientemente descritos y el nodo del árbol DOM del documento HTML donde se va a insertar dicho mapa.

Una vez se hayan recibido la lista de equipos inscritos desde el servidor y el mapa esté dibujado, por cada equipo se insertará un marcador en las correspondientes coordenadas, de la

siguiente manera.

```
var properties = {
    position: point,
    map: map,
};

var marker = new google.maps.Marker(properties);
overlays[overlays.length] = marker;

var infowindow = new google.maps.InfoWindow({
    content: popuphtml,
});

google.maps.event.addListener(marker, 'click', function() {
    infowindow.open(map,marker);
});
```

Lo primero que se realiza es crear el marcador con `google.maps.Marker()` al que se le pasan una serie de propiedades como el punto donde sebe situarse el marcador y el mapa donde se va a insertar. A continuación, el marcador se guardará en un array, previamente declarado, para poder borrarlo más tarde cuando se vuelva a recibir la lista de equipos con sus nuevas posiciones. Seguidamente, para que cuando se haga click en un marcador, aparezca una ventana con la información relativa al equipo, como muestra la figura 5.16, se añade un manejador de eventos a dicho marcador, mediante el método `google.maps.event.addListener()`. A este método se le pasa el marcador que se va enlazar con dicho evento y el nombre de éste con el que se quiere que se ejecute la función del tercer parámetro. En esta función primero se creará la ventana mediante `google.maps.InfoWindow()` incluyendo como contenido, la información relativa al equipo asociado al marcador. Y, después aparecerá por pantalla utilizando su método `.open()`. Por último, una vez que se haya creado el marcador, se insertará en el mapa utlizando el método de la API `.setMap()`.

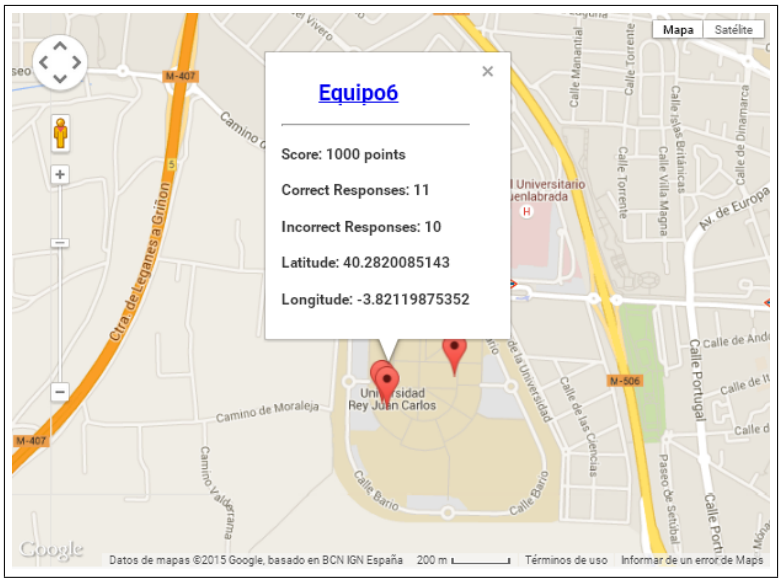


Figura 5.16: Formas de acceder a la página web donde se enviarán los mensajes.

5.3. Jugador.

En esta parte del proyecto se pasará a explicar la nueva versión de la aplicación con la que el usuario podrá participar en las gymkhanas disponibles. Como ya se mencionó en el Capítulo 2 de Objetivos, la meta principal es llevar la aplicación Android desarrollada por Jorge Fernández a una aplicación web accesible desde cualquier navegador. De manera que pueda proveer a los equipos inscritos en las gymkhanas los retos a superar, las pistas que se podrán necesitar, un servicio de mensajería con el que se podrán comunicar con el administrador, etc...

5.3.1. Diagrama de Flujo.

Antes de comenzar a explicar el funcionamiento de la aplicación web para el perfil del jugador, se presentará el diagrama de flujo de la figura 5.17 que describe de manera global el flujo de pantallas que se le va mostrando al usuario a medida que navega por la aplicación.

Dicho esquema muestra el funcionamiento habitual que el jugador realiza, pero dependiendo de las opciones que escoja puede resultar otro diagrama de flujo. Estas opciones pueden ser obtener pistas, usar el servicio de mensajería... que se explicarán más adelante.

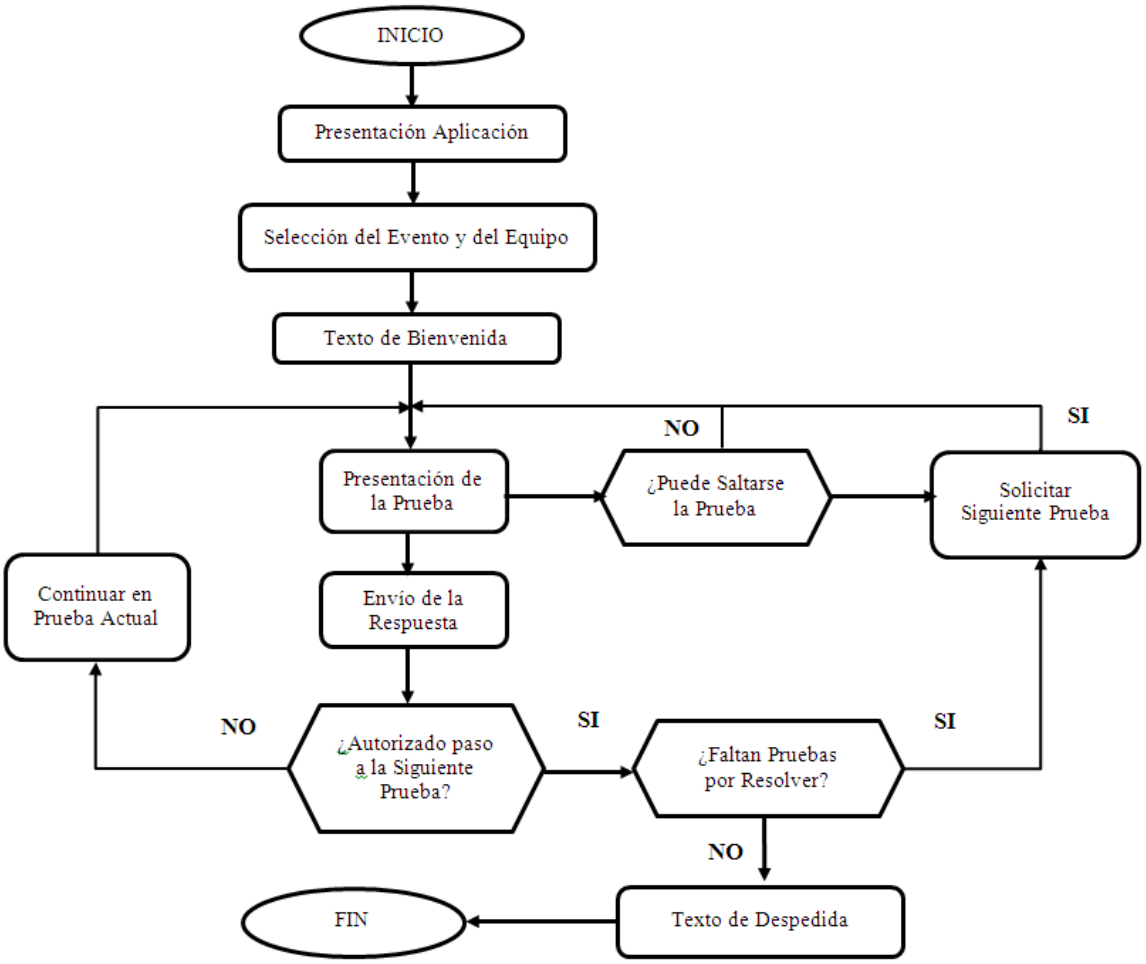


Figura 5.17: Diagrama de Flujo de la aplicación web en la interfaz web del jugador.

5.3.2. Diseño global HTML de las pantallas.

Antes de empezar a describir el funcionamiento de la aplicación, se comentará brevemente la estructura de cada documento HTML que mostrará las diferentes páginas web por pantalla.

Se ha decidido utilizar jQuery Mobile[4][8] ya que ofrece la ventaja de poder crear múltiples páginas en un único archivo HTML, separándolas con un identificador y accediendo a ellas mediante links sin tener que realizar peticiones al servidor. Con este tipo de archivos, se cargará la página principal y el resto permanecerán en segundo plano hasta que el usuario quiera acceder a ellas. En principio todos los documentos HTML de este proyecto se han realizado con una sola página por archivo, salvo por el documento que muestra una de las páginas principales del juego, es decir, aquella que muestra todo lo referente a la participación en una gymkhana. Para

usar estas librerías, se añadirán los links correspondientes a los scripts de jQuery y sus hojas de estilo CSS, en la parte de la cabecera o *head* del documento HTML. En la parte del cuerpo o *body*, es donde se crearán cada una de las páginas por separado. Cada una de ellas seguirá el formato que presenta el código 5.4, el cual hace referencia a un trozo de código con el que se mostrarán los retos de la gymkhana.

```
<div data-role="page" id="reto">

  <div data-role="header" data-position="fixed">
    <div data-role="navbar">
      <ul>
        <li><a href="#reto"   class="ui-btn-active
           ui-state-persist">
          <img src="/gymkhana/templates/css/images/45x45/
             gymkhana.png"><br>
          Reto
        </a></li>
        <li><a href="#ubicacion">
          <img src="/gymkhana/templates/css/images/45x45/mundo.
             png"><br>
          Ubicación
        </a></li>
        <li><a href="#mensajes">
          <span id="smsC"><img  src="/gymkhana/templates/css/
             images/45x45/message.png"></span><br>
          Mensajes
        </a></li>
      </ul>
    </div>

  <div data-role="content">...</div>
  <div data-role="footer">...</div>
```

```
</div>

<div data-role="page" id="ubicacion" >

  <div data-role="header">
    <div data-role="navbar">
      <ul>
        <li><a href="#reto">
          <img src="/gymkhana/templates/css/images/45x45/
             gymkhana.png"><br>
          Reto
        </a></li>
        <li><a href="#ubicacion"  class="ui-btn-active
           ui-state-persist">
          <img src="/gymkhana/templates/css/images/45x45/mundo.
             png"><br>
          Ubicación
        </a></li>
        <li><a href="#mensajes">
          <span id="smsU"><img src="/gymkhana/templates/css/
             images/45x45/message.png"></span><br>
          Mensajes
        </a></li>
      </ul>
    </div>
  </div>

  <div data-role="content">...</div>
  <div data-role="footer">...</div>
</div>
```

Código 5.4: Código HTML-jQuery para la implementación de múltiples páginas en un sólo documento HTML.

Como se puede apreciar en el código 5.4, cada página es creada con un *div* que tiene un atributo *data-role="page"* y un identificador para poder acceder a ellas mediante links, por ejemplo, *id="reto"*. De esta manera, cuando el usuario pulse en un link, *href="#reto"*, el sistema automáticamente buscará por una página que contenga ese identificador y la mostrará por pantalla. Dentro de cada *"page"*, se puede escribir cualquier código HTML, pero es conveniente seguir con una estructura de cabecera, cuerpo y pie de página para diferenciar mejor entre las partes de un documento. Por ello se incluyen tres *div* con diferentes valores para el atributo *data-role="page"*, haciendo referencia al papel que toman dentro del documento: *header* para la cabecera, *content* para el cuerpo o *footer* para el pie de página.

Además de *"page"* también se pueden crear *"dialog"* que tienen la apariencia de ventanas, o *pop-ups*, suspendidas sobre la página desde la que se llaman. Aunque se explicará más adelante su funcionamiento, también se pueden crear varios en el mismo documento HTML como las *"page"*, e incluso haber varios de cada tipo. Un ejemplo de esto último se muestra en el siguiente código 5.5, el cual hace referencia a la estructura del código de la pantalla de autenticación de la aplicación web del jugador.

```
<div data-role="page">
   <div data-role="content" id="content">...</div>
   <div data-role="footer" data-position="fixed">...</div>
</div>

<div data-role="dialog" id="about">
   <div data-role="header" data-theme="a">...</div>
   <div data-role="content" data-theme="b">...</div>
</div>
```

Código 5.5: Código HTML-jQuery para la mezcla de páginas y diálogos en un sólo documento HTML.

5.3.3. Funcionamiento.

Para acceder a la aplicación web, se necesita la IP y el puerto desde donde está corriendo el servidor ya que la URL tiene el siguiente formato:

http://<ip>:<puerto>/gymkhana/mobile/login

Actualmente el servidor está corriendo desde la IP 193.147.51.213 en el puerto 8080, por lo que la URL quedaría de la siguiente manera:

http://193.147.51.213:8080/gymkhana/mobile/login

Este enlace nos dirige a la página web principal de la aplicación (figura 5.18(a)), es decir, la página de presentación desde donde el usuario podrá realizar tres funciones:

- Saber más sobre la aplicación pulsando el botón *"Acerca de"* que aparece en la parte inferior de la pantalla. Si se hace click en dicho botón, nos saldrá una imagen como la de la figura 5.18(b).

- Registrarse. Pulsando el botón correspondiente se redirigirá a una página con un formulario, como el de la figura 5.19, donde el usuario introducirá los datos correspondientes y, una vez aprobados, volverá a la página principal para introducir sus credenciales y poder comenzar.

- Autenticarse. Si el usuario introduce unos credenciales incorrectos se le informará para que vuelva a intentar introducirlos y si, por el contrario son correctos, tendrá acceso al juego.

Una vez que el usuario introduzca sus credenciales correctamente, se le llevará a una página web con tres botones, tal y como muestra la figura 5.20, con los cuales podrá salirse del juego, visualizar su posición en el ranking global de la aplicación (figura 5.21) y acceder a la lista de gymkhanas disponibles.

Si el usuario pulsa en *"Jugar"*, la siguiente pantalla que se muestra es la de la figura 5.22, donde se visualizarán las gymkhanas disponibles en las que puede participar y la información relativa de cada una de ellas: lugar, dificultad...

Tras escoger uno de los eventos de la lista, el usuario tendrá que seleccionar el equipo con el que desea participar de la lista de equipos inscritos en la gymkhana como aparece en la figura 5.23.

(a) Pantalla de autenticación.

(b) Pop-up con información sobre la aplicación.

Figura 5.18: Pantalla principal de acceso a la aplicación.

Figura 5.19: Pantalla para crear un nuevo usuario.

Figura 5.20: Pantalla con el menú de acciones que dispone el jugador.

(a) (b)

Figura 5.21: Pantalla con los rankings de la aplicación.

Figura 5.22: Pantalla con la lista de gymkhanas disponibles.

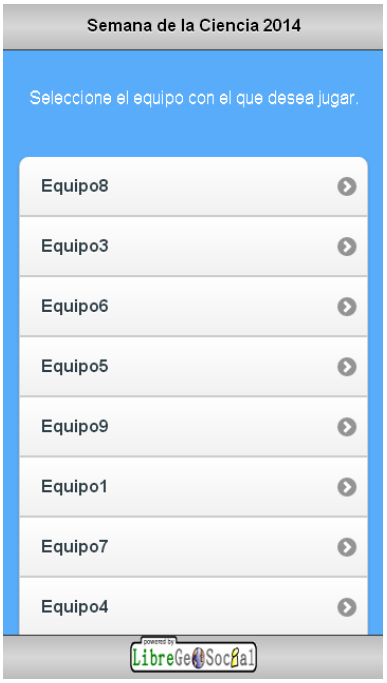


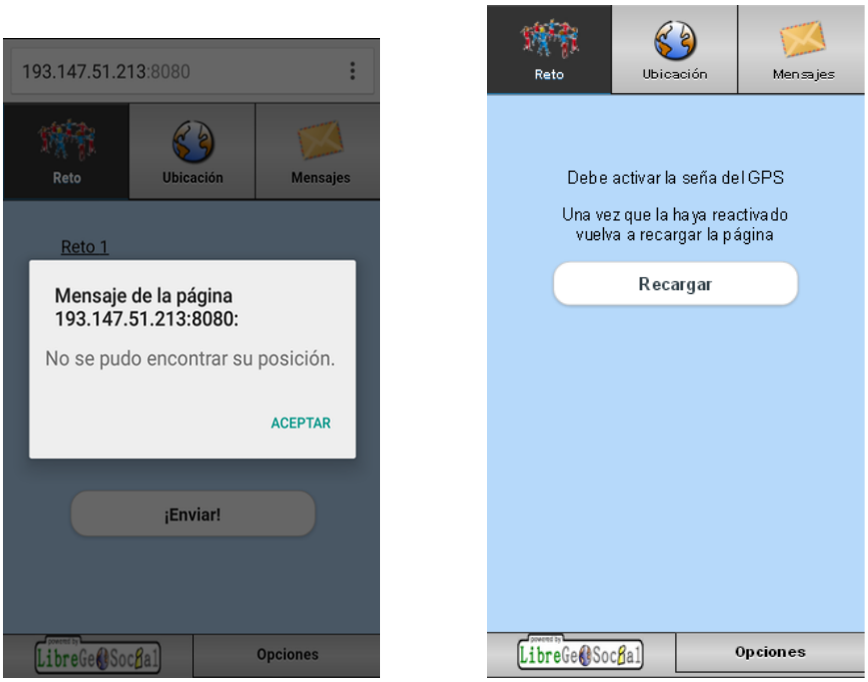
Figura 5.23: Pantalla con la lista de equipos disponibles.

Una vez seleccionados gymkhana y equipo, antes de comenzar con el juego, se mostrará una pantalla como la de la figura 5.24 donde aparecen el mensaje de bienvenida e imagen introducidos por el administrador cuando configuró el evento. A partir de esta pantalla, el equipo empezará a resolver los retos que se le muestran comenzando por aquel que le asignó el mánager.

Figura 5.24: Pantalla de bienvenida a la gymkhana.

En este punto, antes de empezar su participación en una gymkhana, es importante que el usuario tenga activada la señal del GPS, ya que cada respuesta lleva asignada unas coordenadas geográficas con la que se situará al equipo en el mapa geográfico de la página web de monitorización (figura 5.12) que estará visualizando el mánager del evento. Por eso, si no está disponible la señal del GPS se informará al usuario del dispositivo que la active y así poder visualizar el reto a superar, tal y como se muestra en la figura 5.25. Para obtener dichas coordenadas se ha utilizado la API de Geolocalización de HTML5 que se comentará más adelante.

Cuando el usuario tenga activada la señal se le mostrará el evento a superar, que tal y como se comentó en el sub-apartado 5.2.2, existen tres tipos de retos: textuales, fotográficos y de geolocalización. En cuanto al diseño web de éstos, todos siguen la misma estructura, lo único en lo que se diferencian es en la información mostrada al usuario (véase figura 5.26)



(a) (b)

Figura 5.25: Notificación de ubicación desactivada.

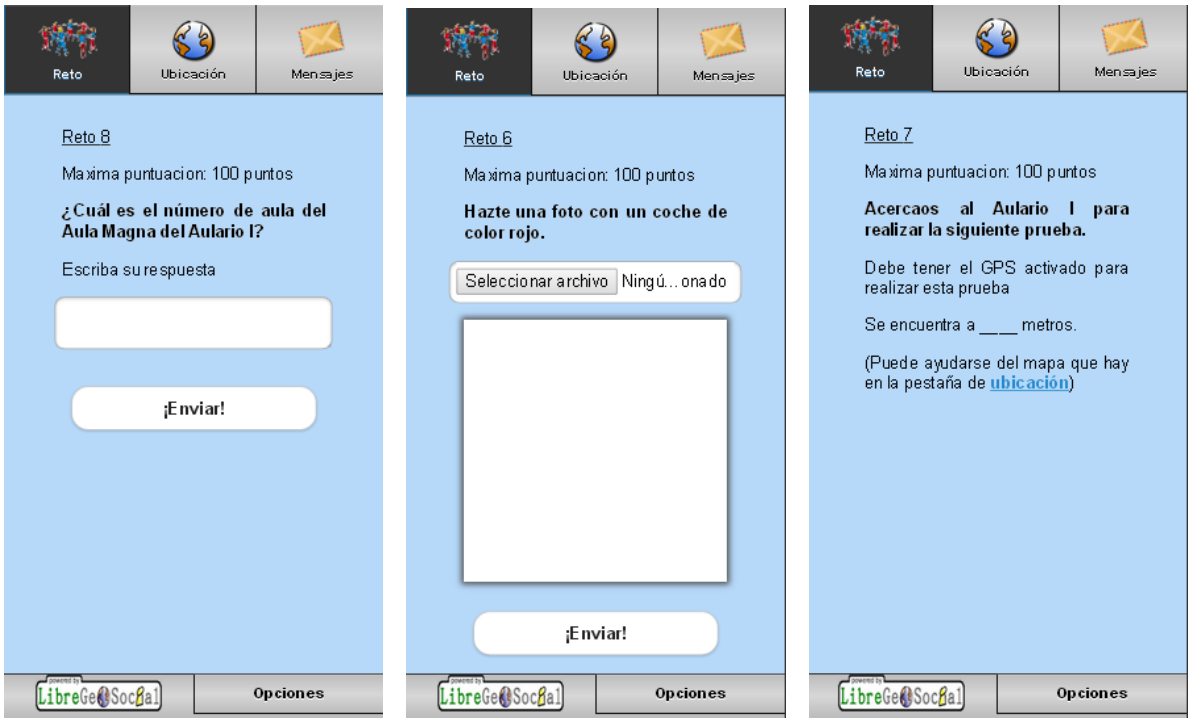
Como se puede ver en las figuras 5.26, las pantallas constan de tres zonas que se describirán a continuación.

Cabecera.

En esta sección de la pantalla se muestran tres pestañas:

- *"Reto"* donde aparecerá la información relativa del reto a superar.
- *"Ubicación"* donde se podrá mostrar un mapa de Google que mostrará la posición geográfica del equipo. Esta pestaña está diseñada principalmente para el reto de geolocalización que se explicará más adelante.
- *"Mensajes"* destinada al servicio de mensajería que estará disponible en todo momento para el equipo.

La información que muestra cada pestaña se explicará de manera más extendida en los siguientes apartados.



(a) Textual. (b) Fotográfico. (c) Geolocalización.

Figura 5.26: Pantallas de los tres tipo de retos que se pueden realizar.

Cuerpo.

Esta sección es la que contiene la información correspondiente a la pestaña seleccionada. En cuanto a la pestaña *"Reto"*, como se ha mencionado a lo largo de esta memoria, existen tres tipos de respuesta que se van a describir en los siguientes subapartados.

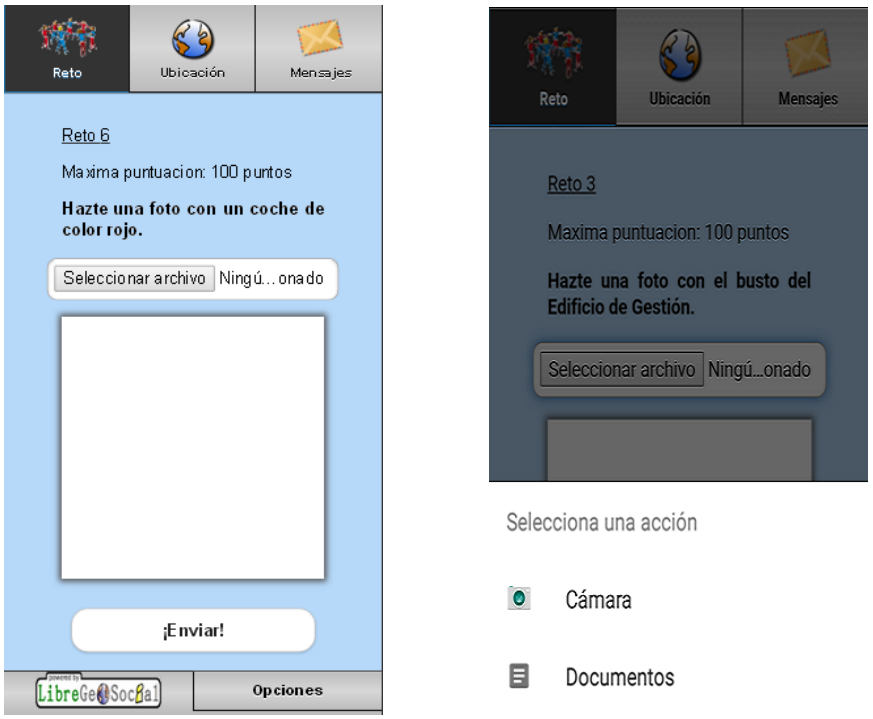
Retos de respuesta textual.

Como indica el título, este tipo de reto es aquel en el que el equipo deberá responder textualmente, es decir, deberá rellenar con una cadena de texto o *string* el cuadro de texto proporcionado en la página web para introducir la respuesta. Como en cualquier input de texto, cuando el equipo pulse en él, aparecerá el teclado QWERTY correspondiente al dispositivo móvil y podrá introducir su respuesta. Ésta será enviada al servidor donde se validará comparando con todas las respuestas posibles que el mánager proporcionó en su configuración, ya que una respuesta se puede escribir de distintas maneras. Por ejemplo, si se pregunta por un número, el jugador puede responder:

- 5.
- Cinco.

Y ambas soluciones son igualmente válidas. De ahí, que el mánager del evento tenga que contemplar todas las respuestas posibles que se pueden dar. Aun así, podrá validar la respuesta desde su pantalla de monitorización. De la misma forma, también puede darla como errónea. Otra de las opciones que el mánager puede utilizar es permitir que los equipos pasen al siguiente reto respondan correcta o incorrectamente la prueba. Cuando se contemple esta acción, no se informará al equipo sobre la resolución del reto, y pasará a realizar la siguiente prueba. En caso contrario, se le notificará que necesita realizar de nuevo el reto para poder continuar.

Retos de respuesta fotográfica.



(a) Página web del navegador.

(b) Ventana que abre el dispositivo móvil para seleccionar la imagen.

Figura 5.27: Pantallas de retos de respuesta fotográfica.

Esta clase de retos son aquellos que piden a los equipos enviar una foto según plantee el enunciado. Como aparece en la figura 5.27(a), la pantalla consta de 4 partes:

- La primera donde se presenta la información del reto: puntuación y enunciado.
- La parte que viene a continuación permite al usuario del dispositivo seleccionar una imagen ya sea tomando una foto desde la cámara o cargándola desde el administrador de archivos. Para ello, sólo hay que añadir al documento HTML la siguiente línea de código:

```
<input type="file" id="take-picture" name="photo" accept="
    image/*">
```

  Con esta línea se crea un campo de selección de archivos que solamente sean imágenes y, además, añade un botón *"Seleccionar archivo"* (como el de la figura 5.27(a)) para poder elegir los archivos (figura 5.27(b)). Las imágenes que se carguen se mostrarán en la siguiente parte de la pantalla.

- La tercera parte es donde aparecerá la imagen que cargue el usuario. Para realizar esta operación, se ha creado un archivo JavaScript que se incluirá en la cabecera del documento HTML, y que sólo funcionará una vez que dicho documento se haya cargado por completo. Tal y como muestra el código 5.6, se crea una variable que hará referencia al input de entrada que mencionamos anteriormente, y se le asociará una función que se ejecutará cuando dicho elemento cambie, es decir, cuando el usuario haya seleccionado una imagen. Una vez dentro de la función, se comprobará si se ha seleccionado alguna imagen, y en caso de ser cierto, se le pasará a otra función, `loadImage()`, para que la introduzca en el cuadrado en blanco de la figura 5.27(a). Con esta nueva función, se cargará la imagen en una etiqueta <img> y se dibujará completamente en un *canvas* previamente declarado en el documento HTML. La etiqueta *canvas*, es nueva en HTML5 por lo que es posible que algún navegador todavía no la soporte, en este caso se avisará al usuario de tal información.

```
function loadImage(url)
{
    var $img = $("<img>");
    $img.attr("src", url);
    $img[0].onload = function(){
          var canvas = $("canvas")[0];
```

```
            var context = canvas.getContext("2d");
            context.drawImage(this, 0, 0, canvas.width, canvas.
               height);
    }
    $img[0].onerror = function()
            alert("Error al cargar la imagen!");
    }
}

var takePicture = document.querySelector("#take-picture");

takePicture.onchange = function (event) {
 var files = event.target.files,
 file;
 if (files && files.length > 0) {
   file = files[0];
   var reader = new FileReader();
            reader.onload = function() { loadImage(reader.result
               ); };
            reader.readAsDataURL(file);
       }
};
```

Código 5.6: Código JavaScript para la visualización de imágenes cargadas por el usuario.

- Y, por último, el botón *"Enviar"* con el que se enviará la imagen al servidor. En caso de que no se haya incluido ninguna imagen en el canvas, se redirigirá al equipo a una página de error donde se le informará que no ha introducido ninguna imagen.

Dado que es imposible que el servidor verifique la imagen de manera automática, dicho servidor considerará el reto como correcto. Desde la interfaz web que se comentó en el subapartado 5.2.2, el organizador del evento es capaz de visualizar la imagen e invalidarla si lo cree necesario, en este caso, el equipo tendrá que volver a realizar el reto.

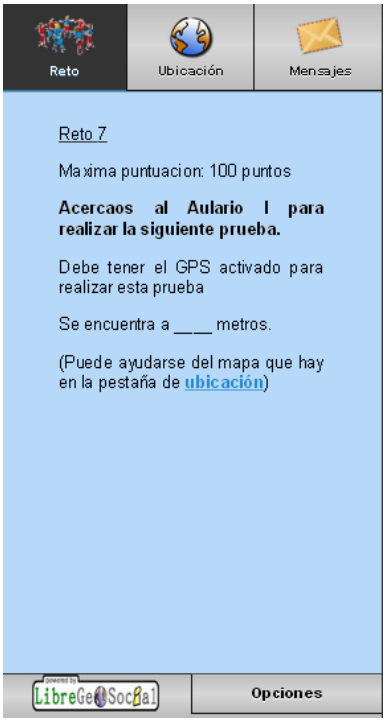


Figura 5.28: Pantallas de retos de geolocalización.

Retos de geolocalización.

En los retos de geolocalización se les pedirá a los equipos que se dirijan a un determinado lugar. Dicho lugar está compuesto por dos coordenadas geográficas configuradas por el organizador de la gymkhana en el momento de creación del reto. También se puede establecer un radio de distancia en torno al lugar objetivo para que se dé la prueba como superada ya que es imposible que un equipo alcance el lugar exacto debido al error que puede introducir la señal del GPS del propio dispositivo móvil por una mala cobertura.

En este tipo de retos, el servidor enviará junto con el enunciado las coordenadas geográficas más el radio máximo permitido para poder superar la prueba. Éstas no serán mostradas por pantalla, sino que servirán para realizar todo el proceso de geolocalización: visualización en el mapa de la pestaña *"Ubicación"*, tal y como muestra la figura 5.28, y cálculo de la distancia entre el equipo y el lugar de destino, teniendo en cuenta el margen de distancia.

Una vez que la distancia entre el equipo y el lugar objetivo sea menor que el radio máximo permitido, se avisará al usuario de que ha llegado al lugar deseado y se enviará al servidor una

confirmación de que dicho reto ha sido superado con éxito.

Para calcular la posición del usuario se han utilizado dos APIs JavaScript, la de Google Maps[12] y la de Geolocalización de HTML5[11], que se comentarán en el siguiente subapartado.

Pie de página.

En esta sección se encuentra el botón de *"Opciones"* que mostrará un menú con las operaciones que se pueden realizar: compra de pistas, salir del juego, saltar el reto... Este menú estará disponible desde cualquiera de las pestañas de la pantalla.

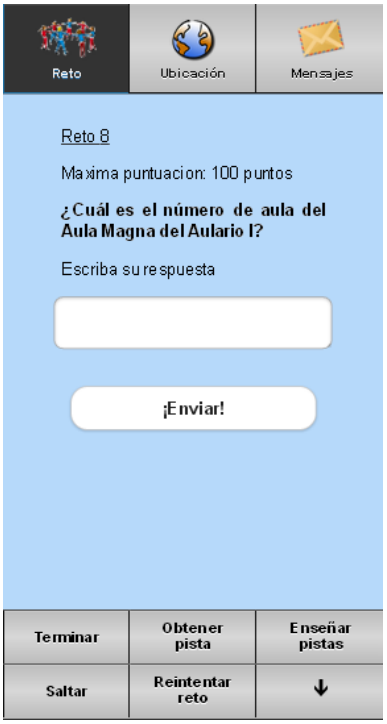


Figura 5.29: Pantallas de retos de respuesta fotográfica.

Como se puede apreciar en la figura 5.29 existen seis acciones que puede realizar el usuario. El primer botón, *"Terminar"*, como su nombre indica sirve para que el equipo finalice su participación en la gymkhana en cualquier momento, no hace falta que haya superado todos los retos. Con el botón que está a su derecha, *"Obtener pista"*, el usuario puede comprar pistas a cambio de puntos, siempre que el mánager del evento las haya proporcionado en la configuración del

reto. Si el equipo desea consultar las pistas adquiridas, tendrá que pulsar el botón correspondiente, *"Enseñar pistas"*. Si por alguna razón, el equipo desea saltarse el reto y el mánager lo ha diseñado para que se pueda realizar dicha acción, deberá pulsar en el botón *"Saltar"* y el reto se podrá realizar posteriormente. Por el contrario, si el equipo decide volver a realizar algún reto que se haya saltado, deberá pulsar en el botón *"Reintentar reto"* y se le mostrará una ventana con la lista de retos que puede reintentar realizar.

Cuando se pulse cualquiera de los botones anteriores, aparecerá una ventana confirmando la operación correspondiente que desea realizar, tal y como se muestra en la figura 5.30, salvo por el botón que contiene una flecha con el que se ocultará el menú de opciones.

Figura 5.30: Ventanas con la información relativa a la opción pulsada.

Para implementar estas ventanas, se han utilizado ciertos atributos de jQuery. Para empezar, a los links $<a>$ se les han asignado atributos como los siguientes:

- *data-role="button"* para especificar que tiene la función de un botón.
- *data-inline="true"* para que todos los botones estén en la misma línea.
- *data-rel="dialog"* para que la siguiente ventana que se muestre parezca un pop-up.

- *data-transition="pop"* para especificar el tipo de transición que se desea hacia la otra página.

Cuando el usuario haga click en cualquiera de estos botones, se hará una petición al servidor para que se muestre la información correspondiente al botón pulsado. El servidor nos mostrará dicha información en una nueva pantalla que tendrá la apariencia de los pop-ups que se muestran en la figura 5.30. Estos documentos HTML ahora se definen como diálogos y no como páginas, es decir, si antes se tenía *data-role="page"* ahora se tiene *data-role="dialog"*. Pero siguen teniendo cabecera y cuerpo (*data-role="header"* y *data-role="content"* respectivamente).

En cuanto al contenido de estos diálogos, si se puede realizar la acción deseada (saltar reto, comprar pistas, finalizar gymkhana...) aparecerán dos botones: uno para confirmar que se quiere seguir adelante y otro para denegar la petición. En el caso contrario, sólo aparecerá el de cancelar la operación. Este último botón tiene la función de volver a la página anterior, es decir, se cierra el diálogo. Para ello, al link $<a>$ no se le especifica ninguna url a la que redirigir, sino que se le pasa un atributo *data-rel="back"* para que vaya un paso atrás en el historial.

Además, con jQuery se puede dar estilo a cada botón o elemento en el documento HTML con el atributo *data-theme*. La versión de jQuery que se está utilizando en este proyecto tiene predefinidos cinco temas[1] ordenados alfabéticamente de la *a* a la *z*. Para utilizarlo nada más que se tiene que añadir dicho atributo al elemento deseado con *data-theme="b"*, por ejemplo.

GPS.

En este apartado, se comentará de manera más exhaustiva la obtención de la posición en la que está situado el equipo. Para ello se han utilizado dos APIs: la correspondiente a la geolocalización de HTML5[11], con la que se obtendrá la localización, y la propia de Google Maps[12], con la que se mostrará dicha ubicación en un mapa. Esta última API se carga desde la siguiente URL, `http://maps.googleapis.com/maps/api/js`, a partir de la cual se cargan todos los símbolos y objetos JavaScript que se utilizarán posteriormente. Además se pueden añadir más librerías agregando a la URL anterior *"libraries=nombre_de_la_librería"*. En este

[1] http://demos.jquerymobile.com/1.3.2/faq/how-does-theming-work.html

caso, se necesitará la librería *"geometry"* para calcular la distancia entre el usuario y el lugar de destino. De esta manera, se cargará la API con la siguiente línea de código.

```
<script src="http://maps.google.com/maps/api/js?libraries=geometry"><
   /script>
```

Para localizar al dispositivo geográficamente se ha utilizado la API de Geolocalización de HTML5, que por razones de privacidad necesita que el usuario autorice el permiso para proporcionar información sobre la ubicación. Por otra parte, es necesario conocer si el navegador soporta dicha API JavaScript, para ello se hará uso de la función `navigator.geolocation` propia de dicha API que devolverá un booleano *true* en caso de que sea soportada y *false* en el caso contrario. En el caso afirmativo, se obtendrá la posición del usuario usando el método `getCurrentPosition()` al que se le pasan tres parámetros: *"success"* que es el nombre de la función que se ejecutará si se ha podido obtener la ubicación del usuario, *"error"* que es la función a ejecutar si se ha producido algún error y *"options"* con el que se pueden establecer ciertas variables de configuración. En cuanto a este último, se pueden configurar los siguientes valores:

- *"enableHighAccuracy"* o Exactitud. Si esta opción está activada, se obtendrá la ubicación del usuario de la manera más precisa posible. Esto puede ser un inconveniente ya que aumenta el uso de batería y los tiempos de respuesta pueden ser más lentos. Por defecto, este valor está desactivado para no consumir muchos recursos.

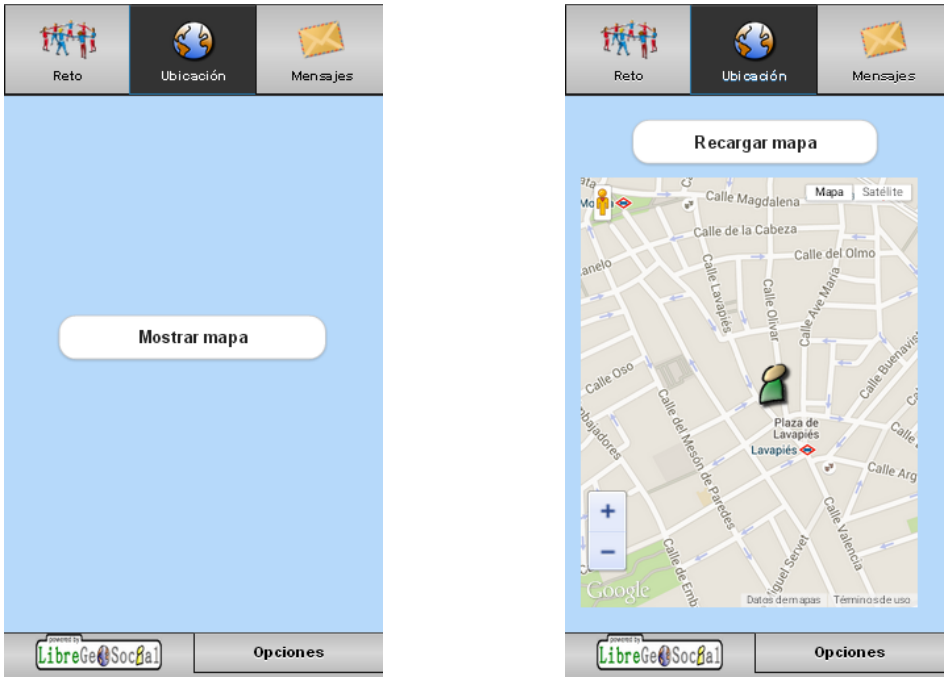
- Timeout. Indica el máximo valor de tiempo, en milisegundos, con el que se puede responder con la posición. El valor por defecto es *"Infinity"* o infinito, es decir, no hay tiempo máximo de respuesta.

- MaximumAge. Indica la antigüedad máxima de una posición en caché que puede ser válida. Su valor por defecto es 0, siempre tiene que buscar la posición real, pero si es *"Infinity"* deberá tomar una posición en caché, si hay, independiente de la antigüedad.

Mientras que el usuario no pulse el botón *"Mostrar mapa"* de la pestaña de *"Ubicación"*, estos parámetros se mantendrán por defecto y no se introducirán. Por el contrario, y como se

comentará más adelante, si este botón es pulsado, dichos parámetros cambiarán.

Teniendo en cuenta el primer escenario, las variables que se han pasado a `getCurrentPosition()` son los nombres de los otros parámetros, por ejemplo, con la función de error se informará al usuario de que no se ha podido encontrar su posición. En el caso de que se haya podido obtener la ubicación del usuario, se ejecutará la función correspondiente a *success* en la que se guardarán las coordenadas geográficas, se mostrará la ubicación del usuario en el mapa que hay en la pestaña *"Ubicación"* y, en el caso de que el reto sea de tipo de geolocalización, se mostrará por pantalla la distancia que hay hasta el lugar objetivo.



(a) Página web principal

(b) Página web tras pulsar el botón *"Mostrar mapa"*

Figura 5.31: Posibles pantallas de la pestaña *"Ubicación"*

Para cualquier tipo de reto, cuando el usuario pulse el botón de *"Mostrar mapa"*, como el de la figura 5.31(a), en la pestaña de *"Ubicación"*, se ejecutará un script diferente pero que tiene la misma funcionalidad. Este botón tiene asociada una función `initLocationProcedure()` que se ejecutará cuando se haga click en él.

```
function initLocationProcedure() {
```

```
    $('#btn1').hide();
    $('#btn2').css('display', 'block');
    getMap();
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(displayAndWatch,
          locError);
    } else {
      alert("Your browser does not support the Geolocation API"
         );
    }
 }
```

Este método oculta el antiguo botón con el que se muestra el mapa y enseña un nuevo botón para recargarlo, como el de la figura 5.31(b). Lo siguiente que realiza es llamar al método `getMap()` que asignará a la variable *map* el mapa de Google Maps donde se mostrarán las posiciones del usuario y de destino. Esta variable contiene el *div* del HTML donde se insertará el mapa, el zoom inicial, el centro y tipo de vista de dicho mapa.

```
function getMap() {
    map = new google.maps.Map(document.getElementById('
       mapholder'),
          {
              zoom: 16,
              center: mapCenter,
              mapTypeId: google.maps.MapTypeId.ROADMAP
          });
}
```

Y, por último, comprueba que el navegador soporte la API de geolocalización como se comentó anteriormente. Si no se puede obtener una posición, se ejecutará `locError()`, que avisará al usuario de tal error.

```
function locError(error) {
   alert("No se pudo encontrar su posición.");
}
```

Si, por el contrario, sí que se puede obtener la ubicación del usuario, se ejecutará `displayAndWatch()` que tiene tres partes. Primero llamará al método `setCurrentPosition()` y le pasará como parámetro la posición obtenida.

```
function displayAndWatch(position) {
   setCurrentPosition(position);
   watchCurrentPosition();
   $('#cnvs').show();
}
```

Este método primero establecerá el marcador del usuario con `google.maps.Marker()`. Ésta es una función de la API de Google Maps que define una localización en el mapa.

```
function setCurrentPosition(pos) {
   var icon = { url: '/gymkhana/templates/css/images/user_48.
      png'  };
   currentPositionMarker = new google.maps.Marker({
         map: map,
         icon:icon,
         position: new google.maps.LatLng(
           pos.coords.latitude,
           pos.coords.longitude
         ),
         title: "¡Usted está aquí!"
   });

   {% ifequal challenge_type 3 %}
         lat2= {{ target_place.y }};
         lon2= {{ target_place.x }};
```

```
            destinMarker = new google.maps.Marker({
                map: map,
                position: new google.maps.LatLng( lat2, lon2 ),
                title: "Destino"
            });
    {% endifequal %}

    map.panTo(new google.maps.LatLng(
            pos.coords.latitude,
            pos.coords.longitude
    ));
}
```

A esta función se le pasa un solo parámetro que especifica las propiedades iniciales del marcador. De entre todos los parámetros que se pueden configurar [2], se han escogido cuatro:

- *"map"* que es la variable, previamente definida, que contiene el mapa en el que se visualizará el marcador.
- *"icon"* que, como su nombre indica, especifica la imagen que será el marcador. En este proyecto, se ha escogido una imagen externa diferente de la que el navegador muestra por defecto.
- *"position"* que contiene la ubicación del marcador en el mapa. Este parámetro es creado con un método, `google.maps.LatLng(lat,lng)`, que establece un punto geográfico con las coordenadas geográficas del usuario.
- *"item"* es una cadena de texto que se mostrará cuando se haga click en el marcador.

La siguiente parte de la función, sólo se ejecutará si el reto es de tipo de geolocalización. En ella se creará el marcador del lugar al que se tienen que dirigir los equipos. En este caso, el marcador se ha establecido de la misma manera que antes pero cambiando algunos parámetros.

[2] https://developers.google.com/maps/documentation/javascript/reference?hl=es#MarkerOptions

El parámetro *"map"* es igual que el de antes, *"position"* ahora se establece con las coordenadas geográficas del lugar de destino, y *"title"* que tendrá un mensaje diferente. El icono que mostrará este marcador es el que Google Maps posee por defecto.

El siguiente paso es centrar el mapa en la posición del usuario. Para ello se utiliza otra función de la API de Google Maps, `panTo()`, a la que se le pasa las coordenadas del usuario.

La otra función que se ejecuta desde `displayAndWatch()` (código 5.3.3), es `watchCurrentPosition()`.

```
function watchCurrentPosition() {

 var options = {
     enableHighAccuracy: true,
     timeout: 60000,
     maximumAge: 0
 };

 navigator.geolocation.watchPosition(
   function (position){
     setUserMarker(userMarker, position);

     var lat=position.coords.latitude;
     var lon=position.coords.longitude;
     var acc=position.coords.accuracy;
     $('#latitude').val(lat);
     $('#longitude').val(lon);
     $('#latitude2').val(lat);
     $('#longitude2').val(lon);
     var type =  $('#type_challenge').text();

     if (type == 3){
       var Dlat = $('#Dlatitude').text();
```

```
        var Dlon = $('#Dlongitude').text();
        var distance_ch = $('#distance').text();
        var distance = GetDistance(lat, lon, Dlat, Dlon);
        var new_distance = distance - acc;
        var meters = distance.toString();
        $('#meters').html(meters.slice(0,5));
        if (new_distance < distance_ch){
          alert("¡Ha alcanzado su destino!");
          $('#dist_difference').val(distance);
          $('#formUb').submit();
        }
      }
    }, locError, options
  );
}
```

Esta función utiliza otro método para obtener la posición del usuario, `navigator.geolocation.watchPosition(success, error, options)`. En este caso, se ha decidido cambiar algunos de los valores por defecto del parámetro *"options"*:

- Se ha activado *"enableHighAccuracy"* para obtener una mayor precisión en la posición del usuario.
- Se ha establecido *"timeout"* a 60.000 para que permita obtener la posición del usuario durante sesenta segundos. Tras ese tiempo, si no ha podido obtener una posición, se lanzará la función de error `locError()` indicando que se ha producido un error de "TIMEOUT".
- La opción *"maximumAge"* no se ha cambiado y sigue teniendo el valor por defecto (cero).

La diferencia que existe con la que se ha utilizado previamente, `navigator.geolocation.getCurrentPosition()`, es que la función de *success* ahora se ejecuta cada vez que se detecte que la posición del usuario ha cambiado. Esta función primero cambiará el marcador del usuario con el siguiente método:

```
function setUserMarker(marker, position) {
     marker.setPosition(
       new google.maps.LatLng(
         position.coords.latitude,
         position.coords.longitude)
     );
  }
```

Y, después, introduce las nuevas coordenadas en el formulario que se enviará cuando el usuario haya llegado al lugar de destino. Para ello se utilizará la API de jQuery[9], asignando el nuevo valor, con el método `.val()`, al nodo con el identificador deseado. Y, por último, si el reto es de tipo de geolocalización se calculará la distancia que hay entre el usuario y el lugar de destino con el método `GetDistance()`, el cual hará uso de otra función de la API de Google Maps, `google.maps.geometry.spherical.computeDistanceBetween()`, que calculará la distancia entre dos puntos geográficos.

```
function GetDistance(Olat, Olon, Dlat, Dlon)
{

  var x1 = new google.maps.LatLng(Olat, Olon);
  var x2 = new google.maps.LatLng(Dlat,Dlon);
  var distancia = google.maps.geometry.spherical.
     computeDistanceBetween(x1, x2);

  return distancia;
}
```

Tras las pruebas realizadas para comprobar el buen funcionamiento de la aplicación, como ya se comentará más detenidamente en el Capítulo 6 de Resultados, no se mostraba correctamente la posición del usuario aunque estuviera en el lugar de destino correcto. Para mitigar este defecto, se ha decidido obtener la exactitud con que la posición se ha tomado, expresada en

metros. Con este nivel de precisión, se calcula una nueva distancia, más aproximada a la real, hasta el lugar de destino. Si esta distancia es menor que la distancia máxima permitida para que el reto se considere adecuado, se avisará al usuario de que ha llegado al lugar de destino y pasará a realizar el siguiente reto disponible.

5.4. Servicio de Mensajería.

Como se ha comentado al principio de esta memoria, uno de los objetivos del proyecto de Jorge Fernández fue el de poder realizar con dispositivos móviles uno de los juegos más populares de la sociedad añadiendo varias mejoras. Una de éstas era la eliminación de los desplazamientos innecesarios durante el transcurso del juego. Para ello se implementó un servicio de mensajería interno con el que se pudieran comunicar los equipos con el mánager del evento, o viceversa. Dichos mensajes se almacenan en el servidor, según se explicó en el modelo de datos del apartado 4.2.1, para luego poder ser enviados en caso de que el cliente los solicite.

Para implementar este servicio en la nueva actualización de dicho proyecto, aparte de emplear las tecnologías que se han ido describiendo a lo largo de esta memoria, se ha utilizado AJAX para comunicarse con el servidor en segundo plano mediante mensajes JSON. Concretamente, las únicas veces que se va a utilizar AJAX son: en la interfaz web del cliente Jugador, cuando esté participando en la gymkhana, y en la interfaz web del cliente mánager cuando esté monitorizando dicho evento.

Cada vez que se quiera obtener información del servidor sin tener que recargar la página, se utilizará la tecnología AJAX empleando el método `.ajax()` de jQuery. De todos los parámetros que se pueden configurar para realizar la solicitud, se ha considerado que los más importantes son los siguientes:

```
$ajax({
  type: string que especifica el tipo de solicitud, GET o POST,
  url:  string con la URL del servidor con el que se quiere
        comunicar,
  data: diccionario con los parámetros que se quieren enviar al
        servidor,
```

```
    async : booleano para indicar si la conexión es síncrona o
            asíncrona,
    success: función que se ejecutará si la solicitud ha
             transcurrido sin ningún problema
});
```

En todos los casos que se utilice dicho método, el parámetro *"async"* se ha puesto a *false* porque se ha decidido que el script no siga ejecutándose hasta que se haya recibido respuesta del servidor, es decir, se desea que la conexión sea síncrona.

Según la URL que se introduzca, en el servidor se ejecutará el método correspondiente que obtendrá la información solicitada y la devolverá al cliente en formato JSON. Si no se ha producido ningún error en la petición, el cliente ejecutará la función de *"success"* que procesará la respuesta del servidor.

Como se comentó en el apartado 5.2.2, el mánager del evento podrá ir monitorizando el transcurso del juego desde una página web en el que se le mostrará un ranking con la puntuación de los equipos inscritos, un mapa con la posición de todos los equipos que estén participando, una lista con las respuestas recibidas y un listado con los mensajes intercambiados. Para formar cada una de estas partes, se harán peticiones AJAX en segundo plano mediante la ejecución de un archivo JavaScript una vez se haya cargado la página web. Según se muestra en el código que viene a continuación, primero se creará el mapa y después se ejecutarán periódicamente las funciones correspondientes a la formación de cada parte, incluida la correspondiente al servicio de mensajería.

```
function configurate() {
  createMap(0);
  getMessages();
  getTeamResponses();
  setInterval('getTeamList();',17000,1);
  setInterval('getTeamResponses();',20000);
  setInterval('getMessages();',15000);
}
```

Para ejecutar dichas funciones periódicamente, se ha utilizado el método `setInterval()` de JavaScript con el que se ejecutará la función que se le pase como primer parámetro cada cierto intervalo de tiempo, descrito en milisegundos en el segundo parámetro. Esta función resulta muy práctica para hacer peticiones periódicas al servidor que devuelvan los mensajes almacenados respecto al evento que se está monitorizando. De esta manera, cuando hayan nuevos mensajes que leer, se avisará al administrador de tal suceso y se añadirán a la lista de mensajes mostrados por pantalla, tal y como se muestra en la figura 5.32.

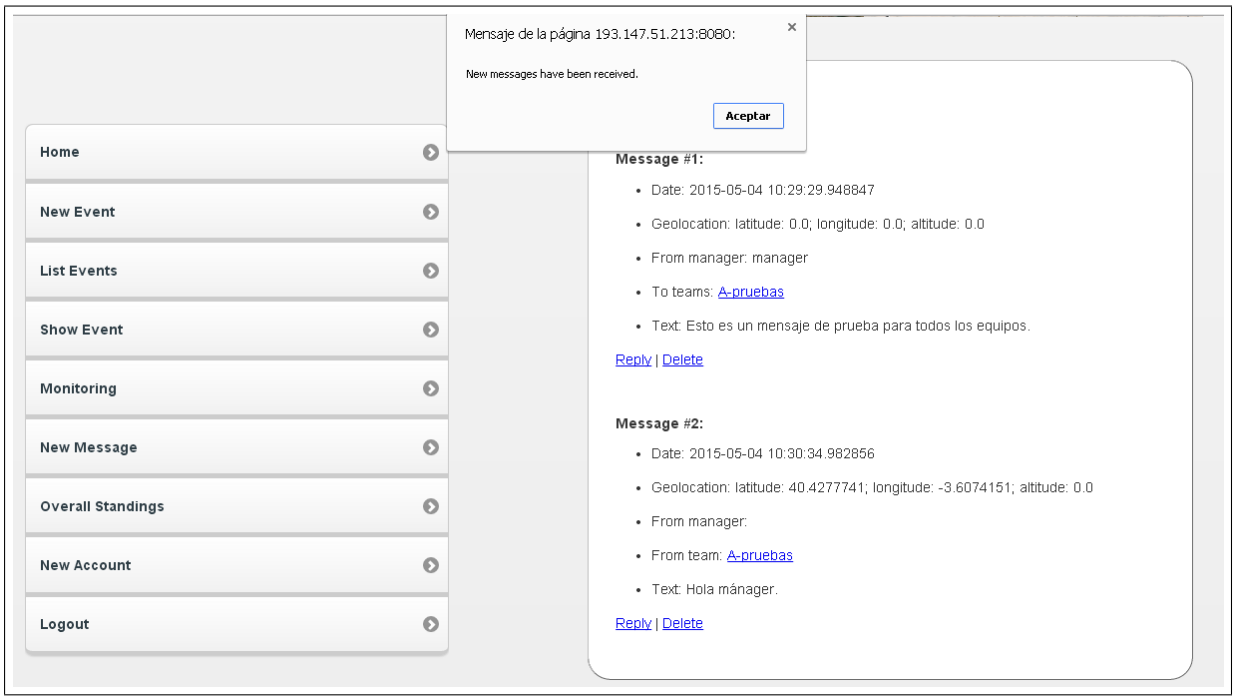


Figura 5.32: Notificación de un nuevo mensaje en la interfaz web del mánager.

Por otro lado, en cuanto a la interfaz web del cliente con papel de Jugador, también se hará uso de AJAX. Al igual que la implementación en la monitorización del evento, cuando se cargue la página HTML que muestra el reto a superar, se ejecutará también un archivo JavaScript el cual se encargará de pedir la lista de mensajes almacenados en el servidor. Para determinar si un equipo ha recibido mensajes nuevos, sin realizar una búsqueda en profundidad sobre la lista de todos los mensajes intercambiados, se ha decidido almacenar en el servidor unos parámetros que hagan más eficaz dicho proceso. Estos parámetros son la fecha y número

de mensajes que se han notificado al usuario desde la última vez. De esta manera, no se recorrerá la lista de mensajes en búsqueda de uno nuevo hasta que el número de mensajes recibidos sea mayor que el número de mensajes que ya se tenía en el cliente.

Específicamente, el script tiene tres partes fundamentales:

- Primero se obtendrán los parámetros que se acaban de describir, para poder compararlos posteriormente con la lista de todos los mensajes intercambiados.
- Después se obtendrán todos los mensajes almacenados en el servidor pertenecientes al evento en el que se está participando.
- Y, por último, se guardarán la última fecha y número de mensajes que se han notificado al equipo.

Para realizar cada una de esas partes, se ha hecho uso de AJAX como ya se describió anteriormente. Cuando se hayan recibido nuevos mensajes, el icono que aparece en la pestaña de *"Mensajes"* cambiará, es decir, la imagen que aparecía antes describiendo la información de la pestaña, se reemplazará por un círculo con el número de mensajes recibidos, tal y como aparece en la figura 5.33. Para realizar dicho cambio se ha utilizado la función `.replaceWith()` de jQuery[9], con el formato que se muestra en la siguiente línea de código:

```
$('#smsC').replaceWith("<span id='smsC' class='pink'>" + num
    + "</span>");
```

Con esta función, se borrará del DOM el nodo que tenga el identificador *"smsC"* con la imagen de los mensajes, y se insertará en su lugar un nuevo nodo con el mismo identificador, pero con una nueva clase para dar estilo al círculo con el número de mensajes recibidos. Además se ha decidido animar dicho círculo para que parpadee unos segundos y así el usuario pueda percibir la notificación. Dicha animación consiste en un círculo que parpadea durante unos segundos hasta quedarse fijo. Para ello se ha hecho uso de la función `.animate()` de jQuery[9] a la cual se le pueden pasar varios parámetros para la configuración de la animación:

- *Propiedades*, que incluyen la lista de valores de estilo CSS que se van modificar durante la animación.

- *Opciones*, para establecer la duración de la animación, la velocidad de transición y funciones para ejecutar según se desee: al principio y final de la ejecución, cada vez que se ejecute un cambio en el elemento animado, cuando se produzca algún error y no se pueda completar la animación...

En esta ocasión, se ha decidido cambiar el valor de la opacidad durante unos segundos para simular el efecto de parpadeo.

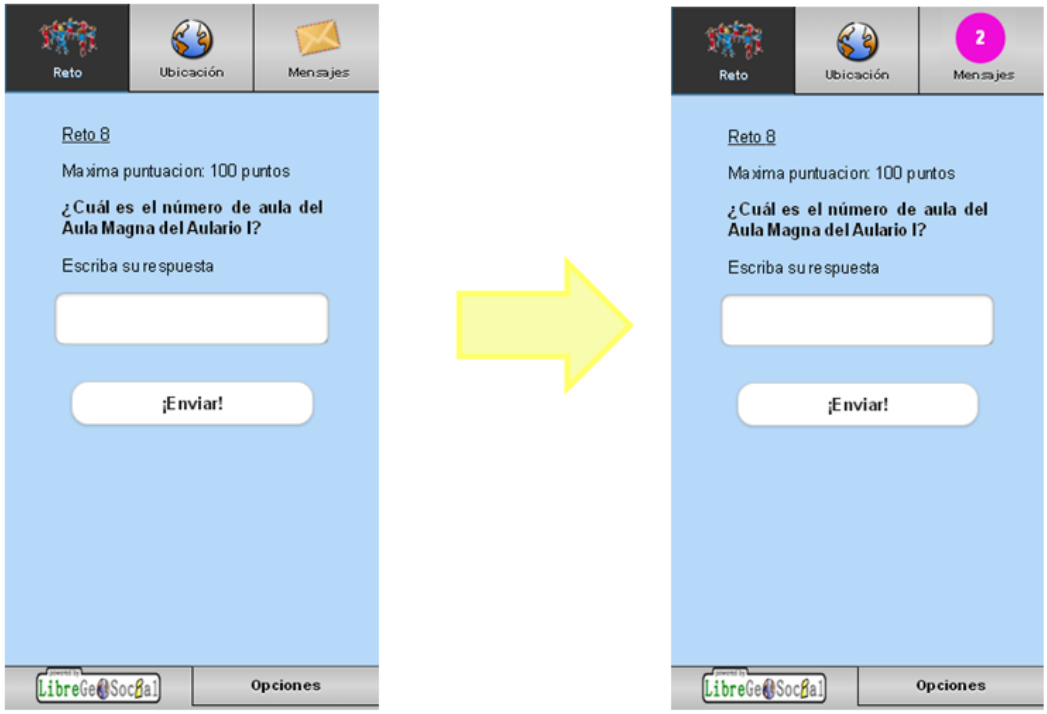


Figura 5.33: Notificación de un nuevo mensaje en la interfaz web del Jugador.

También se pensó que para notificar al usuario se emitiera un tono de alerta. Pero quedó descartado ya que los navegadores, por defecto, no tienen permitido reproducir sonidos automáticamente para así ahorrar en ancho de banda. La única manera de reproducirlos es que el usuario lo autorice en el mismo momento en que lo quiere escuchar, por lo que en este caso no tiene ningún sentido utilizarlo.

Lo que sí se ha podido realizar para notar la alerta, además de visualmente, es que el dispositivo vibre. Para ello se ha utilizado la API de Vibración de HTML5, aunque sólo es soportada por algunos navegadores web tales como Firefox, Chrome, Navegadores de Android y Opera (salvo Opera Mini)[3]. Para que el dispositivo del usuario vibre durante un corto periodo de tiempo se ha utilizado las siguientes líneas de código:

```
if ('vibrate' in navigator){
   navigator.vibrate([80,40,80]);
}
```

Con estas líneas primero se comprueba si el navegador soporta la API de Vibración y, en caso afirmativo, se procede a ejecutar la vibración. Para ello se hace uso de la función `.vibrate()` que toma un solo parámetro el cual puede ser:

- Un entero que indica el intervalo de tiempo que dura la vibración.
- Un array de enteros de cualquier longitud que describen que las vibraciones serán por períodos alternados, es decir, los enteros que estén en posición impar serán los tiempos en los que haya vibración, mientras que los que estén en posición par serán los tiempos en los que no haya.

Ahora se analizará el uso directo del servicio de mensajería de la interfaz web del Jugador. Cuando éste pulse en la pestaña de *"Mensajes"*, le aparecerá un menú como el de la figura 5.34 con las opciones disponibles:

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

Cuando el usuario pulse la primera opción para redactar un nuevo mensaje aparecerá una pantalla como la de la figura 5.35 donde podrá seleccionar, además, al destinatario de los mensajes. Y cuando pulse cualquiera de las otras opciones, se le mostrará la lista de mensajes correspondientes al botón pulsado como el de la figura 5.36(a). En esta última pantalla, aparecerá toda

[3] http://caniuse.com/#feat=vibration

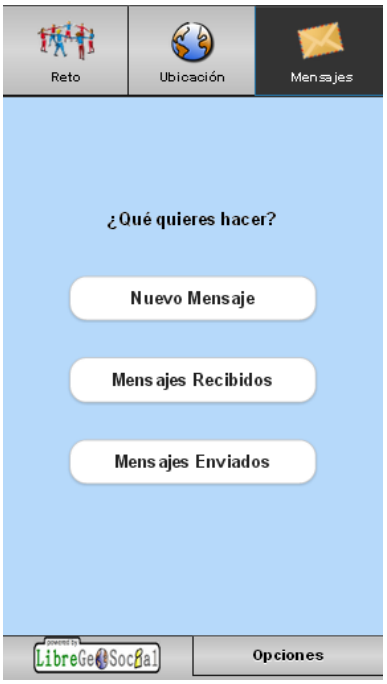


Figura 5.34: Acciones que puede realizar el Jugador con el servicio de mensajería.

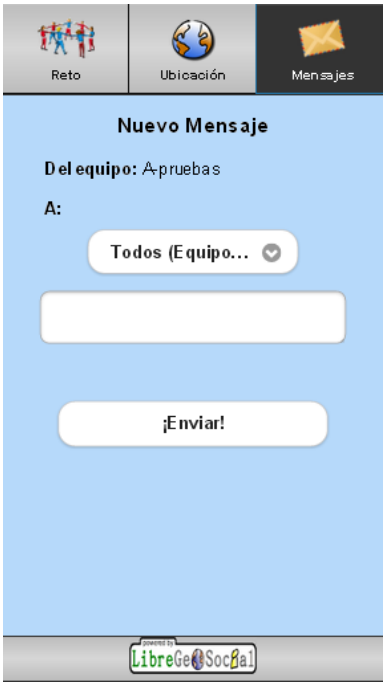
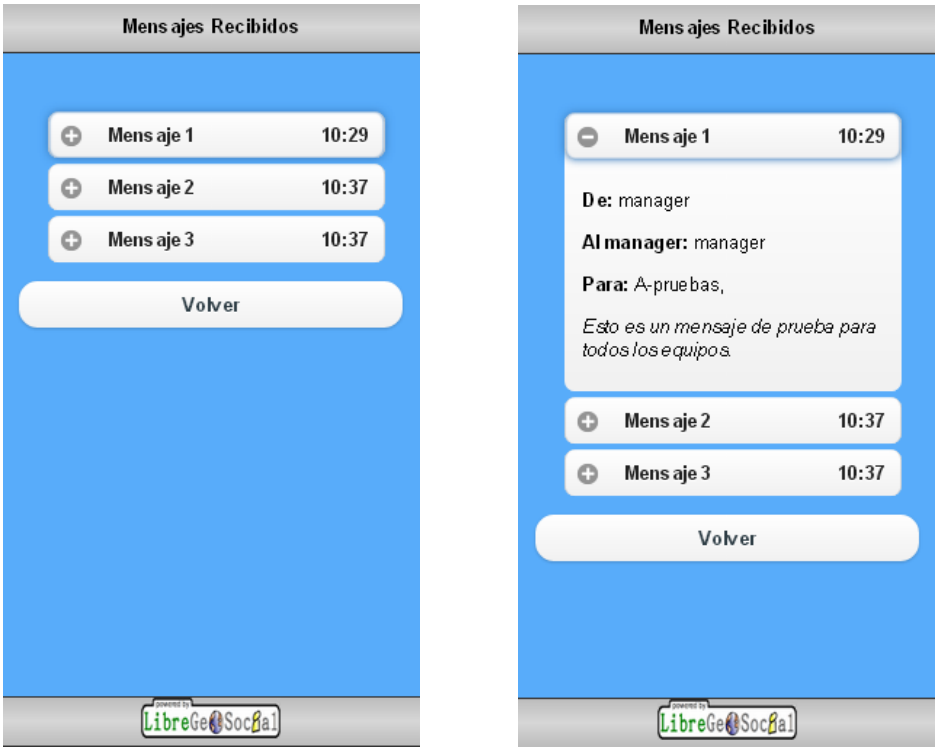


Figura 5.35: Formulario para que el Jugador envíe un mensaje.

la información relativa a cada uno de los mensajes: el remitente y los destinatarios del mensaje, la hora y el propio texto del mensaje. Para mostrar toda la información correspondiente a cada mensaje, el usuario tendrá que pulsar en el elemento de la lista que desee y aparecerá un desplegable con el contenido, tal y como aparece en la figura 5.36(b).



(a) Lista con los mensajes recibidos. (b) Despegable de un mensaje recibido.

Figura 5.36: Pantallas para visualizar los mensajes recibidos y enviados.

Para implementar esta última acción se ha utilizado de nuevo el atributo *"data-role"* dentro de un *"div"* y con valor *"collapsible"*. Como aparece en el código 5.7, dentro de este nodo primero se especifica la cabecera con el número y hora del mensaje usando la etiqueta *<h3>* que será lo que muestre por pantalla inicialmente, y después se describe el contenido del mensaje, que en principio permanecerá oculto. jQuery será el encargado de mostrar u ocultar la información del mensaje correspondiente cada vez que el usuario pulse la cabecera.

```
<div data-role="collapsible" id="sms">

  <h3>Mensaje {{forloop.counter}}<span class="ui-li-aside">{{
     sms.date|time:"H:i"}}</span></h3>
```

```
  {% if sms.from_manager != None %}
   <p><strong>De: </strong>{{sms.from_manager.user.username}}</
      p>

  {% elif sms.from_team != None %}
   <p><strong>De: </strong>{{sms.from_team.group.name}}</p>
  {% endif %}

  {% if sms.to_manager != None %}
   <p><strong>Al manager:</strong> {{sms.to_manager.user}}</p>
  {% endif %}

  {% if sms.to_team != None %}
    <p><strong>Para:</strong>
    {% for team in sms.to_team.all %}
     {{team.group.name}},
    {% endfor %}
    </p>
  {% endif %}

  <p><i>{{sms.text|linebreaks}}</i></p>

</div>
```

Código 5.7: Código para crear mensajes con contenidos despegables.

Capítulo 6

Resultados

En este capítulo se presentarán brevemente los resultados obtenidos de algunas de las pruebas realizadas con el fin de garantizar el buen funcionamiento del presente Proyecto Fin de Carrera.

6.1. Primer experimento.

En el primer experimento no se realizó ninguna prueba sobre el nuevo sistema, sino que fue una primera toma de contacto con el antiguo sistema para entender el funcionamiento de la aplicación y así poder implementar su actualización. Otro objetivo fue el de probar la aplicación Android con la nueva dirección IP y puerto del servidor.

6.1.1. Desarrollo.

La prueba tuvo lugar el día 8 de Abril de 2014 en el Campus de Fuenlabrada de la Universidad Rey Juan Carlos. Se presentaron un total de veinte personas para participar en una gymkhana previamente configurada, más dos encargados del control y monitorización de dicho evento. Los voluntarios fueron unos estudiantes de 4º de la E.S.O que realizaron una visita al campus. Se formaron cinco equipos de cuatro personas que ya poseían móviles Android.

Se plantearon un total de doce 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. Los retos se diseñaron

para que los voluntarios no tuvieran que salir del recinto del Campus de Fuenlabrada de la Universidad Rey Juan Carlos.

El experimento comenzó tras la llegada de los citados al Laboratorio 007 del Edificio de Laboratorios II, a los cuales se les presentó el proyecto y se les explicó el funcionamiento de la aplicación. Tras la breve introducción al sistema, los equipos empezaron su participación en la gymkhana, mientras que los organizadores controlaban las acciones de los equipos: su ubicación, sus respuestas a los retos y los mensajes intercambiados. Una vez que todos los equipos completaron todos los retos a superar, volvieron al laboratorio para dar sus opiniones sobre la aplicación y mostrarles cómo el organizador del evento realizó el seguimiento del evento desde su interfaz web.

6.1.2. Resultados.

A todos los voluntarios les funcionó bien la aplicación, por lo que no hubo problemas con la nueva IP y el nuevo puerto del servidor. También se probó que la aplicación y el servidor siguen funcionando correctamente.

Los voluntarios avisaron de que habían enviado mensajes al mánager de la gymkhana pero no habían recibido respuesta. Esto se debía a que en la monitorización de la gymkhana no aparecía la lista de mensajes ya que cuando se hacía una petición con XMLHttpRequest al servidor para pedir la lista de dichos mensajes, la función `.onreadystatechange()` asociada se llamaba dos veces: la primera llamada contiene la lista pero la segunda no, por lo que no mostraba ningún mensaje por pantalla. Esto se cambió en el nuevo sistema del presente Proyecto Fin de Carrera usando la tecnología AJAX a través del método `.ajax()` de jQuery como se explicó en el apartado 5.4.

6.2. Segundo experimento.

La segunda prueba que se realizó ya si fue con el nuevo sistema implementado en el presente proyecto. Los objetivos de este experimento fueron:

- Comprobar el buen funcionamiento de la nueva aplicación web.

- Confirmar que la interfaz web del Jugador se visualiza correctamente en cualquier dispositivo móvil.

- Comprobar que en la página web que monitoriza el evento se muestra, en todo momento, la información relativa a la gymkhana en curso.

6.2.1. Desarrollo.

El experimento tuvo lugar el día 3 de Noviembre de 2014 como parte de una de las actividades programadas para la decimocuarta edición de la Semana de la Ciencia en el Campus de Fuenlabrada de la Universidad Rey Juan Carlos. Acudieron al evento un total de 23 personas, aproximadamente, para participar en la gymkhana configurada para tal evento, más dos encargados de la monitorización de dicho juego. Los voluntarios eran estudiantes de los grados de Ingeniería de Telecomunicación, Enfermería, Aministración y Dirección de Empresas y Educación Primaria, que se distribuyeron en ocho equipos de varios miembros en cada uno de ellos.

Cada equipo ya contaba con varios dispositivos para poder participar en la gymkhana. Los sistemas operativos de los móviles fueron los más comunes: Android, Windows Phone e iOS, y los navegadores web que se utilizaron fueron los navegadores predeterminados de cada dispositivo y Google Chrome.

En esta ocasión se plantearon un total de quince pruebas 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. Y, al igual que el anterior experimento, se diseñaron para que los voluntarios no tuvieran que salir del recinto del Campus de la Universidad.

El experimento tomó partida en el Laboratorio 007 del Edificio de Laboratorios II presentando brevemente la aplicación desarrollada y su necesidad de realizar ese evento para comprobar que todo funcionaba correctamente. Además se explicó el manejo de la aplicación y pautas a seguir en caso de error. Tras esa breve introducción se les dio la URL con la que accederían a la aplicación web para poder empezar a jugar, y una vez completados todos los retos volvieron al laboratorio para dar sus opiniones sobre la aplicación y comentar algunos errores.

6.2.2. Resultados.

Comenzando la gymkhana, surgió el primer problema. A la mayoría de los equipos no les cargaba la página web correctamente debido a que no se pudo conectar al servicio CDN de jQuery donde están almacenadas las librerías, las cuales se encargan del estilo y funcionamiento de la interfaz web. Esto pudo ser debido a dos razones:

- Que el servicio estuviera caído y, por lo tanto, no se pudiera conectar con él.
- Que las librerías del CDN estuvieran bloqueadas por un filtro o un proxy.

Se supuso que fue la primera de las razones ya que después de que terminara el evento se volvió a acceder a la aplicación para averiguar el fallo, utilizando la barra de herramientas de Google Chrome. Se comprobó que se cargaban todas las librerías y, por tanto, la aplicación web funcionaba correctamente. Aun así, como ya se comentó en el apartado 5.1, se ha decidido que si no se encuentran disponibles esas librerías remotamente, se carguen de la versión hospedada en el servidor de la gymkhana.

El otro problema surgió en los retos de geolocalización, los cuales no mostraban correctamente en el mapa la posición de los equipos, haciendo que no se pudieran realizar dichos retos. Esto es debido a varios factores que afectan a la exactitud en la obtención de la localización del dispositivo usando la API de Geolocalización de HTML5. Uno de esos factores es el modo con el que el navegador web obtiene la posición del usuario.

- Puede utilizar el GPS del dispositivo móvil.
- También puede obtenerla teniendo en cuenta los puntos de acceso Wi-Fi.
- O triangulando la señal de varias torres de telefonía móvil cercanas.

Buscando información en Internet sobre cuál de estas opciones es la que se utiliza, se ha llegado a la conclusión de que no hay ninguna por defecto. Los navegadores web son los que deciden finalmente cuál de estas opciones escoger. Al ser esto así, se encontró una solución que mitiga este defecto, usar la exactitud incluida en la posición obtenida y, como se comentó en el apartado 5.3.3, calcular una nueva distancia.

Por el contrario, el servicio de mensajería funcionó correctamente tanto para los equipos como para la monitorización del mánager, gustando mucho entre los voluntarios.

6.3. Otros experimentos.

Las siguientes pruebas realizadas se desarrollaron fuera del entorno de la universidad.

Tras realizar los cambios necesarios para corregir los errores encontrados en los últimos experimentos, se establecieron como objetivos:

- Intentar que los retos se geolocalización fueran más fáciles de superar, ajustando la posición del usuario con el nivel de precisión correspondiente.
- Probar la nueva interfaz web del administrador de la gymkhana utilizando la nueva API JavaScript de Google Maps, y realizando las peticiones al servidor en segundo plano con AJAX.
- Comprobar que se cargan las librerías necesarias de jQuery para que se muestren bien las pantallas.

6.3.1. Resultados.

Tras la realización de diferentes pruebas se llegó a la conclusión de que el problema producido en el anterior experimento donde no se mostraban correctamente las páginas web, fue debido a que el CDN de jQuery estaba caído, ya que en todas las pruebas realizadas no ha habido ningún problema de ese tipo.

En cuanto a los retos de geolocalización, funcionaron mejor que en versiones anteriores aunque aun así la posición del usuario sigue sin ser exacta. Además, se ha comprobado que si están activados a la vez los datos móviles y el Wi-Fi, se consigue mayor precisión ya que combina la señal obtenida del navegador con los puntos de acceso Wi-Fi cercanos.

Por último, la interfaz web del mánager funciona correctamente, la lista de mensajes y de respuestas permanecen siempre visibles una vez se han solicitado al servidor, y se van actualizando a medida que se van recibiendo mensajes nuevos y respondiendo los retos. Lo mismo pasa con la tabla de estadísticas de cada equipo y el mapa con las posiciones de cada uno de éstos.

Capítulo 7

Conclusiones

En este apartado se va a presentar un resumen de todo el trabajo realizado. Primero, se van a comparar los objetivos planteados en el Capítulo 2 con los resultados obtenidos, y se van a plantear posibles ideas a implementar en un futuro. Finalmente, se ofrecerá una visión global de lo aprendido durante la carrera, incluido el desarrollo del presente Proyecto Fin de Carrera.

7.1. Análisis de objetivos.

A lo largo de la presente memoria se ha tratado de mostrar cómo se han alcanzado todos los objetivos planteados en el Capítulo 2. Ahora es momento de analizar en qué medida se han logrado.

El objetivo principal que se perseguía era el de migrar la antigua aplicación Android a una aplicación web accesible desde cualquier tipo de navegador, objetivo que se ha satisfecho en su gran mayoría. La única parte que no ha logrado conseguir al 100 % es conseguir la posición exacta del usuario que está participando en la gymkhana. Pero teniendo en cuenta, que todavía no se puede controlar el origen (GPS, Wi-Fi o antenas de telefonía) desde donde se obtiene dicha posición, se ha dado con una solución que ha resultado muy efectiva. Por lo que se puede concluir que el objetivo principal del presente Proyecto Fin de Carrera se ha cumplido exitosamente.

También se ha logrado que la interfaz web del administrador sea accesible y adecuadamente visible desde cualquier dispositivo, móvil o no. De esta manera se le dota de cierta movilidad

para configurar los retos, si así lo desea, desde cualquier lugar, no sólo con un ordenador.

Otra meta que se ha conseguido es que esta aplicación sea fácil de usar, por lo que se ha mejorado el canal de aprendizaje, *mobile learning* o *m-learning*, que ofrecía el antiguo sistema. Antes estaba restringido a usuarios con dispositivos Android ya que por entonces era el sistema operativo en auge. Pero actualmente, existen una gran variedad de sistemas operativos y navegadores web, por lo que era necesario realizar esta actualización del sistema y así expandirse.

Por el contrario, el único objetivo que no se ha podido alcanzar es la optimización del software para que el consumo de batería no fuera excesivo. Lo habitual es que una aplicación nativa consuma menos que una aplicación web si no hace uso de una conexión a Internet. En este sistema tanto la aplicación Android como la aplicación web necesitan el acceso a la red para poder funcionar, pero es cierto que ésta última consume más datos. Sin embargo, con la aplicación web el acceso a determinados recursos del dispositivo (audio, vibración, gps?) es limitado no pudiendo utilizarse en algunos casos, por lo que esta aplicación no consumiría tanta batería como la de Android. A modo de conclusión, las aplicaciones de este sistema están compensadas respecto al uso de batería[13-15].

7.2. Trabajos futuros

Un sistema nunca está completamente desarrollado, siempre estará en continua evolución a medida que nuevas tecnologías vayan surgiendo. De hecho, el presente Proyecto Fin de Carrera se basa en esta afirmación. Por esto, en este apartado se van a introducir algunas mejoras o nuevas ideas a desarrollar en un futuro.

- Asegurar la exactitud de la posición geográfica del dispositivo. La mayoría de los dispositivos móviles actuales cuentan con un chip GPS, por lo que no es de extrañar que un futuro, surjan nuevas APIs JavaScript con las que se pueda acceder a dicho sistema, previamente autorizado por el usuario, y así conseguir una localización geográfica más precisa. También se podría dar al desarrollador la opción de escoger, o priorizar, cuál quiere que sea el origen (GPS, Wi-Fi o antenas de telefonía móviles) del que quiere obtener la posición.

- Poder reproducir audio libremente. Si los navegadores web tuvieran una opción para autorizar la reproducción automática de sonidos, al igual que para permitir obtener la ubicación del dispositivo, se podría emitir un sonido cada vez que el usuario reciba un mensaje de la aplicación.

- Aumentar la variedad de tipos de retos. A modo de ejemplo, se podrían añadir retos tipo test, en los que se presentara un enunciado y varias posibles soluciones, o retos en los que para pasar de reto se tuviera que resolver un puzzle También se podrían combinar los retos de geolocalización con retos de orientación en los que se presentara un enunciado con el sitio al que los equipos tuvieran que dirigirse, pero en lugar de utilizar un mapa, usaran una brújula.

- Conseguir inmediatez en el servicio de mensajería interno. Para ello se puede utilizar la librería *Server-Sent Events* o SSE de HTML5, en la parte del cliente, para subscribirse a un canal en el servidor, y que cada vez que se reciba un mensaje para él, se le envíe inmediatamente. Pero para ello, se necesitaría que el servidor pudiera manejar dichos eventos. Esto se podría conseguir utilizando librerías como *gevent*. De esta manera, surgirían dos mejoras: que sólo se enviaría el mensaje, por lo que disminuiría los datos intercambiados, y que el cliente no tuviera que estar solicitando y procesando cada cierto tiempo la lista de todos los mensajes intercambiados

7.3. Lecciones aprendidas.

A lo largo de estos años de carrera se han aprendido varios aspectos sobre sistemas de telecomunicación, electrónica y telemática. Concretamente, este proyecto está centrado en la telemática. Éste se ha basado en conocimientos adquiridos, en su mayoría, de asignaturas como *Información Audiovisual en Redes de Ordenadores* cuando se aprendió Python, con *Sistemas Telemáticos II* se obtuvieron las primeras nociones de HTML, y *Servicios y Aplicaciones Telemáticas* cuando se unieron ambas tecnologías y se ampliaron conocimientos.

En base a esto, se ha conseguido desarrollar una aplicación web utilizando un servidor ya previamente implementado. Al desarrollar este proyecto se han podido utilizar tecnologías aprendidas durante la carrera como es implementar un servidor web Django con Python, y se

han aprendido nuevas versiones de lenguajes como HTML o CSS. Aunque es cierto que el servidor ya estaba implementado, para mantener la antigua aplicación Android y desarrollar la nueva aplicación web, se han tenido que crear nuevos métodos similares a los que ya estaban pero con algunas diferencias ya que, en la mayoría de los casos, no había que enviar datos por JSON sino que se creaba directamente la página web. Por lo tanto, en lugar de procesar la información en el cliente, se procesa directamente en el servidor.

Por otro lado, para implementar el cliente se han ampliado los conocimientos en HTML y CSS utilizando sus últimas versiones del lenguaje. Y se han aprendido otras tecnologías como JavaScript y jQuery. Estos nuevos conocimientos se han adquirido con la ayuda de libros[3-6] y con las documentaciones de dichas tecnologías en Internet, cuyas referencias se encuentran en el apartado Bibliografía de esta memoria.

Finalmente, se ha aprendido como desarrollar una aplicación web dedicada a cualquier usuario. Hasta ahora todos los trabajos realizados durante la carrera iban destinados al alumno que los desarrollaba y al profesor de la asignatura, pero la realización de este proyecto significaba expandir el número de destinatarios, y mostrar una interfaz web que fuera de fácil uso para cualquier tipo de usuario.

Bibliografía

[1] Jorge Fernández González, *DISEÑO, IMPLEMENTACIÓN Y DESPLIEGUE DE UN SERVICIO DE TELECOMUNICACIÓN PARA M-LEARNING EN MÓVILES ANDROID: GYMKHANAS DE NUEVA GENERACIÓN*, ETSIT URJC, 2010.

[2] Django Arquitecture, https://docs.djangoproject.com/en/1.8/faq/general/#faq-mtv

[3] J.M. Gustafson, *HTML5 Web Application Development By Example Beginner's guide*, Packt Publishing Ltd, 2013.

[4] Raymond Camden,Andy Matthews, *jQuery Mobile Web Development Essentials*, Packt Publishing Ltd, *Second Edition*, 2013.

[5] Jonathan Chaffer, Karl Swedberg, *Learning jQuery*, Packt Publishing Ltd, *Fourth Edition*, 2013.

[6] Jack Franklin, *Beginning jQuery*, Apres, *Fourth Edition*, 2013.

[7] Desarrollo web HTML5, *W3Schools*, http://www.w3schools.com/html/html5_intro.asp

[8] jQuery Mobile Demos, *jQuery Mobile Demos*, http://demos.jquerymobile.com/1.3.2/

[9] API de jQuery, *jQuery API Documentation*, https://api.jquery.com/

[10] Soporte de Tenologías, http://caniuse.com/ y http://jquery.com/browser-support/

[11] API de Geolocalización, *Mozilla Developers Network*, https://developer.mozilla.org/es/docs/WebAPI/Using_geolocation

[12] API JavaScript de Google, *Google Developers*, https://developers.google.com/maps/documentation/javascript/reference?hl=es

[13] http://www.accensit.com/index.php/en/accensit-blog-en/150-mobile-platforms.html

[14] http://radar.oreilly.com/2013/05/measuring-the-impact-of-web-page-structures-on-battery-usage-in-mobile-devices.html

[15] http://www.ideup.com/blog/estrategia-movil-desarrollo-de-una-app-nativa-web-app-o-soluciones-hibridas

[16] Emuladores online, *BrowserStack*, https://www.browserstack.com/