MyConference: Sistema informático para la configuración de congresos y eventos profesionales

Sistemas Informáticos
Facultad de Informática
Universidad Complutense de Madrid

Daniel Escoz Solana
Samuel Méndez Galán
Pedro Morgado Alarcón

Directoras:
Dra. Guadalupe Miñana Ropero
Dra. Victoria López López

Junio 2014

MyConference: Sistema informático para la configuración de congresos y eventos profesionales

Sistemas Informáticos
Facultad de Informática
Universidad Complutense de Madrid

Junio 2014

Autorización de Difusión

a 23 de junio de 2014,

Los abajo firmantes, matriculados en Sistemas Informáticos de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente proyecto fin de carrera de Sistemas Informáticos: "MyConference: Sistema informático para la configuración de congresos y eventos profesionales", realizado durante el curso académico 2013-2014 bajo la dirección de la Dra. Guadalupe Miñana Ropero y de la Dra. Victoria López López con la colaboración de la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

DANIEL ESCOZ SOLANA, SAMUEL MÉNDEZ GALÁN, PEDRO MORGADO ALARCÓN

Agradecimientos

A nuestras familias y amigos, por todo el apoyo que nos han mostrado durante esta etapa tan importante en nuestras vidas.

Resumen

MyConference es un sistema informático formado por varias aplicaciones orientado a la gestión de congresos y eventos profesionales.

Su objetivo es servir como punto común de gestión y visualización de un congreso tanto para los organizadores como para los asistentes. A los organizadores les permite disponer de una plataforma en la que introducir toda la información necesaria para describir el congreso, ya sean ponentes de las conferencias o el público que asiste al evento. A los asistentes les permite tener en su bolsillo una herramienta que pueden consultar en cualquier momento para saber dónde ir, cual es la siguiente conferencia o de qué temas van a tratar las ponencias para decidir a cuáles ir.

Para conseguir este objetivo hemos diseñado e implementado una aplicación para cada tipo de usuario de un congreso: un formulario web para los organizadores y una aplicación móvil para los asistentes, ambas con los mismos principios de sencillez y claridad en su presentación y uso.

Palabras clave: Congresos, Gestión, Aplicación, Servicio Web, Android, MongoDB, RESTful API.

Abstract

MyConference is a system composed of multiple applications oriented towards the management of conferences and professional events.

Its goal is to serve as a common place for management and visualization of a conference, both for organizers and for the audience. For organizers it allows them to use a platform in which to enter every bit of information needed for the conference, be it speakers or conference audience. For the audience, it lets them have a tool in their pockets they can check any time to know where to go, what's the next event or which topics are to be treated in keynotes so they can decide which to go to.

To fulfill this goal, we designed and implemented an application for both types of users: a set of web forms for organizers and a mobile application for the audience, both using the same simplicity and clarity principles for usage and presentation.

Keywords: Conferences, Management, Application, Web Service, Android, MongoDB, RESTful API.

Índice

Agradecimientos v

Resumen vii

Abstract ix

1. Introducción 1

1.1. Presentación y motivación 1

1.2. Estado del Arte 2

1.2.1. TED Connect 2

1.3. Organización de la Memoria 3

2. Tecnologías Utilizadas 5

2.1. Alojamiento del Servicio Web 5

2.1.1. Heroku 5

2.2. Desplegado del Servicio Web 6

2.2.1. Node.js 6

2.2.2. Jade 7

2.2.3. SASS/SCSS 8

2.2.4. Bootstrap 8

2.2.5. Google Maps 9

2.2.6. Transloadit 9

2.3. Base de Datos 9

2.3.1. MongoDB 9

2.3.2. SQLite 10

2.4. Control de Versiones 10

2.4.1. Git 11

2.4.2. GitHub . . . 11
2.5. Herramientas de Desarrollo . . . 11
2.5.1. Sublime Text . . . 11
2.5.2. Eclipse . . . 12
2.5.3. Android SDK . . . 12
2.6. Otras Herramientas . . . 13
2.6.1. Foreman . . . 13

3. Servicio Web y Base de Datos **15**
3.1. API . . . 16
3.1.1. Servidor Web . . . 16
3.1.2. Acceso . . . 17
3.1.3. Autenticación . . . 18
3.1.4. Servicios Disponibles . . . 19
3.2. Base de Datos . . . 26
3.2.1. Entidades Fundamentales . . . 27
3.2.2. Entidades de Apoyo . . . 27
3.3. Web . . . 28
3.3.1. Archivos de configuración . . . 29
3.3.2. Rutas . . . 29
3.3.3. Interfaz Gráfica . . . 29

4. Aplicación Móvil **31**
4.1. Android . . . 31
4.2. Desarrollo de la Aplicación Móvil . . . 33
4.2.1. Estructura . . . 33
4.2.2. Librerías externas . . . 36
4.3. Base de datos interna . . . 40
4.4. SharedPreferences . . . 43
4.5. Permisos . . . 43

5. Casos de Uso **49**
5.1. Casos de Uso del Formulario Web . . . 49
5.2. Casos de Uso de la Aplicación Móvil . . . 63

6. Pruebas y Resultados **75**

6.1. Prototipo 75

6.2. Prueba de la aplicación en el PIC 2014 76

7. Trabajo Futuro **79**

7.1. Aplicación Móvil para iOS 79

7.1.1. Requisitos 80

A. Guía de Despliegue **83**

A.1. Despliegue de la API 84

A.1.1. Configuración con Foreman 84

A.1.2. Configuración de MongoDB 85

A.1.3. Configuración de Mandrill 85

A.1.4. Configuración General 86

A.1.5. Heroku 86

A.2. Despliegue de la Web 88

A.2.1. Configuración de MongoDB 88

A.2.2. Configuración de Transloadit 88

A.2.3. Configuración General 89

A.2.4. Heroku 90

A.3. Despliegue de la App Android 91

B. Guía de uso **93**

B.1. Acceso a la aplicación 93

B.1.1. Registro 93

B.1.2. Inicio de sesión 94

B.2. Visualización de congresos 94

B.2.1. Usuarios 96

B.2.2. Anuncios 96

B.2.3. Documentos 97

B.2.4. Lugares 97

B.2.5. Organizadores 98

B.2.6. Ponentes 98

B.2.7. Agenda 99

B.3. Creación de congresos 99

B.3.1. Usuarios 100
B.3.2. Anuncios 101
B.3.3. Documentos 102
B.3.4. Lugares 102
B.3.5. Organizadores 103
B.3.6. Ponentes 104
B.3.7. Agenda 104

Bibliografía **107**

Índice de figuras

1.1. Diagrama de la arquitectura del sistema 2
1.2. Captura de la aplicación TEDConnect. 3

2.1. Ejemplo de código escrito en **SASS** traducido a **CSS**. 8

3.1. Diagrama de la arquitectura relevante al Servicio Web 15
3.2. Diagrama de la arquitectura relevante a la API 16
3.3. Diagrama del Proceso de Registro 20
3.4. Diagrama del Proceso de Autenticación 21
3.5. Diagrama de la arquitectura relevante a la Base de Datos 26
3.6. Diagrama de la arquitectura relevante a la Web 28

4.1. Diagrama de la arquitectura relevante a la aplicación android 31
4.2. *Action Bar* en nuestra app gracias a `ActionBarSherlock` 36
4.3. Agenda de un congreso con `AmazingListView` 37
4.4. Pantalla de la aplicación con los lugares relevantes de un congreso. . 39
4.5. Relación entre las clases del paquete es.ucm.myconference. 45
4.6. UML del paquete es.ucm.myconference.accountmanager 46
4.7. Diagrama UML de la clase `NavigationDrawerActivity`. 47

6.1. Pantallas del prototipo para las Jornadas de Gerencia Universitaria . 76
6.2. Capturas de la aplicación para el PIC 2014 77

7.1. Capturas de pantalla de la aplicación iOS 79
7.2. Logo de Xcode 80

B.1. Página inicial de la web. 93
B.2. Formulario de *registro*. 94

B.3. Formulario de *inicio de sesión*. 94
B.4. Página con la lista de los congresos del usuario. 95
B.5. Página principal de un congreso. 95
B.6. Sección de *usuarios*. 96
B.7. Sección de *anuncios*. 97
B.8. Sección de *documentos*. 97
B.9. Sección de *lugares*. 98
B.10. Sección de *organizadores*. 99
B.11. Sección de *ponentes*. 100
B.12. Sección de la *agenda*. 100
B.13. Formulario de *creación de nuevos congresos*. 101
B.14. Formulario de *invitación de usuarios*. 101
B.15. Correo con el *código de activación*. 101
B.16. Formulario para añadir nuevos *anuncios*. 102
B.17. Formulario para añadir nuevos *documentos*. 102
B.18. Formulario para añadir nuevos *lugares*. 103
B.19. Formulario para añadir nuevos *organizadores*. 104
B.20. Formulario para añadir nuevos *ponentes*. 105
B.21. Formulario para añadir nuevos *eventos* en la agenda. 105

Capítulo 1

Introducción

1.1. Presentación y motivación

MyConference es un conjunto de aplicaciones orientado a la gestión de congresos y eventos profesionales que utiliza un servicio web y una aplicación móvil para realizar sus funciones.

La motivación adherente a este proyecto surge de la llegada de los dispositivos móviles a nuestros hogares. También ha sido fundamental el lanzamiento del sistema operativo móvil libre y de código abierto (*Android*) y la gran utilidad que tiene, hoy por hoy, un *Smartphone* en la vida diaria. Esto ha provocado que nuestro interés en desarrollar una aplicación bajo esta plataforma sea inmenso y haya desembocado en el desarrollo de este proyecto.

El desarrollo de aplicaciones móviles para dispositivos *Android* no es una temática impartida en ninguna asignatura de los planes de estudio que hemos cursado, pero las asignaturas que hemos estudiado en la carrera a lo largo de estos años nos han dado una base de conocimientos suficientemente amplia para poder embarcarnos en un proyecto de este tipo.

Las nuevas tecnologías nos han permitido tener internet en el bolsillo, esto ha provocado fuertes cambios en la sociedad y con ello la necesidad de adaptar y modernizar las ciudades en todos sus servicios y sectores. En este nuevo escenario, las universidades quieren adaptar la organización de sus congresos a estas nuevas tecnologías, siendo así marco de referencia para el resto de sectores.

Actualmente la manera que tienen los congresos de comunicarse con el ciudadano es a través de webs propias, cada una con un estilo diferente y con formas de proporcionar información de maneras muy distintas.

Lo que nos motivó a realizar este proyecto fue el poder aportar nuestros conocimientos de la carrera a la comunidad universitaria y científica proporcionándoles una herramienta única y sencilla que les haga más fácil la vida a la hora de asistir a un congreso. Esta aplicación no pretende sustituir la web de los congresos, sino más bien introducir las nuevas tecnologías para que los asistentes a un congreso puedan

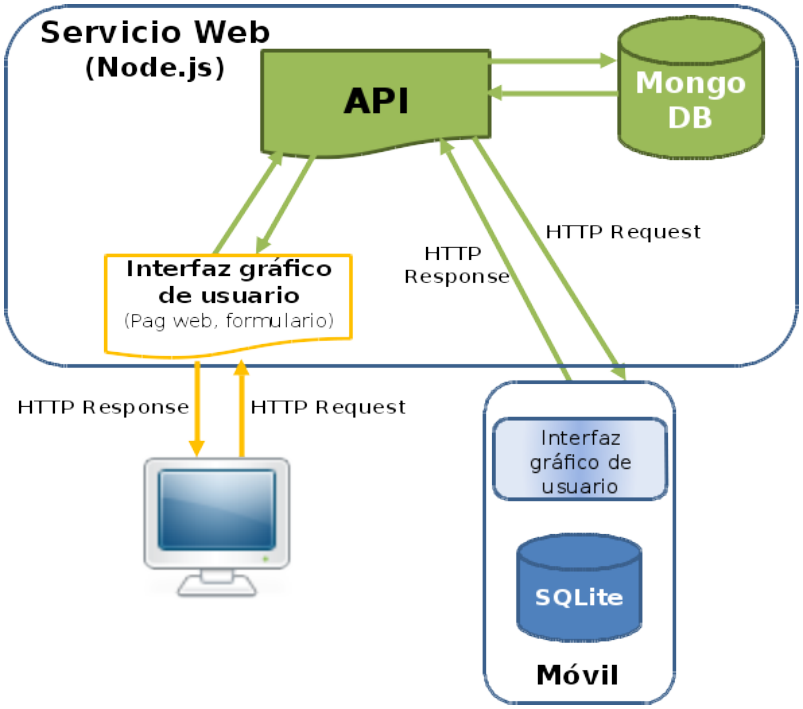


Figura 1.1: Diagrama de la arquitectura del sistema

obtener de una manera más fácil y cómoda la información y novedades del congreso por parte de los asistentes.

El resultado de este proyecto ha sido una herramienta integral y fiable, que por un lado permite al organizador de un congreso introducir de una manera gráfica y amigable toda la información que un asistente necesita saber, y por otro lado proporciona al asistente a un congreso tener en su dispositivo móvil toda la información sobre ese congreso u otros a los que vaya a asistir sin tener que lidiar con varias páginas web que pueden no ser aptas para su visualización en este tipo de dispositivos.

La figura 1.1 muestra un diagrama de la arquitectura global del sistema.

El proyecto se ha desarrollado cuidando especialmente el rendimiento del dispositivo en términos de batería y consumo de datos, así como su visualización en diferentes terminales. Como resultado se ha obtenido una aplicación móvil y una web amigables y óptimas que permiten controlar toda la información de una manera rápida, cómoda y fácil de usar por usuarios no expertos.

1.2. Estado del Arte

1.2.1. TED Connect

TEDConnect es una aplicación para Android que se encarga de presentar la información de las charlas que ellos organizan [4].

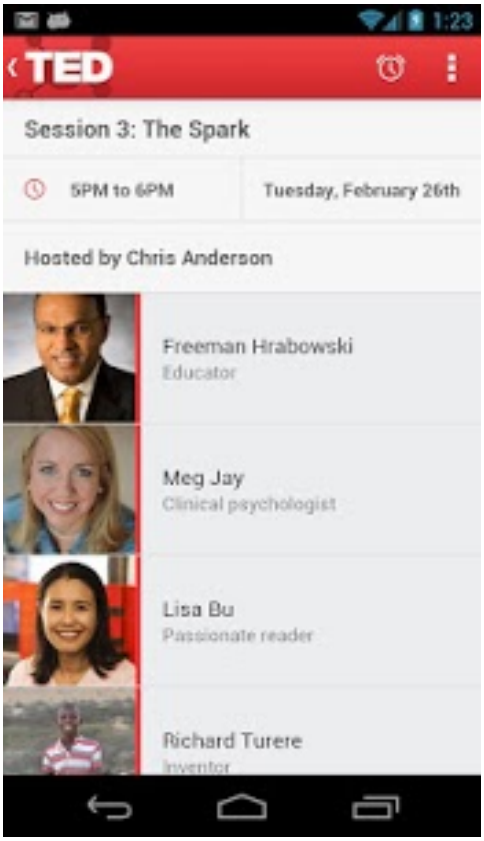


Figura 1.2: Captura de la aplicación TEDConnect.

Se pueden ver los asistentes y los ponentes, enviar mensajes entre ellos, tener el programa de todas las charlas, comidas y actividades y ver el mapa de las localizaciones de las mismas.

La organización y la presentación de los contenidos era una buena base para desarrollar nuestra aplicación. Una pantalla de la aplicación se puede ver en la figura 1.2.

1.3. Organización de la Memoria

Esta memoria está organizada de la siguiente manera:

- El presente capítulo introductorio donde se encuentra una breve explicación de qué es *MyConference*, la idea y motivación por la cuál se desarrolló y sus aspectos fundamentales.
- El segundo capítulo habla sobre las tecnologías utilizadas para el desarrollo de la aplicación.
- El tercer capítulo explica la estructura del *Servicio Web*, tanto de la API como del formulario Web, y de la Base de Datos interna.

- El cuarto capítulo desarrolla la aplicación móvil en Android, explicando su estructura, las librerías externas utilizadas y la base de datos interna.
- El quinto capítulo explica el funcionamiento del formulario Web y de la aplicación móvil mediante *casos de uso*.
- El sexto capítulo muestra las pruebas que se han realizado de la aplicación, mostrando los resultados obtenidos de ellas.
- El séptimo capítulo habla sobre el trabajo futuro que se puede realizar en la aplicación para continuar con su desarrollo.

Además de estos capítulos, la memoria tiene los siguientes apéndices:

- El apéndice A muestra una *Guía de Despliegue* que explica como instalar y configurar todo lo necesario para poder continuar con el desarrollo de la aplicación, tanto para la API como para el formulario Web y la aplicación móvil.
- El apéndice B explica como funciona el formulario Web para la creación de los congresos mediante una *Guía de Uso*.

Capítulo 2

Tecnologías Utilizadas

2.1. Alojamiento del Servicio Web

MyConference hace uso de dos aplicaciones web para funcionar. Estas aplicaciones, de las que hablaremos posteriormente, están preparadas para trabajar en entornos de *Plataforma como servicio* (*PaaS* , Platform as a Service), que permite al desarrollador desplegar aplicaciones en la nube de manera sencilla y rápida sin tener que administrar la máquina en la que se instala. Suele identificarse como una evolución del *Software como servicio* (*SaaS*, Software as a Service), pero mientras que *SaaS* permite la ejecución de una aplicación concreta, *PaaS* permite al programador escribir aplicaciones para un tipo de plataforma concreto.

Para MyConference, el *PaaS* elegido ha sido Heroku, puesto que el despliegue así como la configuración de las aplicaciones es muy sencillo y flexible.

2.1.1. Heroku

Heroku [14] es una plataforma como servicio que soporta distintos lenguajes de programación.

Heroku inicialmente solo soportaba la ejecución de aplicaciones web desarrolladas en Ruby, pero posteriormente se extendió el soporte a Java, Node.js, Scala, Clojure, Python y PHP.

Una de las grandes ventajas de *Heroku* es poder obtener un Dyno (unidad de computación que ejecuta una aplicación) de manera gratuita. De esta manera se puede correr una aplicación web que consuma pocos recursos sin ningún tipo de gasto. Este sistema de Dynos es totalmente escalable, de tal manera que si la aplicación crece

y necesita más recursos es tan sencillo como ampliar la capacidad de los Dynos que necesita la aplicación.

Heroku dispone de una gran cantidad de extensiones (add-ons) que añaden funcionalidad extra a las aplicaciones. Por ejemplo, la extensión MongoLab permite el uso de bases de datos hechas con MongoDB.

Para el desarrollo de MyConference hemos usado dos Dynos gratuitos de Heroku: uno para la ejecución de la API y otro para la Web. De esta manera hemos obtenido de manera gratuita un servicio en el que poder desarrollar la aplicación y poder testearla de manera sencilla y rápida.

2.2. Desplegado del Servicio Web

2.2.1. Node.js

Node.js [18] es un entorno de programación basado en JavaScript. Es la tecnología usada para el desarrollo tanto de la API como de la Web.

Dispone de una gran cantidad de módulos que añaden funcionalidad al entorno. Algunos ejemplos de estos módulos son:

- Express: framework para el desarrollo de aplicaciones web.
- Winston: permite el guardado de todos los logs de una aplicación.
- Path: para el manejo de rutas dentro de los directorios de la aplicación.
- Mongoose: permite el manejo de manera sencilla de bases de datos hechas con MongoDB.

Estos módulos se pueden instalar de manera rápida con npm, gestor oficial de paquetes de Node.js. A partir de la versión 0.6.3 de *Node.js*, npm se instala automáticamente junto con el entorno. Su uso es muy simple, tan solo hay que usar el siguiente comando:

```
$ npm install package-name
```

Módulos Node.js

Express

Express [8] es un microframework web para Node.js fácilmente extendible y sencillo de usar.

Es el motor principal de la web.

Restify

Restify [22] es un microframework inspirado en Express, con una sintaxis y un uso prácticamente idéntico, pensado y diseñado para servicios REST en lugar de para webs tradicionales. Además incluye un cliente para realizar peticiones a servicos REST, escritos o no con Restify.

Es el motor principal de la API de MyConference en su versión para servidor y una parte muy importante de la web en la versión cliente.

Mongoose

Mongoose [17] es un *Object-Document Mapper*, una librería diseñada para hacer de puente entre una base de datos MongoDB y un lenguaje orientado a objetos, JavaScript en este caso. Nos permite trabajar con *modelos*, objetos que representan las diferentes colecciones de MongoDB y darles estructura y comportamiento, facilitando el desarrollo de aplicaciones que utilicen este tipo de base de datos.

Winston

Winston está diseñado para ser una librería simple y universal para guardar los logs de una aplicación. Permite separar los logs en diferentes niveles, de esta manera se puede elegir qué hacer con cada nivel, por ejemplo, guardarlos en base de datos, guardarlos en archivos locales o simplemente mostrarlos por consola.

2.2.2. Jade

Jade [15] es un motor de plantillas de alto rendimiento implementado en JavaScript para Node.js. Tiene una sintaxis limpia, sensible a los espacios en blanco para escribir código HTML.

2.2.3. SASS/SCSS

Sass [23] (Syntactically Awesome Stylesheets) es un lenguaje de hojas de estilos, una extensión de CSS al que añade mucha funcionalidad, como por ejemplo el uso de variables, reglas anidadas, mixins e imports inline.

Los archivos *Sass* suelen llevar la extensión .scss, pero también se puede utilizar la extensión .sass, aunque es menos común.

Para poder utilizar de estos ficheros, primero hay que compilarlos a archivos de CSS, no se pueden usar directamente los archivos `.scss`. En la figura 2.1 se muestra un ejemplo de código SASS traducido a CSS.

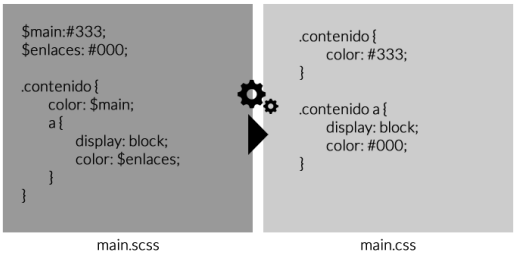


Figura 2.1: Ejemplo de código escrito en **SASS** traducido a **CSS**.

2.2.4. Bootstrap

Bootstrap [5] es uno de los frameworks front-end más populares para el desarrollo de aplicaciones web adaptables.

El diseño web adaptable (Responsive Web Design) consigue adaptar cualquier aplicación web a cualquier tipo de pantalla.

2.2.5. Google Maps

Google Maps [12] es, además de una aplicación web y móvil, un conjunto de librerías para mostrar los mapas de Google en nuestras aplicaciones. Para la web, dispone de una API JavaScript con utilidades para mostrar mapas, marcadores, información de dichos marcadores, etc. MyConference utiliza dichos mapas para visualizar la selección de la posición de un lugar.

2.2.6. Transloadit

Transloadit [26] se trata de un servicio de transformación y procesado de ficheros en general, imágenes en particular. Se utiliza en la web de MyConference para permitir la subida de imágenes.

2.3. Base de Datos

Como cualquier aplicación que maneje una gran cantidad de datos, MyConference hace uso de bases de datos para el almacenado, procesado y posterior recuperación de los datos.

Debido a la inicial incertidumbre acerca de cómo representaríamos los datos y de la prevista flexibilidad en la representación de éstos, optamos por utilizar una base de datos documental, en concreto MongoDB, para almacenar los datos del sistema. La aplicación móvil, por su lado, hace uso de una réplica local de los datos mediante SQLite.

2.3.1. MongoDB

MongoDB [16] es una base de datos documental, la más popular entre las *NoSQL* (bases de datos que no utilizan SQL como lenguaje de consulta y que generalmente utilizan otro tipo de estructuras).

Se trata de una base de datos organizada en colecciones. Cada colección contiene documentos, pequeños objetos representados mediante *BSON*, un formato binario basado en la estructura de JSON. Estos documentos no poseen estructura, sino que pueden contener cualquier tipo de campos y datos. Esto hace que sea una base de datos ideal para proyectos con datos poco estructurados, con estructuras flexibles o con una velocidad de cambio grande.

2.3.2. SQLite

SQLite [24] es una librería que implementa una base de datos autocontenida en un único fichero y que no requiere configuración ni servidores. Se trata de un tipo de base de datos ideal para ser usado de almacenamiento interno de datos o configuraciones para aplicaciones tanto móviles como de escritorio, y son muchos los projectos de software libre que la utilizan para este fin.

2.4. Control de Versiones

Escribir código es un proceso que produce errores constantemente y trabajar en grupo de forma independiente pero en los mismos ficheros es un proceso complejo si utilizamos herramientas sencillas (copiar-pegar, mover archivos), y puede dar lugar a muchos quebraderos de cabeza. Todos estos problemas se pueden resolver utilizando software de control de versiones. Las características básicas del control de versiones son:

- Registro completo de cambios en el código, permitiendo revertir el historial o un cambio concreto.
- Separación de *ramas*, permitiendo mantener múltiples niveles de estabilidad (versión estable, versión en desarrollo, versión experimental, etc.) en el mismo repositorio.
- Operaciones de integración entre varias ramas de código (múltiples programadores, múltiples versiones), de forma automática o manual.

Históricamente se ha utilizado software de control de versiones *centralizado*, en el que un servidor central almacena todo el código y las revisiones de este mientras los programadores obtienen una copia local únicamente de la última revisión de la rama en la que trabajan. Los cambios en el código, una vez empaquetados en *commits*, se envían directamente al servidor.

Hoy en día, sin embargo, la tendencia es a migrar a software de control de versiones *distribuído*, en el que no hay un servidor centralizado (aunque se suele utilizar un repositorio "principal" que juega el mismo rol). En su lugar, los diferentes programadores conservan en su máquina una copia completa de todo el historial de versiones de cada una de las ramas con las que trabajan. Este tipo de sistemas suelen tener una gestión de mezclas entre ramas mucho más potente, lo que permite que los cambios se integren generalmente sin problemas en las copias del resto de programadores o del repositorio central.

Para MyConference, siguiendo con esta tendencia, hemos elegido Git, un sistema de control de versiones distribuído ampliamente utilizado por proyectos tanto libres como propietarios, junto con el servicio de *hosting* de repositorios GitHub.

2.4.1. Git

Git [10] es un sistema de control de versiones distribuído diseñado para funcionar con proyectos tanto grandes como pequeños.

Se creó en el año 2005 como alternativa al sistema propietario *BitLocker* que se utilizaba para mantener el kernel Linux. Su desarrollo inicial lo realizó Linus Torvalds, responsable del kernel, en apenas unas semanas. Posteriormente ha crecido en tamaño y funcionalidad hasta la reciente liberación de la versión 2.0.0, pero sigue siendo un sistema con un buen rendimiento que no se degrada con el tamaño o la complejidad del historial de versiones o del código.

2.4.2. GitHub

GitHub [11] es un servicio de almacenamiento de repositorios Git que añade funcionalidad como un sistema de *issues*, wikis, gráficas y otras características complementarias al control de versiones en sí.

2.5. Herramientas de Desarrollo

Para crear MyConference se han utilizado multitud de tecnologías, pero todas ellas necesitan de unas herramientas básicas como editores o kits de desarrollo. A continuación se detallan los más importantes.

2.5.1. Sublime Text

Sublime Text [25] es un editor de texto multiplataforma orientado principalmente a la programación. Se trata de un editor rápido, potente y con multitud de *plug-ins* que permiten añadir casi cualquier funcionalidad imaginable.

Para MyConference se ha utilizado la versión 3, todavía en desarrollo pero muy estable.

2.5.2. Eclipse

Eclipse [7] es un *entorno de desarrollo integrado* (IDE, por sus siglas en inglés) utilizado para desarrollar aplicaciones.

Como características principales, dispone de un editor de texto con resaltado de sintaxis y autocompletado, la compilación es en tiempo real, incluye pruebas unitarias con JUnit, control de versiones, integración con Ant, asistentes para crear nuevos proyectos, clases, test... y refactorización. Asímismo, se le puede instalar gran cantidad de *plug-ins* para hacerlo aún más completo.

Para nuestro proyecto hemos utilizado la versión 3.6 denominada *Helios*.

2.5.3. Android SDK

Android SDK es un paquete que incluye APIs y las herramientas de desarrollo de Android necesarias para desarrollar, compilar, testear y debuguear apps.

También incluye el ADT (*Android Developer Tools*), que es un plugin para *Eclipse* que hace que el desarrollo sea mucho más sencillo, proporcionando una interfaz gráfica para las apps, menús contextuales para crear proyectos rápidamente, añadir los paquetes de Android de manera sencilla y generar los `.apk` necesarios para distribuir las apps.

2.6. Otras Herramientas

2.6.1. Foreman

Foreman [9] (no confundir con "The Foreman", otro proyecto similar) es un gestor de procesos para proyectos que requieren ejecutar múltiples scripts o múltiples instancias de un mismo script en paralelo.

Capítulo 3

Servicio Web y Base de Datos

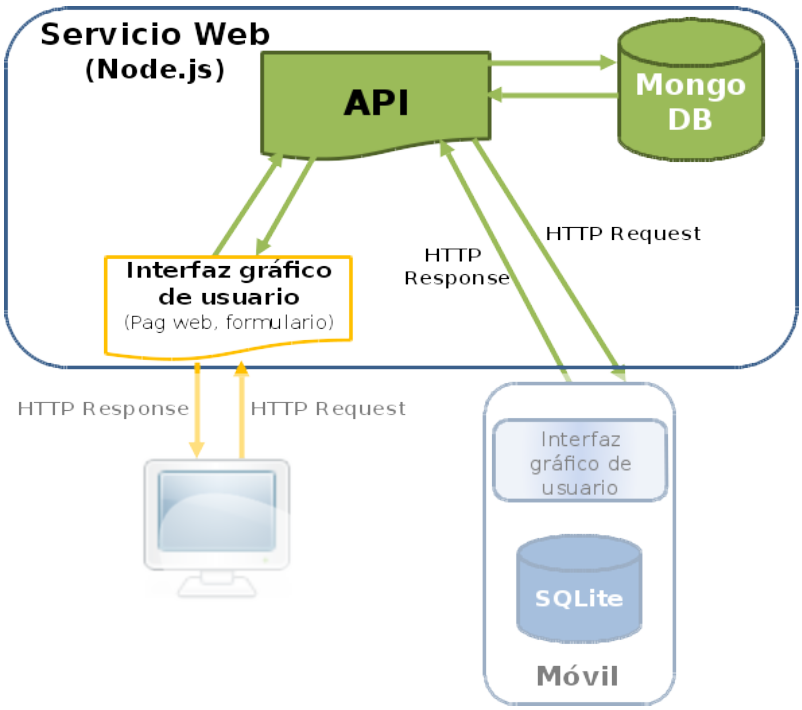


Figura 3.1: Diagrama de la arquitectura relevante al Servicio Web

En este capítulo describiremos los servicios web disponibles en MyConference. En un primer lugar hablaremos del centro neurálgico de MyConference, la API, encargada de la gestión y supervisión de los datos. Posteriormente hablaremos de la base de datos que almacena toda la información de MyConference. Por último, hablaremos del formulario web que permite a los usuarios introducir datos en el sistema.

La figura 3.1 muestra la parte relevante del sistema que se explicará en este capítulo, de acuerdo con la arquitectura general de MyConference.

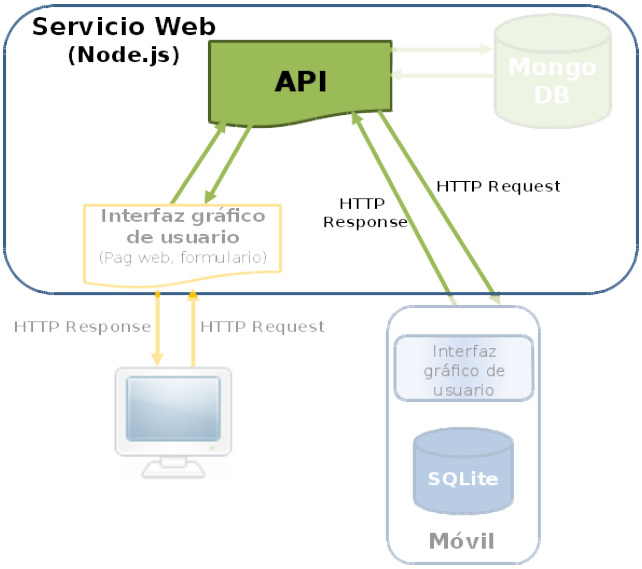


Figura 3.2: Diagrama de la arquitectura relevante a la API

3.1. API

La API (Application Programing Interface) es un servicio HTTP que regula el acceso a todos los datos de MyConference. Es el único punto de toda la infraestructura que tiene acceso a la base de datos y es la única parte que no tiene una utilidad directa para el usuario.

El resto de aplicaciones de MyConference (Web, Android) son interfaces de usuario que realizan *llamadas* (peticiones HTTP) a la API, identificándose a sí mismas como aplicaciones de la plataforma e identificando al usuario a través de sus credenciales (e-mail, contraseña).

La API está implementada en su totalidad utilizando Node.js (Véase 2.2.1) con una serie de bibliotecas que facilitan la implementación en general o en ciertas áreas en particular.

En la figura 3.2 podemos ver cómo está ubicada la API dentro de la arquitectura general del sistema.

3.1.1. Servidor Web

Puesto que se trata de una API completamente web y Node.js es un simple entorno de ejecución JavaScript, el servidor web está programado directamente en la aplicación. Afortunadamente, Node.js cuenta con una biblioteca estándar que incluye soporte para servidores HTTP, y en su repositorio de paquetes hay un gran número de bibliotecas para facilitar el desarrollo.

Para escribir APIs web, concretamente, existe `restify`, un pequeño framework pensado para este tipo de servidores. Mediante esta librería, dividimos toda la implementación en *rutas*, pequeñas definiciones de funciones que responderán a una petición a una ruta HTTP concreta.

A su vez, estas funciones, en su mayoría, realizarán una o varias llamadas a la base de datos. Para trabajar con ella se utilizan una serie de *modelos* implementados mediante `mongoose`, una biblioteca para dar forma a las colecciones de MongoDB como modelos y poder trabajar directamente con estos modelos.

3.1.2. Acceso

El acceso a los datos, así como su modificación, se realiza mediante peticiones HTTP.

Todos los datos se envían y reciben en formato JSON. Cualquier otro formato pedido (mediante la cabecera `Accept`) o enviado (mediante `Content-Type`) será rechazado por el servidor. Esto simplifica el código de la API, puesto que JSON es un formato de representación nativo de JavaScript, el lenguaje en el que está implementada.

Para acceder a los datos es necesario disponer de un testigo de acceso, cuya obtención veremos más adelante en la sección 3.1.3. Para enviar este testigo usaremos la cabecera HTTP `Authorization`, con el valor `Token` *`token-string`* (donde *token-string* es el valor del testigo del que disponemos).

Los datos de la aplicación están divididos en *entidades*, que representan colecciones de datos similares. Para cada entidad disponemos, en general, de las siguientes acciones:

- Ver en detalle un elemento, mediante una petición `GET /entidad/id`. Se obtiene una lista con los atributos del elemento y los elementos relacionados.
- Crear un nuevo elemento de un tipo, mediante una petición `POST /entidad`. Se deben enviar los atributos iniciales del elemento.
- Eliminar un elemento existente, mediante una petición `DELETE /entidad/id`. Se obtiene una confirmación.

Las entidades pueden proporcionar otras acciones en determinadas circunstancias, como se describe en el apartado Servicios Disponibles.

De la misma manera, todas las peticiones utilizan los códigos HTTP estándar, entre ellos:

- `401 Unauthorized` para errores de autenticación.
- `403 Forbidden` para operaciones para las que el usuario no tiene permiso.
- `404 Not Found` para elementos no existentes.
- `500 Internal Server Error` para problemas internos de la API.

3.1.3. Autenticación

Todos los servicios que ofrece la API de MyConference están protegidos. El sistema de autenticación funciona siguiendo un esquema similar al de los estándares OAuth y OpenID. Para acceder a los servicios que ofrece la API de MyConference es necesario obtener un *testigo de acceso*, que irá unido a un usuario y se enviará en cada petición en una cabecera HTTP.

Para obtener un testigo de acceso se debe realizar una petición a una ruta específica (`/auth`) de la API, indicando lo siguiente:

- Qué aplicación se está usando para realizar la petición (Identificada mediante un UUID).
- Qué dispositivo está realizando la petición (Cadena de texto arbitraria, permite diferenciar diferentes sesiones del mismo usuario mediante la misma aplicación).
- Las credenciales de acceso a la aplicación.

Existen múltiples tipos de credenciales de acceso, detallados más adelante. Si todos los datos son correctos, ocurrirá lo siguiente:

- Se crea un nuevo testigo de acceso asociado al usuario cuyas credenciales coinciden.
- Se crea un nuevo testigo de actualización asociado al nuevo testigo de acceso.
- Se invalidan todos los testigos de acceso y de actualización anteriores que fueran generados usando la misma aplicación y el mismo dispositivo.
- Se devuelven ambos testigos, sus fechas de caducidad y el identificador del usuario identificado.

El *testigo de actualización* devuelto tiene como función generar un nuevo testigo de acceso en caso de que este caduque. Los testigos de acceso tienen un tiempo de uso de unos pocos días, mientras que los testigos de actualización duran semanas. Esto minimiza la posibilidad de uso de un testigo de acceso interceptado por una tercera parte.

Para que un intento de autenticación tenga éxito, las credenciales usadas deben coincidir con algunas credenciales almacenadas en la base de datos. Los siguientes son los tipos de credenciales que se definen:

- **Usuario y contraseña**: Utilizamos un nombre de usuario (correo electrónico) y contraseña registrados previamente para realizar la autenticación.
- **Testigo de Actualización**: Se utiliza un testigo de actualización para realizar la autenticación. La aplicación y el dispositivo con los que se generó el testigo usado deben coincidir con los datos de la petición.

- **Testigo de Autorización**: Se utiliza un testigo de autorización obtenido mediante otros medios para realizar la autenticación. La aplicación y el dispositivo con los que se generó el testigo usado deben coincidir con los datos de la petición.

Los testigos de autorización existen para realizar accesos mediante servicios externos, actualmente no implementados.

3.1.4. Servicios Disponibles

A continuación se describen todos los servicios de la API disponibles, con el siguiente formato:

Título del Servicio (`VERBO /ruta/al/servicio`)

En caso de hacer falta algún parámetro en la URL, se mostrará rodeado con corchetes angulares, `<así>`. Si hacen falta parámetros en el cuerpo (en formato JSON), se especificarán en forma de lista:

- `argumento`: Descripción del argumento

Salvo que se indique lo contrario, todas las llamadas requieren del envío de un testigo de acceso válido. Para ello se utilizará la cabecera HTTP `Authorization` con el valor `Token` *`abcdef`*, donde *abcdef* es el testigo de acceso.

Se mostrará a su vez una descripción del servicio, así como de los posibles errores que se pudieran dar en cada uno.

Autenticación y Registro

Registro (`POST /auth/signup`)

- `application_id`: ID de aplicación desde la que se realiza la petición.
- `device_id`: ID del dispositivo desde el que se realiza la petición.
- `user_data`: Un objeto JSON con los campos:
  - `email`: Correo electrónico del usuario a registrar.
  - `password`: Contraseña del usuario a registrar.

La figura 3.3 muestra el proceso de registro sin errores.

El registro no necesita testigo de acceso, puesto que es un proceso previo a la obtención de éste. Los datos de usuario provistos en el cuerpo de la petición se integrarán en el sistema y se creará un nuevo usuario que podrá posteriormente identificarse con ellos.

El correo electrónico ha de ser válido y la contraseña ha de tener al menos 8 caracteres. No existen más restricciones en la contraseña, ni en contenido ni en longitud.

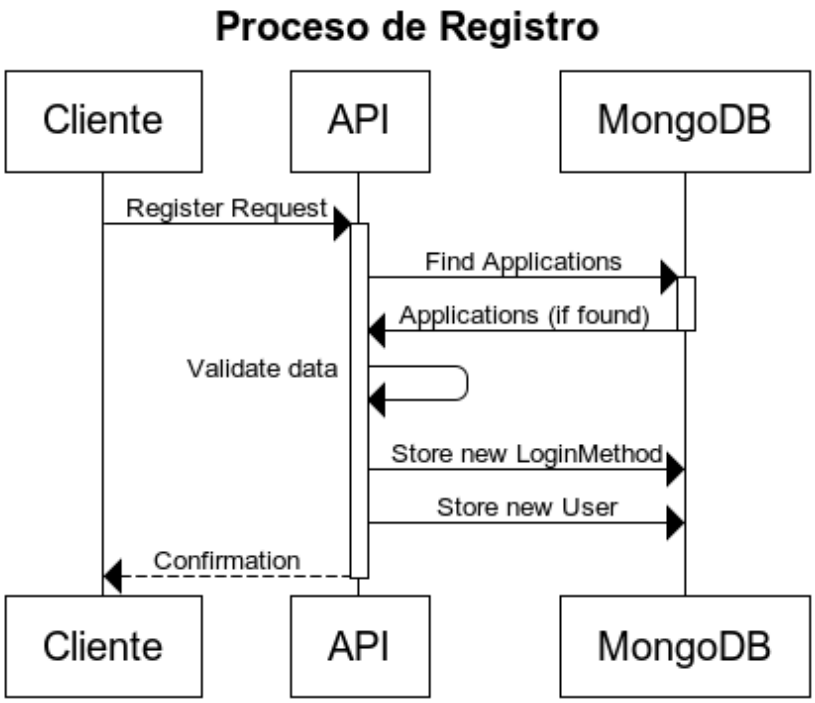


Figura 3.3: Diagrama del Proceso de Registro

Autenticación (`POST /auth`)

- `application_id`: ID de aplicación desde la que se realiza la petición.
- `device_id`: ID del dispositivo desde el que se realiza la petición.
- `credentials`: Un objeto JSON con los campos:
  - `type`: Tipo de credenciales a utilizar. El resto de campos en este objeto dependerán exclusivamente del tipo de credenciales que se estén usando. Este campo admite credenciales de tipo `password` y `refresh`.
  - `email`: (si `type` es `password`) E-mail del usuario a identificar.
  - `password`: (si `type` es `password`) Contraseña del usuario a identificar.
  - `refresh_token`: (si `type` es `refresh`) Testigo de actualización de un usuario ya identificado previamente.

La figura 3.4 muestra el proceso de autenticación sin errores.

Esta llamada no necesita testigo de acceso, puesto que es precisamente con ella como podemos conseguir uno.

El campo `device_id` es libre y puede contener cualquier cosa, pero está presente para posibilitar que un mismo usuario abra múltiples sesiones en la misma aplicación, por ejemplo, en varios PCs o dispositivos Android. Este valor debería ser tan único como fuera posible, por lo que, aunque no es un requisito, se recomienda usar un UUID o

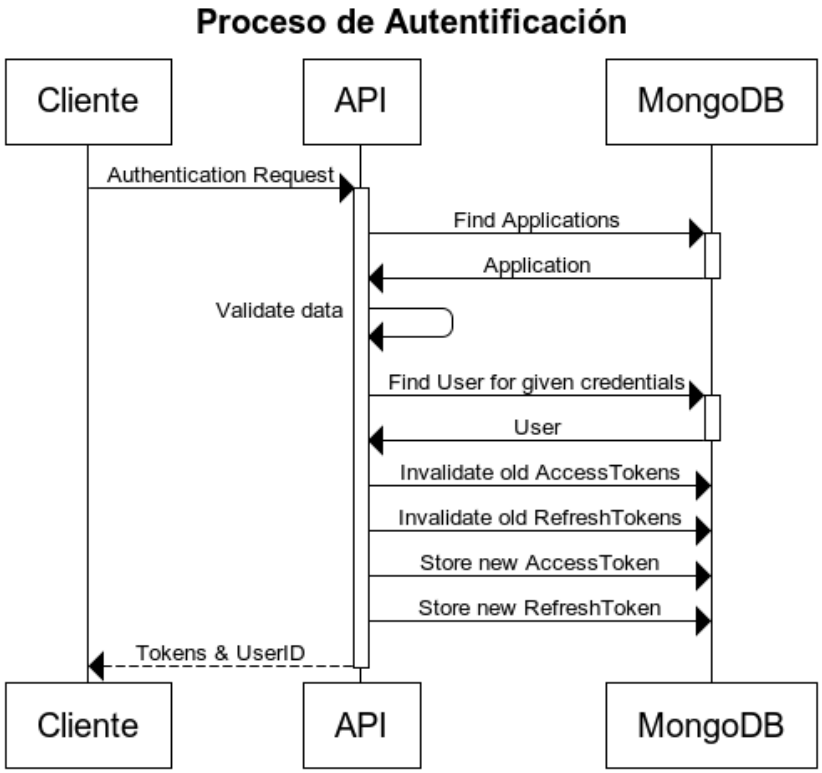


Figura 3.4: Diagrama del Proceso de Autenticación

algún esquema similar. Además, si estamos realizando una llamada con credenciales de tipo `refresh`, este campo ha de coincidir con el de la llamada que generó dicho testigo de actualización originalmente.

El resultado de una llamada válida a este servicio será un objeto JSON con los siguientes campos:

- `access_token`: Nuevo token de acceso para la aplicación.
- `access_expires`: Fecha de caducidad del testigo de acceso (en formato ISO).
- `refresh_token`: Nuevo token de actualización para la aplicación.
- `refresh_expires`: Fecha de caducidad del testigo de actualización (en formato ISO).
- `user`: Información del usuario con el que acabamos de iniciar sesión, como los campos:
  - `id`: Identificador del usuario.
  - `url`: Ruta completa a la que realizar una llamada para obtener toda la información del usuario.

Desautorización (`POST /auth/logout`)

Este servicio no necesita datos extra. Tras realizar esta llamada, el testigo de acceso y su testigo de actualización asociado serán deshabilitados, desautorizando al usuario de todo acceso con dichos testigos.

Este servicio está pensado para utilizarse por las aplicaciones para implementar el "cierre de sesión" característico de cualquier aplicación o servicio web de una forma segura que no engañe al usuario, que realmente desautorice el acceso en lugar de simplemente olvidarlo y dejarlo caducar.

Usuarios y Congresos

Datos de usuario (`GET /users/<user-id>`)

Devuelve la información del usuario indicado en `user-id` como un objeto JSON con los siguientes campos:

- `id`: Identificador del usuario.
- `url`: Ruta completa del usuario en la API.
- `conferences`: Un array JSON con objetos JSON con los siguientes campos.
  - `id`: Identificador del congreso.
  - `url`: Ruta completa del congreso en la API.
  - `role`: Rol del usuario en el congreso (`owner`, `assistant`)

Congresos del usuario (`GET /users/<user-id>/conferences`)

Devuelve la lista de congresos del usuario indicado en `user-id` como una lista de objetos JSON con los siguientes campos:

- `id`: Identificador del congreso.
- `url`: Ruta completa del congreso en la API.
- `name`: Nombre del congreso.
- `description`: Descripción completa del congreso.
- `role`: Rol del usuario en el congreso (`owner`, `assistant`)

Información de Congreso (`GET /conferences/<conference-id>`)

Devuelve la información del congreso indicado en `conference-id` como un objeto JSON con los siguientes campos:

- `id`: Identificador del congreso.
- `url`: Ruta completa del congreso en la API.
- `name`: Nombre del congreso.
- `description`: Descripción completa del congreso.
- `documents`: Array JSON con los documentos (enlaces) del congreso.
- `venues`: Array JSON con los lugares del congreso.
- `announcements`: Array JSON con los anuncios del congreso.
- `organizers`: Array JSON con los organizadores del congreso.
- `speakers`: Array JSON con los ponentes del congreso.
- `agendaEvents`: Array JSON con los eventos del programa del congreso.
- `usuarios`: Array JSON con los usuarios asistentes al congreso.

El contenido exacto de los objetos de cada uno de los arrays se especificará más adelante.

Creación de Congresos (`POST /conferences`)

- `name`: Nombre del congreso.
- `description`: Descripción completa del congreso.

Crea un nuevo congreso y añade al usuario identificado como único propietario del mismo.

Eliminación de Congresos (`DELETE /conferences/<conference-id>`)

Elimina completamente el congreso indicado en `conference-id` y todos los datos relacionados con él, de forma irreversible.

Para que esta llamada tenga efecto, el usuario debe der propietario del congreso.

Entidades del Congreso

Los congresos tienen una serie de entidades directamente dependientes de estos (documentos, ponentes, organizadores, etc.) que se manejan igual unos con respecto a otros. Por este motivo, describiremos primero las entidades y sus campos por separado y, posteriormente, las acciones que podemos realizar sobre ellas.

Todas las entidades tienen los siguientes campos en común:

- `id`: Identificador del objeto.
- `url`: Ruta completa del objeto en la API.
- `conference`: Información del congreso al que pertenece este objeto. Dependiendo de si el objeto forma parte de una lista o se ha obtenido individualmente, habrá más o menos campos, pero podemos contar con los campos `id` y `url` del congreso.

Las entidades disponibles y sus campos son las siguientes (El nombre de la entidad en plural, tal como se escribiría en la ruta del servicio):

`agenda-events` Programa del congreso, en forma de *eventos*.

- `title` Título del evento.
- `description` Descripción del evento.
- `date` Fecha y hora del evento, en formato ISO.

`announcements` Novedades del congreso, siempre ordenadas en orden cronológico inverso (del más reciente al más antiguo).

- `title` Título del anuncio.
- `body` cuerpo del anuncio.
- `date` Fecha y hora del anuncio, en formato ISO.

`documents` Enlaces a documentos del congreso, como carteles, *call for papers*, webs oficiales...

- `title` Título del documento.
- `description` Descripción del documento.
- `type` Tipo de documento. Sólo se admite `link`.
- `data` URL del documento.

`organizers` Miembros del comité organizador del congreso.

- `name` Nombre del organizador.
- `origin` Procedencia del organizador.
- `details` Detalles del organizador.
- `group` Grupo o categoría en la que incluir al organizador.

`speakers` Ponentes del congreso.

- `name` Nombre del ponente.
- `charge` Cargo del ponente.
- `description` Detalles del ponente.
- `picture_url` URL de la foto del ponente.

`venues` Lugares importantes del congreso, como el recinto en el que se celebra.

- `name` Nombre del lugar.
- `details` Detalles del lugar.
- `location` Objeto JSON con dos campos, `lat` y `lng`, indicando la latitud y longitud del lugar.

La forma de listar todos los elementos de una entidad para una conferencia es obtener la información de dicha conferencia. No existen servicios de listado por entidades.

Información de Entidades (`GET /<entity>/<id>`)

Obtiene toda la información del elemento `id` de la entidad `entity`. El resultado es un objeto JSON con los campos descritos para la entidad.

Creación de Entidades (`POST /<entity>`)

Crea un nuevo elemento de la entidad `entity`. Además de los campos descritos para la entidad con la información del objeto, se ha de incluir un campo `conference` con el identificador del congreso al que pertenecerá. El usuario ha de ser propietario de este congreso para poder añadir un elemento en él.

Eliminación de Entidades (`DELETE /<entity>/<id>`)

Elimina el elemento `id` de la entidad `entity`. Para que este servicio funcione correctamente, el usuario debe ser propietario del congreso al que pertenece el objeto.

Códigos de Invitación

Para poder añadir a otros usuarios a un congreso, hay que invitarlos mediante correo electrónico. Para ello se dan los siguientes servicios en la API

Invitar a un Usuario (`POST /invite-codes`)

- `recipient_email`: Correo electrónico al que enviar la invitación. No tiene por qué estar registrado en MyConference ni ser el mismo que el del usuario que lo acepta.
- `recipient_name`: Nombre completo del usuario al que invitar.
- `conference`: Identificador del congreso al que invitar al usuario. El usuario actual ha de ser propietario de este congreso.

Envía un correo electrónico con un enlace a la web y un código para introducir en dicho enlace a la dirección indicada.

Utilizar un Código de Invitación (`POST /invite-codes/<code>/redeem`) Esta llamada no necesita ningún parámetro. Al finalizar esta llamada el usuario se añadirá a la lista de asistentes del congreso al que pertenece el código de invitación.

3.2. Base de Datos

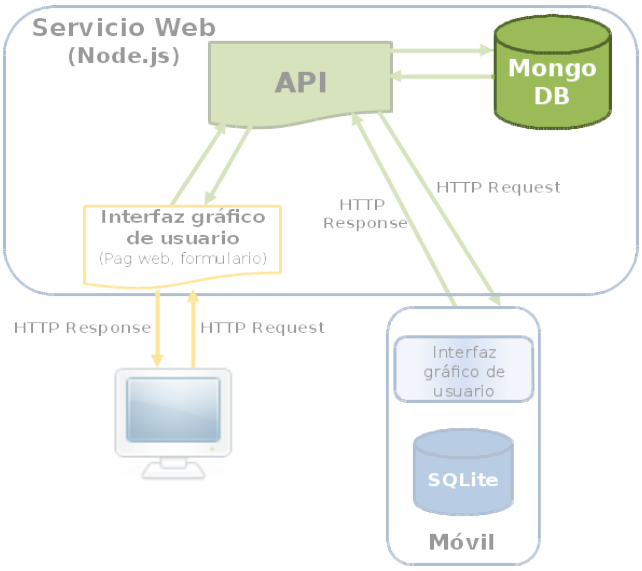


Figura 3.5: Diagrama de la arquitectura relevante a la Base de Datos

La base de datos (MongoDB) está estructurada en colecciones. Cada colección representa una *entidad* de MyConference. Estas entidades se pueden distinguir en *entidades fundamentales*, aquellas que aportan los datos básicos del sistema, y *entidades de apoyo*, aquellas necesarias para el correcto funcionamiento de la aplicación pero que el usuario no percibe.

En la figura 3.5 puede verse la base de datos dentro de la arquitectura general de Myconference. En particular, nótese que la base de datos únicamente interactúa con la API. Ningún otro componente del sistema tiene acceso directo a la base de datos por motivos de seguridad.

3.2.1. Entidades Fundamentales

No se describirán los campos concretos de estas entidades, puesto que son idénticos a los descritos en la sección 3.1.4.

`Conference` El centro de MyConference, representa un congreso.

`User` Representa un usuario registrado en MyConference. Nótese que esta entidad no posee información sobre el inicio de sesión del usuario.

`AgendaEvent` Represetna un evento dentro del programa de un congreso.

Cada evento tiene información acerca de la fecha en la que se celebra y de lo que se hará o de lo que se hablará en él.

`Announcement` Representa un anuncio (novedad) dentro de un congreso.

Cada anuncio contiene información acerca del momento en que se anuncia y qué se está anunciando.

`Document` Representa un enlace a un documento del congreso.

Cada documento contiene un tipo de documento (únacamente pueden ser tipo `link`) y la URL a la que referencian, además de una descripción y un título.

`Organizer` Representa un miembro del comité organizador.

Cada organizador contiene un nombre, una procedencia y un grupo (comité) al que pertenece.

`Speaker` Representa un ponente del congreso.

Cada ponente contiene su nombre, una foto y una descripción.

`Venue` Representa un lugar del congreso.

Cada lugar contiene un nombre, una descripción y una dirección (latitud/longitud).

3.2.2. Entidades de Apoyo

Existen otras entidades en la base de datos que aportan funcionalidad pero no representan datos de negocio. Estas aportan a la aplicación información sobre autenticación, principalmente.

`LoginMethod` Representa un método de identificación de un usuario. El único tipo de identificación disponible es `password`, que utilizará un correo electrónico y una contraseña para la identificación.

`Application` Representa una aplicación oficial de MyConference. El ID de la aplicación se podrá utilizar para registrar nuevos usuarios o autenticar usuarios existentes.

`AccessToken` Representa un testigo de acceso a la API, que deberá usarse para acceder a los datos.

`RefreshToken` Representa un testigo de actualización, que podrá usarse para obtener un nuevo testigo de acceso sin volver a identificar al usuario.

`InviteCode` Representa un código de invitación enviado a un usuario para unirse a un congreso.

3.3. Web

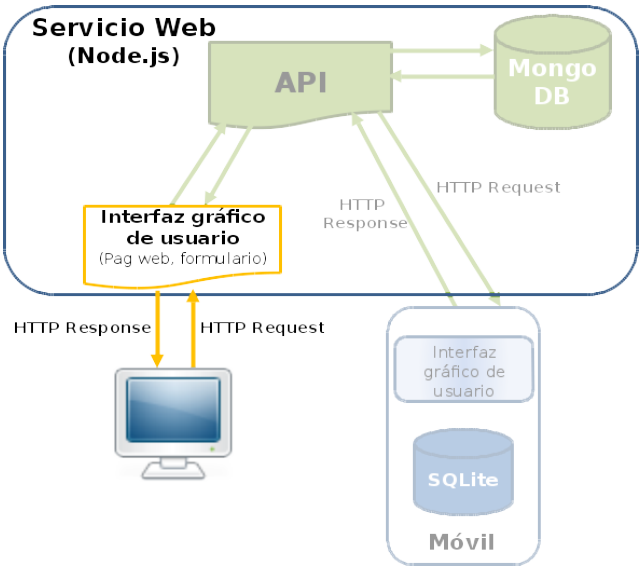


Figura 3.6: Diagrama de la arquitectura relevante a la Web

La Web es la encargada de la gestión de los congresos de MyConference. Desde la aplicación Web se crean los nuevos congresos y se añade toda la información correspondiente a estos. Aunque también se pueden visualizar los datos creados desde la Web, su función principal es la creación y borrado de datos de los congresos.

La figura 3.6 muestra la web dentro de la arquitectura general de MyConference.

La Web está basada en *Express*, framework de desarrollo de aplicaciones web de Node.js.

Como se comentó anteriormente, la función principal de la Web es la creación y el borrado de datos, además de la visualización de estos datos. Para poder realizar estas operaciones es necesario establecer conexiones con la API mediante llamadas HTTP.

La aplicación Web se divide en tres partes: archivos de configuración, rutas e interfaz gráfica.

3.3.1. Archivos de configuración

Los *archivos de configuración* de la aplicación web se encargan de:

- Configurar el módulo *Winston* para usar logs en la aplicación.
- Configurar las sesiones de la aplicación.
- Configurar el módulo *Node SASS* para que compile los archivos de SASS/SCSS a CSS.
- Establecer como archivos estáticos los archivos del directorio `public`.
- Establecer Jade como motor para las vistas.
- Comprobar si los tokens de refresco han caducado.
- Enlazar las rutas de la aplicación, explicadas en la sección 3.3.2
- Configurar el puerto en el que escucha la aplicación.
- Configurar el entorno de desarrollo, que puede ser producción o desarrollo.
- Configurar la base de datos MongoDB en función del entorno de desarrollo.
- Establecer la URL de la API en función del entorno de desarrollo.
- Configurar el módulo *Restify*.

3.3.2. Rutas

el directorio *routes* es la que contiene todos los archivos con las llamadas HTTP a la API. Las llamadas están organizadas en diferentes archivos según los datos que se requieran.

Todos los archivos están enlazados desde los archivos de configuración.

3.3.3. Interfaz Gráfica

Todos los elementos de diseño se guardan en el directorio `public`. En esta carpeta se encuentran los archivos de CSS y JavaScript que requiere la aplicación.

Para la interfaz de usuario se han utilizado las siguientes librerías de apoyo:

Bootstrap (Ver 2.2.4) prácticamente todo el estilo de la página se ha podido hacer de manera fácil y rápida gracias a este framework, obteniendo un resultado sencillo y vistoso añadiendo unas pocas líneas de código.

Google Maps (Ver 2.2.5) usada para el mapa que permite visualizar los lugares añadidos para un congreso.

Fuera del directorio `public` está el directorio `sass`, que contiene todos los archivos `.scss` que usamos en la aplicación. Esta carpeta está fuera del directorio `public` ya que realmente la aplicación no usa estos archivos para el diseño, solo los compila y los traduce a archivos CSS que se guardan dentro del directorio `public`. Estos archivos CSS son los que se usan realmente en la aplicación.

Capítulo 4

Aplicación Móvil

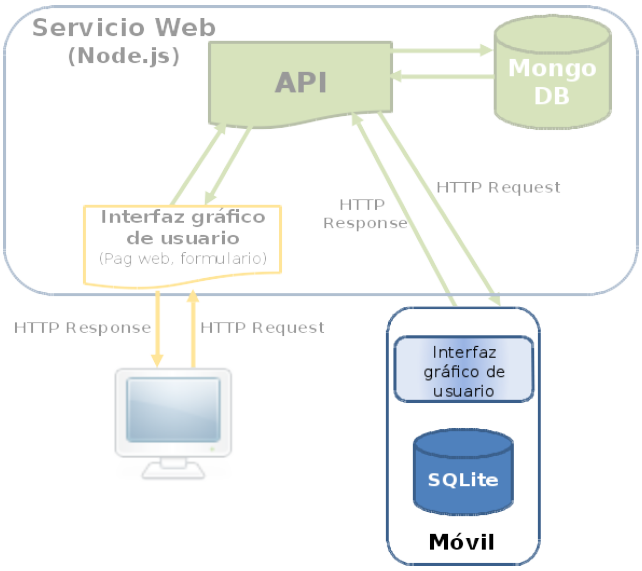


Figura 4.1: Diagrama de la arquitectura relevante a la aplicación android

Para mostrar toda la información sobre los congresos se ha creado una aplicación para dispositivos móviles con Android. En este capítulo se habla sobre él y su entorno, así como de la estructura y de las características de nuestra aplicación para *My Conference*.

La figura 4.1 muestra cómo interactúa la aplicación móvil con el resto del sistema.

4.1. Android

Android [3] es la plataforma móvil más popular del mundo. Se encuentra en cientos de millones de dispositivos en más de 190 países alrededor del mundo. Es la plataforma móvil base más extendida y crece muy rápido – cada día otro millón de usuarios enciende por primera vez su nuevo dispositivo Android y se inicia en el mundo de las apps, los juegos y demás contenido digital.

Además, Android ofrece la capacidad de crear apps y juegos para gente de todas partes del mundo y distribuirlas instantáneamente a través de su tienda abierta a todo el mundo.

Hemos decidido usar Android principalmente por las siguientes razones:

- **Alizanza mundial y una gran base**

  Construído gracias al aporte de la comunidad de código abierto de Linux y con más de 300 asociados hardware, software y móviles, Android se ha convertido en el sistema operativo con un crecimiento mayor.

  Que sea de código abierto lo ha convertido en el favorito tanto para usuarios como para desarrolladores, creciendo rápidamente el consumo de apps. Los usuarios de Android se descargan más de 1.5 billones de apps y juegos de la Google Play cada mes.

  Con la ayuda de sus aliados, Android intenta superar los límites hardware y software para llevar al usuario final y a los desarrolladores una experiencia única llena de posibilidades. Y para los desarrolladores en concreto, la innovación de Android hace que puedan diseñar apps que se diferencien del resto, usando para ello las tecnologías más avanzadas.

- **Framework de desarrollo potente**

  Android ofrece todas las herramientas para crear las mejores apps. A partir de un modelo simple permite desplegar las apps a lo largo y ancho del mundo y para una gran cantidad de dispositivos – desde móviles hasta tablets, y más.

  Android también permite diseñar apps agradables a la vista y que aprovechen todas las capacidades hardware de las que dispone cada dispositivo. Se adapta automáticamente a la interfaz gráfica para que se vea lo mejor posible en cada dispositivo y te da tanto control como sea necesario para configurar cada tipo de dispositivo. Por ejemplo, se puede crear una app que esté optimizada tanto para móviles como para tablets. La interfaz gráfica se crea a partir de simples y ligeros archivos XML, unos para las partes comunes de todos los dispositivos y otros para las partes específicas de móviles o tablets. Y en tiempo de ejecución es Android el encargado de seleccionar los archivos necesarios basándose en el tamaño de la pantalla, la densidad, la localización y más parámetros.

  Para ayudar a los desarrolladores, Android ofrece el paquete llamado *Android Developer Tools*, que es un entorno de desarrollo integrado Java con todo tipo de funcionalidades para crear, debuggear y terminar todo el proceso antes de publicar una app. Usándo este IDE se puede desarrollar para cualquier dispositivo Android existente y utilizar máquinas virtuales para recrear cualquier funcionalidad hardware presente en estos dispositivos.

- **Un mercado libre para distribuir tus apps**

  Google Play es el mercado principal para vender y distribuir apps de Android. Cuando publicas una app, llegas a todos los rincones de Android.

  Como mercado abierto, tu tienes el control de cómo quieres vender tu producto. Se puede publicar cuando se quiera, tantas veces como se quiera, y llegar al

público que se quiera. Se puede publicar para todo el mundo o enfocarse en un sector más pequeño, en una serie de dispositivos o solo interesarse por dispostivos con ciertas capacidades harware.

Se pueden monetizar las apps de la forma que se crea mejor – apps de pago o gratis, con publicidad dentro de la app o con contenido descargable. También se tiene el control del precio que se pone a una app o a los contenidos descargables desde la misma, y se pueden cambiar esos precios siempre que se quiera.

Google Play también te ayuda a tener mayor visibilidad dentro del mercado. A medida que una app gana popularidad, se le reservan sitios estratégicos dentro del mercado y se coloca en el *top ventas* para que sea más accesible al público.

4.2. Desarrollo de la Aplicación Móvil

La app está diseñada para dispositivos con versiones Android 2.3 (Gingerbread) y superiores, hasta la versión actual 4.4 (KitKat). Esto es debido a que la versión 2.3, que salió en el año 2010, a día de hoy representa un 16.2 % de los dispositivos Android totales [6]; no darles soporte para que puedan utilizar la app hubiese sido un error. Esto ha tenido diversas consecuencias que se especificarán durante el desarrollo de los siguientes apartados.

4.2.1. Estructura

La app está dividida en 3 paquetes: uno principal con las clases de cada una de las secciones, otro para gestionar lo referente a la cuenta de usuario y otro misceláneo.

Paquete principal: es.ucm.myconference

Este paquete contiene:

- La que se podría considerar la clase principal - `NavigationDrawerActivity` - se encuentra en este paquete, que se encarga de crear el menú lateral desde el que se accede a cada sección de la app y la que gestiona el cambio entre dichas secciones, pasándoles la información necesaria para su uso. Esta clase gestiona el guardado del estado de la app por posibles eventos externos que pudiesen ocurrir, como un giro en la pantalla, una llamada o el apagado del dispositivo. Este estado se recupera al volver a mostrarse la app. El diagrama UML con los atributos y métodos de esta clase junto con la relación con otras clases se muestra en la figura 4.7
- Observadores para que cuando se produzca algún cambio en la base de datos referente al ingreso o borrado de un congreso, se actualice el menú.
- Una clase (Fragmento) para cada apartado dentro de la app:

- What's new - Muestra las últimas noticias relacionadas con el congreso en formato de lista, con la fecha de publicación, un título y una breve descripción.
- Call for papers - Es un enlace a un archivo pdf con los temas sobre los que deben ir las presentaciones de los conferenciantes.
- Organising Committee - Dividido en una lista por categorías, es un listado de todos los conferenciantes que van a participar, listando su nombre y procedencia.
- Keynote Speakers - Son los ponentes principales y, por tanto, se les lista en un apartado diferente al resto. Incluímos una foto, una descripción y su procedencia y área de estudio.
- Conference Venue - Un listado de las direcciones de los lugares donde se van a celebrar las diferentes presentaciones y un mapa con la ubicación exacta. Si se desea, se puede seleccionar un lugar y buscar en Google Maps el trayecto hasta él.
- Travel Information - Una pequeña introducción a la ciudad y/o el país donde se celebra el congreso.
- Conference Program - El programa del congreso, con una lista con los horarios de cada presentación y el lugar donde se celebra.
- Links - Enlaces de interés que abren el navegador y redireccionan a una página web.
- About - Una descripción del congreso en sí y sobre su propósito.

Todos estos fragmentos se encargan de realizar consultas a la base de datos y mostrar los resultados de una manera ordenada. Para ello utilizan algunas clases que se encuentran en el paquete misceláneo, que se detalla más adelante.

- Las clases para el registro y el logueo del usuario a través de la API. En la respuesta a la llamada se encuentran todos los tokens necesarios para mantener logueado al usuario y poder volver a loguearlo si la sesión ha caducado. Estos tokens junto con información del email y un ID se guardan globalmente para su uso desde otras clases.
- La actividad con la que empieza la aplicación, que es la pantalla de inicio con el nombre del congreso y su logotipo.

Un diagrama de cómo están relacionadas estas clases se puede ver en la figura 4.5.

Paquete cuenta del usuario: es.ucm.myconference.accountmanager

Este paquete contiene la lógica más compleja de la aplicación. En él se encuentran las clases para la gestión de cuentas de usuario y la gestión de la base de datos interna del dispositivo.

Para la gestión de la cuenta del usuario hemos utilizado unas clases que nos da Android para su gestión y almacenamiento [28]. Cuando un usuario se registra, no

solo lo hace a través de nuestra API y queda almacenado en nuestro servidor y base de datos, sino que también queda sincronizada esa cuenta con el teléfono. Desde la pantalla de Ajustes de su dispositivo, el usuario puede ver su cuenta de *MyConference* y activar/desactivar la sincronización para recibir nuevos datos. Para poder realizar esto hemos tenido que implementar varias clases que se encuentran en este paquete:

- `AccountAuthenticator` - El encargado de añadir una nueva cuenta al sistema y guardar el token de autorización (Auth token).
- `AccountAuthenticatorActivity` - La única interacción con el usuario. Presenta una pantalla donde el usuario introduce su email y una contraseña, y se comprueba que efectivamente lo que ha escrito es una dirección de email correcta y que la contraseña tiene mínimo 8 caracteres. Si todo va bien y el usuario se registra correctamente se guarda la cuenta con su token de autorización y se entra en la aplicación.
- `AuthenticatorService` - Servicio a través del cual nos comunicamos con el autenticador.

Para crear y gestionar la base de datos interna del dispositivo, que no tiene nada que ver con la base de datos global de MyConference, a la cual se accede a través de la API, hemos incluído varias clases. Por un lado la clase - `SyncAdapter` - que se encarga de guardar todos los datos que se reciben a través de la API en las distintas tablas de la base de datos interna. De esta forma la app tiene siempre una "copia" de los datos y no tiene que estar accediendo a la base de datos global cada vez que se quiere mostrar algo. Esto consumiría mucha batería y, lo que es peor, la tarifa de datos del usuario. Solo se sincronizará con la API cuando el usuario pulse el botón de Actualizar.

En la figura 4.6 se puede ver un diagrama UML de este paquete.

El resto de clases para gestionar la base de datos interna se explican en la sección 4.3.

Paquete miscelánea: es.ucm.myconference.util

Este paquete contiene:

- Clases Adapter: A la hora de mostrar los datos, las clases predefinidas de Android pueden llegar a ser limitadas en cuanto a la relación de qué datos le envío y cómo queremos que los muestre. A veces es necesario heredar de una clase base e implementar sus métodos a nuestro gusto. En este paquete se encuentran esas clases para varias de las secciones, que se denominan *adapters*.
- Para la gestión de las descripciones de los ponentes principales utilizamos 2 clases, para que el *look&feel* sea mejor. Estas clases se encargan de colocar el texto alrededor de una imagen y que no quede todo descentrado.
- Una clase para almacenar constantes y así tenerlas en un sitio común, y otra clase para datos estáticos, como son los iconos del menú lateral.

4.2.2. Librerías externas

Aparte de las librerías propias de Android, hemos utilizado las siguientes librerías externas:

- *ActionBarSherlock* para poder conseguir el aspecto de una app actual en versiones anteriores a la 3.0 (Honeycomb) [1].
- *Amazing List View* para conseguir un aspecto bonito y sencillo para el programa de las conferencias [2].
- También se ha utilizado *Google Play Services* para poder usar la aplicación de Google Maps dentro de nuestra app.

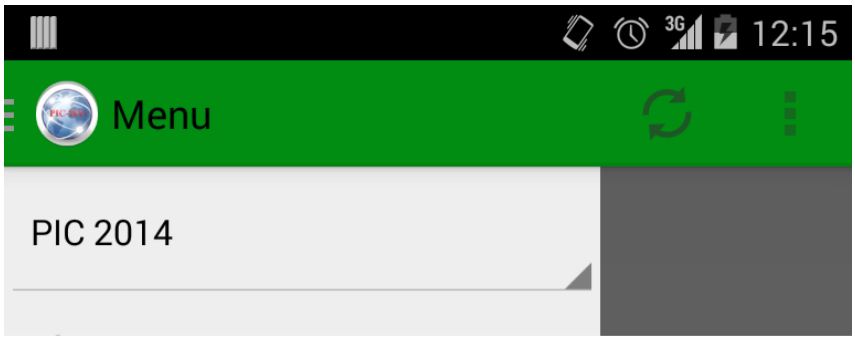


Figura 4.2: *Action Bar* en nuestra app gracias a `ActionBarSherlock`

ActionBarSherlock

Con la llegada de la versión de Android 3.0 (Honeycomb) se introdujo un nuevo elemento a la interfaz de usuario de las apps: la *Action Bar*. Este elemento proporciona al usuario su localización dentro de la aplicación, así como acceso a las acciones de usuario y modos de navegación. Como nuestra app ha sido diseñada para ser compatible con versiones desde la 2.3 ha sido necesario añadir una librería para poder implementarla. Hemos escogido la librería `ActionBarSherlock` por ser una de las más extendidas y, por tanto, mayor documentación y resolución de problemas por parte de la comunidad de desarrolladores. Otra opción hubiese sido desarrollar nuestra app sin este elemento, pero queríamos que se pareciese lo máximo posible al diseño de una app para las últimas versiones, que es lo que el usuario quiere. No una interfaz anticuada y obsoleta.

En la *Action Bar* incluímos el título de la sección actual por la que se navega y un menú para desconectarse de la app. Presionando en el logo de la app se abre el menú lateral con una lista de todas las secciones disponibles. Además, si el menú lateral está abierto, se muestra el botón de actualizar para cargar nuevos datos a la base de datos interna del dispositivo. En la figura 4.2 se puede ver el aspecto final.

Uso Para poder utilizar `ActionBarSherlock` dentro de nuestra app tuvimos que hacer dos cosas:

- Crear una clase abstracta `MyConferenceActivity` que extendiera a la clase `SherlockFragmentActivity`.
- Crear una clase `MyConferenceFragment` que extendiera a `SherlockFragment`.

A partir de ese momento, cualquier actividad que se creara extendería a la clase `MyConferenceActivity` y cualquier fragmento de `MyConferenceFragment`. Esto hacía que automáticamente se introdujese la *Action Bar* en la interfaz gráfica.

Para obtener una referencia a la *Action Bar* solo era necesario llamar al método `getSupportActionBar()`, como si se utilizara la que viene con la librería de soporte, ya que `ActionBarSherlock` sobreescribe ese método y nos devuelve la suya. El resto de código utilizado ha sido habilitar el botón de "home", situado en la esquina izquierda de la barra, utilizando la función `setHomeButtonEnabled(true)`; y realizar una llamada a la función `setDisplayHomeAsUpEnabled(true)` para poder mostrar el icono de abertura y cierre del menú lateral. También hemos utilizado la función `setTitle(String title)` para poder cambiar el título al de la sección actual en la que nos encontráramos.

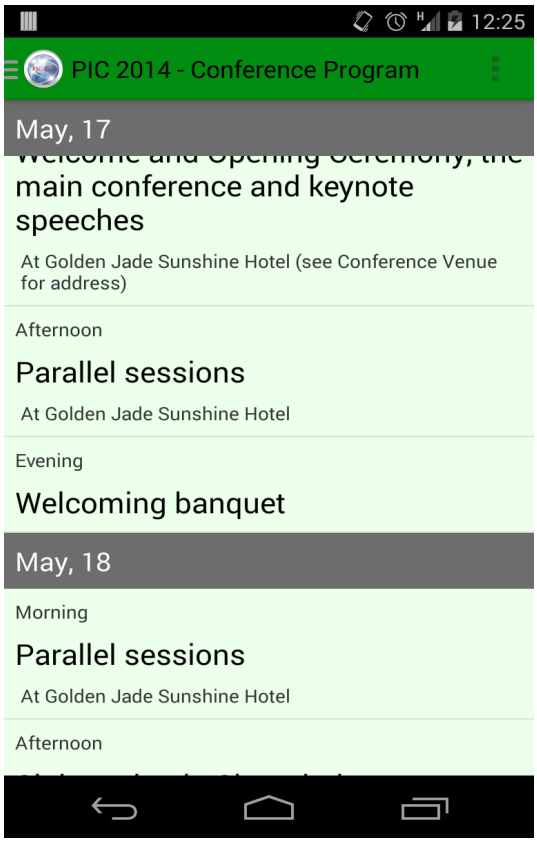


Figura 4.3: Pantalla de la aplicación que muestra la agenda de un congreso

AmazingListView

La forma más clara de mostrar el programa de un congreso es hacerlo con una lista ordenada por días y, dentro de cada día, el horario programado para cada conferencia. Las listas por defecto de Android son bastante simples a la hora de mostrar los datos de manera elegante, así que buscamos una librería externa para darle un aspecto más profesional. Con la librería `AmazingListView` hemos sido capaces de poner la fecha como título para cada una de las conferencias que se desarrollen ese día, y que este título sea visible hasta que aparezca en pantalla el siguiente día. Es una buena forma de no perder de vista la fecha en una lista larga de eventos.

Uso A la hora de incluir esta librería en nuestro proyecto hemos tenido que crear varios archivos que, aunque no ha sido difícil, si ha sido un proceso laborioso, porque hay que ir con cuidado y paso a paso para no equivocarse. Por suerte para nosotros, era una librería que habíamos utilizado en un proyecto anterior como prueba a esta app de `MyConference`, y teníamos una buena base.

- Se crearon 3 archivos XML para la interfaz gráfica. El principal contenía simplemente la definición de un widget de `AmazingListView` con una lista como elemento. Los otros 2 eran el aspecto visual que tendrían las cabeceras de cada sección de la lista y los elementos de la lista en si. En nuestro caso existe una cabecera por cada día de duración del congreso, y cada elemento de la lista contiene una hora de comienzo, un título y una descripción.
- Se creó un fragmento denominado `ProgramFragment` que albergaba la lógica básica de unir cada archivo XML y la consecución de los datos a mostrar.
- Se creó un `AmazingSimpleCursorAdapter` que gestionaba el orden de los elementos, cuáles iban con qué cabeceras y las transiciones entre cabeceras. Esta es la clase más "liosa" porque es la que hay que retocar si algo no se muestra como se esperaba.

En la figura 4.3 se puede ver el aspecto con el que la aplicación presenta la agenda de un congreso. Esta forma de mostrar una lista la hemos podido conseguir utilizando `AmazingListView`.

Google Maps APIv2

Para mostrar la localización de los lugares relevantes del congreso (hoteles, donde se celebran las conferencias...) hemos utilizado Google Maps, incluyendo un pequeño mapa dentro de nuestra app con un marcador en cada sitio.

Al principio añadimos una lista de lugares, cada uno con un enlace que abría la propia aplicación de Google Maps instalada en el dispositivo para mostrar su ubicación, pero decidimos aprovechar que se podía incluir en la propia app para que de manera clara y de un vistazo se pudiesen ver todos los lugares. Actualmente, la aplicación permite las dos cosas. Por un lado el mapa completo con todos los lugares de interés y por otro el listado para que el usuario pueda utilizar las funciones más avanzadas de los mapas, como el creado de una ruta desde su posición hasta la ubicación del congreso. En la figura 4.4 se puede ver como queda esta pantalla.

Figura 4.4: Pantalla de la sección de lugares utilizando *Google Maps*

Uso Para poder utilizar los mapas es necesario incluir la librería *Google Play Services* así como disponer de una API Key única para la app. Es una tarea un poco compleja, pero existe un buen tutorial en la página de desarrolladores de Android [27].

En la clase creada para añadir el mapa hace falta un objeto del tipo `GoogleMap` al que se le asigna un mapa llamando a la función `getMap()`. A partir de aquí queda únicamente la configuración del mapa.

Lo primero que tuvimos que hacer para la configuración del mapa fue añadir un marcador por cada lugar existente en la base de datos. Cuando un usuario pincha sobre el marcador, aparece una ventana de información con el nombre del lugar y si se pincha en esa ventana, lanzamos un `Intent` para abrir la aplicación de Google Maps instalada en el dispositivo con ese lugar pinchado como origen. Si el usuario no tuviese instalada la aplicación simplemente se muestra un mensaje de error en forma de `Toast` comentando que no ha sido posible abrir el mapa.

Y el siguiente paso era centrar el mapa en la ciudad donde se vaya a celebrar el congreso, para así ver todos los marcadores juntos. La API de Google Maps es bastante completa y está bien documentada, así que se pueden conseguir buenos efectos de transición llamando a dos o tres funciones. En nuestro caso creamos un objeto de tipo `CameraPosition` para fijar el lugar donde se tenía que situar la cámara y el zoom que se tenía que aplicar sobre la región, y llamar a la función `animateCamera()` que

se encarga de una transición suave y agradable a la vista.

4.3. Base de datos interna

En la etapa de planificación de la app para Android sabíamos que tenía que haber una base de datos en el dispositivo para almacenar todos los datos que recibiríamos a través de la API. De no ser así, cada vez que el usuario abriese la app tendrían que descargarse todos los datos de nuevo para poder mostrarlos, y este comportamiento tiene varias desventajas:

1. **Rendimiento**. Descargar todos los datos y mostrarlos puede ser un proceso lento, pues depende ya no solo de la cantidad de datos a descargar, sino de la velocidad de conexión a internet del usuario y de la velocidad de su dispositivo.
2. **Batería**. Descargar datos de internet durante un tiempo prolongado o muchas veces en cortos períodos de tiempo suele agotar la batería del dispositivo más rápido, y no es de agrado para un usuario ver cómo se gasta su batería por usar nuestra app. La desinstalaría.
3. **Consumo de datos**. Si no se está conectado por WIFI se utiliza la tarifa de datos que tenga contratada el usuario. Y eso es dinero que no se puede malgastar cada vez que se abre una app.

La forma lógica de atacar el problema era crear una base de datos local y que el usuario decidiera cuándo quería actualizar la información de la que disponía si hubiesen nuevos datos, excepto la primera vez que se abre la app que se tiene que descargar todo. Así, aunque el usuario no disponga conexión a internet, ya tiene en su dispositivo toda la información almacenada para su consulta.

Para la creación y gestión de bases de datos Android incorpora una librería denominada SQLite que utiliza el lenguaje SQL de una forma sencilla y utilizando muy pocos recursos del sistema.

La base de datos se organiza en tablas. Cada tabla se compone de filas y columnas. Las filas se corresponden con los registros y las columnas son los campos.

Siguiendo el ejemplo disponible en la página de desarrolladores de Android hemos creado una base de datos con las siguientes tablas [29]:

- Congresos
- Documentos
- Lugares
- Anuncios
- Ponentes principales
- Comité organizador

- Agenda

Todas las tablas tienen los campos comunes siguientes: la primera columna `ID` de tipo entero y autoincremental. Esto quiere decir que cada vez que se inserta un registro se le asigna un id diferente; y la segunda columna denominada `conf_uuid`, que es el identificador propio de cada congreso, ya que dentro de la app puede haber varios congresos a los que el usuario vaya a asistir y, por ejemplo, todos los anuncios se guardan en la misma tabla, y tiene que poderse diferenciar a qué congreso pertenece cada anuncio. Este identificador se consigue a partir de la tabla de *Congresos*, cuyos campos se explican en la siguiente sección.

El resto de campos son propios de cada tabla:

Congresos Se compone de 3 campos: el nombre del congreso, una descripción que luego se mostrará en la sección *About* de la app, y el campo date con la fecha de creación. Los 3 campos son de tipo `TEXT`.

_id	conf_uuid	title	description	date
INTEGER	TEXT	TEXT	TEXT	TEXT

Documentos Esta tabla está pensada para posibles `.pdf` o `.doc` que se suban de manera informativa. Contiene los campos título, descripción, el tipo que tiene el archivo y datos, que es el archivo en si. Todos los campos son de tipo `TEXT`.

_id	conf_uuid	title	description	type	data
INTEGER	TEXT	TEXT	TEXT	TEXT	TEXT

Lugares Se guarda el nombre y una breve descripción de tipo `TEXT`. La descripción suele ser la dirección completa de la ubicación. También se almacena como dato de tipo `REAL` la latitud y la longitud, que nos sirven para posicionar los marcadores en el mapa de Google Maps.

_id	conf_uuid	name	lat	lng	details
INTEGER	TEXT	TEXT	REAL	REAL	TEXT

Anuncios Los anuncios son pequeños textos dónde se resume algún evento o novedad importante. Para ello se almacena un título, un cuerpo del anuncio y la fecha en que se publica el anuncio. A la hora de mostrarlos se ordenan por fecha del más reciente al más antiguo. Todos los campos son de tipo `TEXT`.

_id	conf_uuid	title	body	date
INTEGER	TEXT	TEXT	TEXT	TEXT

Ponentes principales Como requieren más información que otros ponentes o que los organizadores, se almacenan en una tabla aparte. Sus campos son: el nombre, el cargo que desempeña en su vida profesional, el lugar del que proviene, una descripción de su actividad, una foto y, opcionalmente, uno o varios links a páginas personales. Todos los campos son de tipo `TEXT`.

_id	conf_uuid	name	charge	origin	description	picture_url	links
INTEGER	TEXT	TEXT	TEXT	TEXT	TEXT	TEXT	TEXT

Comité organizador Esta tabla es la más poblada ya que hay mucha gente detrás de la organización de un congreso. Se almacena su nombre, la ciudad o país de donde proviene, una breve descripción si la hay y el grupo al que pertenece dentro del comité, que se utiliza como cabecera a la hora de mostrarlos dentro de la app.

_id	conf_uuid	name	origin	details	groups
INTEGER	TEXT	TEXT	TEXT	TEXT	TEXT

Agenda En ella se almacena el programa del congreso, donde cada conferencia tiene un título, una descripción y su fecha (día y hora).

_id	conf_uuid	title	description	date
INTEGER	TEXT	TEXT	TEXT	TEXT

Para gestionar la base de datos existen dos clases dentro del paquete `es.ucm.myconference.accountmanager`:

- `SqlHelper`, que extiende a SQLiteOpenHelper. Es la clase que se encarga de crear las tablas de la base de datos cuando se instala la app en el dispositivo. Al crear la base de datos se le da un nombre y una versión, que se va incrementando a medida que se van añadiendo nuevas tablas.

  A través de comandos SQL se crean los diferentes campos de las tablas y se les asigna un tipo: `TEXT` para texto y `REAL` o `INT` para números. No nos ha hecho falta ningún tipo más pues la inmensa mayoría es texto. Esto hace que la base de datos no sea demasiado pesada.

  Asimismo, esta clase se encarga de actualizar la base de datos cuando una nueva versión está disponible.

- `ConfsProvider`, que extiende a ContentProvider. Cuando se accede a la base de datos para una búsqueda o una inserción nueva, ésta es la clase que lo gestiona. `ContentProvider` se traduce como Proveedor de Contenidos.

  Para poder acceder al contenido de la base de datos es necesario que esté identificado, y eso se hace a través de las URIs. Una URI es una cadena de texto que permite identificar un recurso de información. Lo primero que se hace es asignar dos URIs a cada una de las tablas de la base de datos: una para

realizar consultas de un solo registro y otra para varios. También se asigna un tipo para cada URI, dependiendo si es individual o colectiva. Y a continuación están los métodos que se encargan de las consultas, inserciones, borrados y actualizaciones de registros.

Los `ContentProvider` utilizan una clase denominada `Cursor` para devolver los resultados de las consultas. Son colecciones que se recorren de la misma manera que el patrón *iterator*, y que utilizan los adapters para mostrar los datos. Las inserciones devuelven la URI del nuevo registro almacenado, y los borrados y actualizaciones devuelven el número de registros afectados.

4.4. SharedPreferences

Android ofrece otra forma de guardar datos de manera persistente cuando instalamos una app, y son las SharedPreferences.

Esta información se almacena de manera privada, de modo que solo nuestra app puede acceder a ella. Son pares clave-valor cuyos tipos sólo pueden ser booleans, floats, ints, longs, y strings.

En nuestro caso utilizamos este tipo de almacenamiento para guardar el nombre del usuario, el email, su identificador, y los tokens de acceso y refresco que nos proporciona la API al registrarse o hacer login en la app. También se utiliza durante la aplicación para almacenar si el usuario se desloguea de la app antes de salir o para saber si es la primera vez que accede a la app.

Si el usuario sale de la app sin desloguearse, al volver a entrar aparecerá en la pantalla inicial. Si por el contrario pulsa en Logout para salir, al entrar se le solicitarán los credenciales.

Almacenar un booleano indicando si es la primera vez que entra en la app lo utilizamos para mostrar el menú lateral durante un período de tiempo, según la guía de diseño de Android. [13]

4.5. Permisos

Al instalar una app Android en un dispositivo se muestran los distintos permisos que necesita el usuario aceptar para que la aplicación pueda funcionar.

Para nuestra app hacen falta los siguientes permisos:

- INTERNET. Es necesario poder conectarse a internet para la descarga de los datos desde nuestra base de datos al dispositivo y para descargar los mapas de Google Maps.
- READ\WRITE_SYNC_SETTINGS. Permite leer y controlar las opciones de sincronización. Se utiliza, por ejemplo, para sincronizaciones periódicas o para ver si en un momento determinado se puede sincronizar con el servidor.

- AUTHENTICATE_ACCOUNTS. Se utiliza para poder autenticar la cuenta de usuario que creamos en el dispositivo.
- GET_ACCOUNTS. Al iniciar una sincronización es necesario localizar la cuenta de usuario desde la que se manda la petición.
- ACCESS_NETWORK_STATE. Permite a la API de Google Maps saber el estado de la conexión para ver si es posible descargar la información.
- READ_GSERVICES. Da acceso a los servicios web de Google.
- ACCESS_COARSE_LOCATION. Se utiliza para permitir que se localice la posición del dispositivo, ya sea por WIFI o por la red de datos.
- ACCESS_FINE_LOCATION. Para espacios más reducidos, poder utilizar el GPS para posicionar el dispositivo.
- WRITE_EXTERNAL_STORAGE. Utilizado por los mapas de Google Maps para cachear los mapas en la memoria externa del dispositivo.

También es necesario incluir que se utiliza OpenGL para los mapas en su versión 2. Sin esto en el dispositivo no se visualizarán los mapas.

Todos estos permisos se incluyen en `AndroidManifest.xml`, archivo obligatorio en toda app de Android que contiene información de la misma necesaria para el sistema.

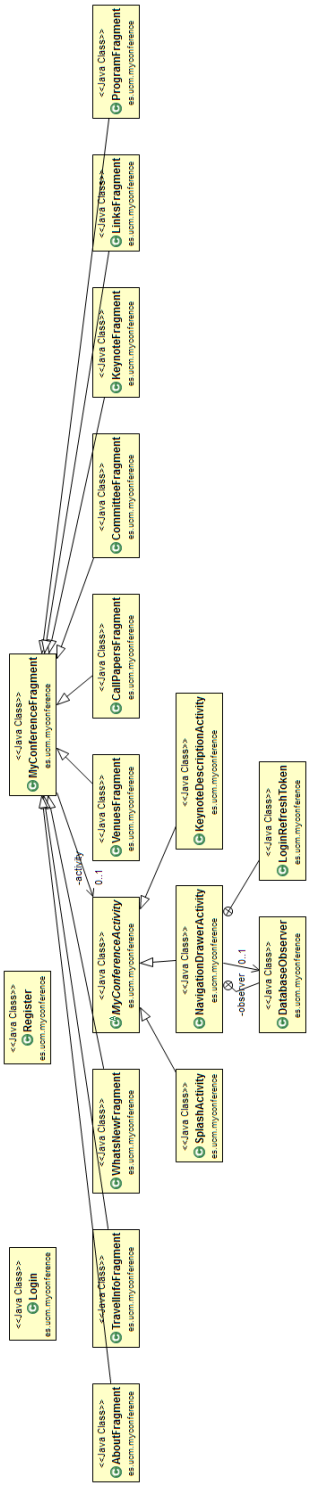


Figura 4.5: Relación entre las clases del paquete es.ucm.myconference.

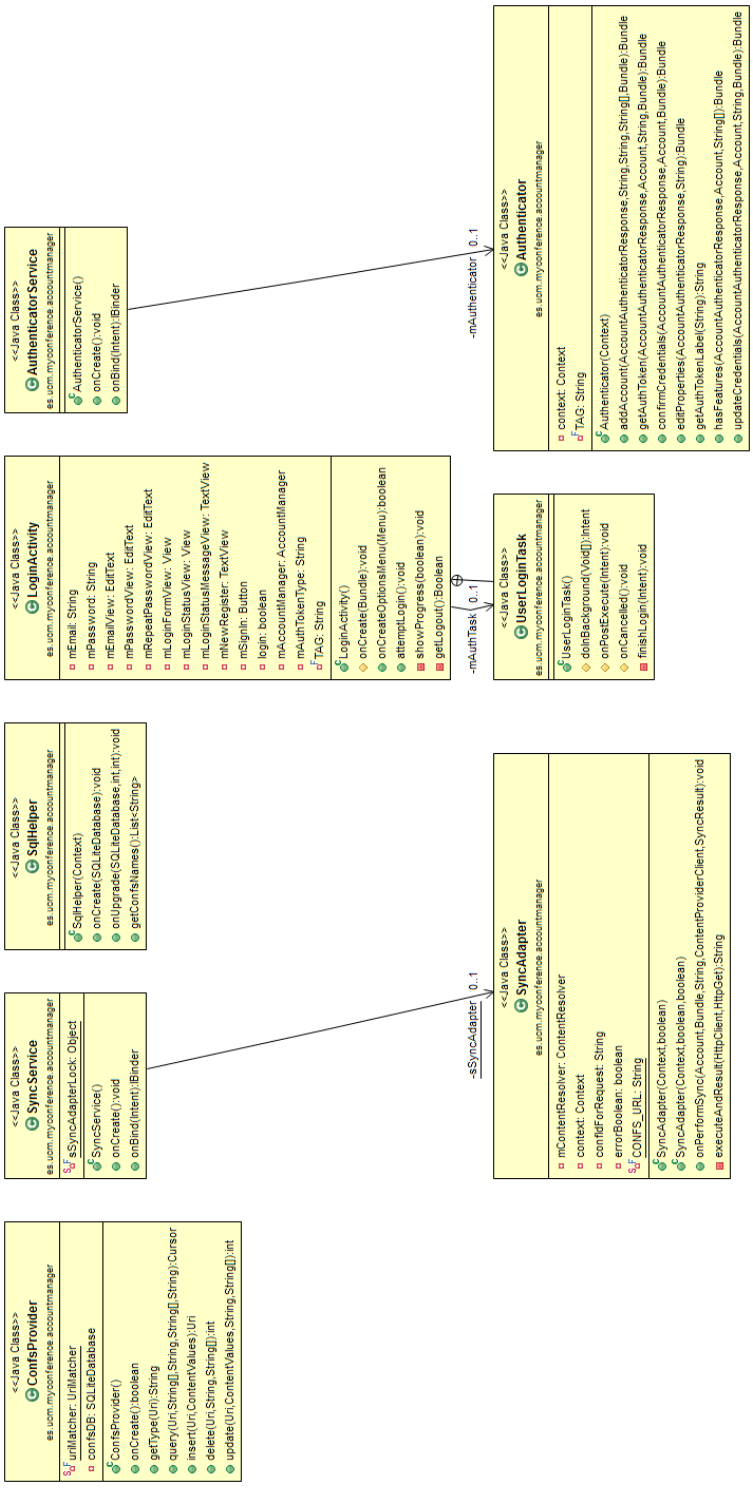


Figura 4.6: Diagrama UML de las clases del paquete `es.ucm.myconference.accountmanager`.

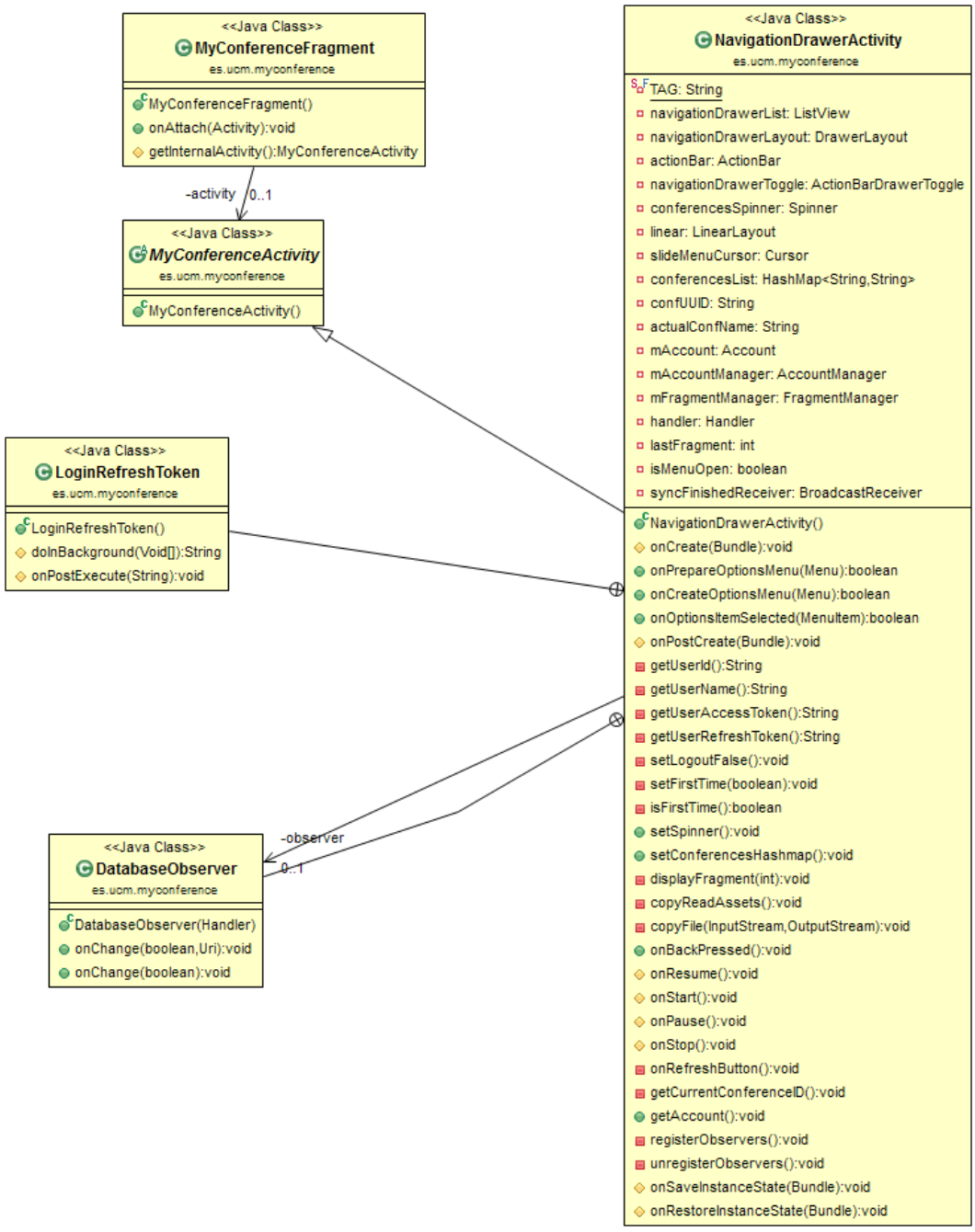


Figura 4.7: Diagrama UML de la clase `NavigationDrawerActivity`.

Capítulo 5

Casos de Uso

5.1. Casos de Uso del Formulario Web

Caso de uso: registro de usuario

1. **Objetivo**

   Registrar al usuario en la aplicación.

2. **Actores**

   Usuario sin registrar.

3. **Precondición**

   El usuario ha accedido a la web y está en página de registro.

4. **Postcondición si éxito**

   El usuario queda registrado.

5. **Postcondición si fallo**

   Se muestra un mensaje de error indicando el fallo que ha ocurrido.

6. **Entradas**

   Formulario con los siguientes campos obligatorios:

   - E-mail.
   - Contraseña.
   - Repetir contraseña.

7. **Salidas**

   Acceso a la aplicación o mensaje de error en caso de fallo.

8. **Secuencia normal**

   a) Comprobar que no haya ningún campo vacío. Si hay algún campo vacío, S-1.

b) Comprobar que sea un e-mail válido. Si no es un e-mail válido, S-2.

c) Comprobar que la contraseña tenga 8 caracteres mínimo. Si la contraseña es demasiado corta, S-3.

d) Comprobar que los campos contraseña y repetir contraseña sean iguales. Si no coinciden, S-4.

e) Comprobar que el e-mail no ha sido registrado previamente en la aplicación. Si ha sido registrado previamente, S-5.

f) El usuario es registrado en la aplicación y se inicia sesión automáticamente.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error indicando los campos faltantes.
   - S-2. Se muestra un mensaje de error indicando que el e-mail no tiene un formato válido.
   - S-3. Se muestra un mensaje de error indicando que la contraseña es demasiado corta.
   - S-4. Se muestra un mensaje de error indicando que los campos contraseña y repetir contraseña tienen que coincidir.
   - S-5. Se muestra un mensaje de error indicando que el e-mail ya está registrado.

Caso de uso: inicio de sesión

1. **Objetivo**

   Identificar al usuario mediante sus credenciales para el acceso a la aplicación.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha accedido a la web y se ha registrado previamente.

4. **Postcondición si éxito**

   Se identifica al usuario y accede a sus datos en la aplicación.

5. **Postcondición si fallo**

   Se muestra mensaje de error indicando el fallo que ha ocurrido.

6. **Entradas**

   Formulario con los siguientes campos obligatorios:

   - E-mail.
   - Contraseña.

7. **Salidas**

   Acceso a la sesión del usuario o mensaje de error en caso de fallo.

8. **Secuencia normal**

   a) Comprobar que no haya ningún campo vacío. Si hay algún campo vacío, S-1.

   b) Comprobar que sea un e-mail válido. Si no es un e-mail válido, S-2.

   c) Comprobar que la contraseña tenga 8 caracteres mínimo. Si la contraseña es demasiado corta, S-3.

   d) Comprobar que el e-mail está registrado en la aplicación. Si no ha sido registrado previamente, S-4.

   e) El usuario inicia sesión.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error indicando los campos faltantes.
   - S-2. Se muestra un mensaje de error indicando que el e-mail no tiene un formato válido.
   - S-3. Se muestra un mensaje de error indicando que la contraseña es demasiado corta.
   - S-4. Se muestra un mensaje de error indicando que el e-mail no está registrado en la aplicación.

Caso de uso: mostrar los datos del usuario

1. **Objetivo**

   Mostrar al usuario sus datos mediante llamadas a la API. Estos datos pueden ser:

   - Lista de congresos, tanto los creados por el propio usuario como a los que ha sido invitado a asistir.
   - Lista de los usuarios invitados al congreso.
   - Lista de los anuncios del congreso.
   - Lista de los documentos del congreso.
   - Lista de lugares del congreso.
   - Lista de los organizadores del congreso.
   - Lista de los ponentes del congreso.
   - Lista de los eventos de la agenda del congreso.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha iniciado sesión con su cuenta y se han recibido correctamente los datos de la API.

4. **Postcondición si éxito**

   Se muestra una lista con los datos según la sección en la que esté el usuario.

5. **Postcondición si fallo**

   Se muestra mensaje de error indicando que no se han recibido correctamente los datos.

6. **Entradas**

   Ninguna.

7. **Salidas**

   Lista con los datos correspondientes.

8. **Secuencia normal**

   a) Se hace una llamada a la API pidiendo los datos correspondientes. Si no se puede realizar la llamada error, S-1.

   b) Se muestra la lista con los datos.

9. **Secuencia alternativa**

   - S-1. Se muestra mensaje de error indicando que no se han recibido correctamente los datos.

Caso de uso: creación de congresos

1. **Objetivo**

   Crear un nuevo congreso.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha accedido a la web y ha iniciado sesión.

4. **Postcondición si éxito**

   Se crea un nuevo congreso.

5. **Postcondición si fallo**

   Se muestra un mensaje de error indicando que no se ha podido crear el congreso.

6. **Entradas**

   Formulario con los siguientes campos:

   - Nombre del congreso.
   - Descripción del congreso.

7. **Salidas**

   El nuevo congreso.

8. **Secuencia normal**

   a) Rellenar los campos del formulario.

   b) Comprobar que los campos tienen el formato correcto. Si algún campo no tiene el formato correcto, S-1.

   c) Se crea el congreso.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error informando que algún campo tiene un formato incorrecto.

Caso de uso: invitación de usuarios

1. **Objetivo**

   Invitar a usuarios al congreso para que puedan visualizar sus datos.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha accedido a la web y ha iniciado sesión.

4. **Postcondición si éxito**

   Se manda al otro usuario un correo con la invitación.

5. **Postcondición si fallo**

   Se muestra un mensaje de error indicando que no se ha podido mandar la invitación.

6. **Entradas**

   Formulario con los siguientes campos:

   - Nombre del usuario al que se quiere invitar.
   - E-mail del usuario al que se quiere invitar.

7. **Salidas**

   Correo con un código de activación.

8. **Secuencia normal**

   a) Rellenar los campos del formulario.

   b) Comprobar que los campos tienen el formato correcto. Si algún campo no tiene el formato correcto, S-1.

   c) Se manda la invitación.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error informando que algún campo tiene un formato incorrecto.

Caso de uso: activación de congresos

1. **Objetivo**

   Activar el congreso al que el usuario ha sido invitado.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha accedido a la web y ha iniciado sesión.

4. **Postcondición si éxito**

   Se añade la información del congreso a la cuenta del usuario.

5. **Postcondición si fallo**

   Se muestra un mensaje de error indicando que no se ha podido activar el congreso.

6. **Entradas**

   Código de activación.

7. **Salidas**

   Datos del congreso.

8. **Secuencia normal**

   a) Abrir el enlace que se proporciona en el correo.
   b) Introducir el código de activación en el enlace abierto previamente.
   c) Se valida el código de activación. Si falla, S-1.
   d) Se añade la información del congreso en la cuenta del usuario.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error informando que el código de activación no es válido.

Caso de uso: creación de anuncios

1. **Objetivo**

   Crear un nuevo anuncio.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha accedido a la web y ha iniciado sesión.

4. **Postcondición si éxito**

   Se crea un nuevo anuncio.

5. **Postcondición si fallo**

   Se muestra un mensaje de error indicando que no se ha podido crear el anuncio.

6. **Entradas**

   Formulario con los siguientes campos:

   - Título del anuncio.
   - Cuerpo del anuncio.

7. **Salidas**

   El nuevo anuncio.

8. **Secuencia normal**

   a) Rellenar los campos del formulario.

   b) Comprobar que los campos tienen el formato correcto. Si algún campo no tiene el formato correcto, S-1.

   c) Se crea el anuncio.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error informando que algún campo tiene un formato incorrecto.

Caso de uso: creación de lugares

1. **Objetivo**

   Crear un nuevo lugar.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha accedido a la web y ha iniciado sesión.

4. **Postcondición si éxito**

   Se crea un nuevo lugar.

5. **Postcondición si fallo**

   Se muestra un mensaje de error indicando que no se ha podido crear el lugar.

6. **Entradas**

   Formulario con los siguientes campos:

   - Nombre del lugar.
   - Detalles del lugar.
   - Coordenadas geográficas o dirección del lugar.

7. **Salidas**

   El nuevo lugar.

8. **Secuencia normal**

   a) Rellenar los campos del formulario.

   b) Comprobar que los campos tienen el formato correcto. Si algún campo no tiene el formato correcto, S-1.

   c) Se crea el lugar.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error informando que algún campo tiene un formato incorrecto.

Caso de uso: creación de organizadores

1. **Objetivo**

   Crear un nuevo organizador.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha accedido a la web y ha iniciado sesión.

4. **Postcondición si éxito**

   Se crea un nuevo organizador.

5. **Postcondición si fallo**

   Se muestra un mensaje de error indicando que no se ha podido crear el organizador.

6. **Entradas**

   Formulario con los siguientes campos:

   - Nombre del organizador.
   - Procedencia del organizador.
   - Detalles del organizador.
   - Grupo del organizador.

7. **Salidas**

   El nuevo organizador.

8. **Secuencia normal**

   a) Rellenar los campos del formulario.

   b) Comprobar que los campos tienen el formato correcto. Si algún campo no tiene el formato correcto, S-1.

   c) Se crea el organizador.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error informando que algún campo tiene un formato incorrecto.

Caso de uso: creación de ponentes

1. **Objetivo**

   Crear un nuevo ponente.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha accedido a la web y ha iniciado sesión.

4. **Postcondición si éxito**

   Se crea un nuevo ponente.

5. **Postcondición si fallo**

   Se muestra un mensaje de error indicando que no se ha podido crear el ponente.

6. **Entradas**

   Formulario con los siguientes campos:

   - Nombre del ponente.
   - Procedencia del ponente.
   - Cargo del ponente.
   - Descripción del ponente.

7. **Salidas**

   El nuevo ponente.

8. **Secuencia normal**

   a) Rellenar los campos del formulario.

   b) Comprobar que los campos tienen el formato correcto. Si algún campo no tiene el formato correcto, S-1.

   c) Se crea el ponente.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error informando que algún campo tiene un formato incorrecto.

Caso de uso: creación de eventos de la agenda

1. **Objetivo**

   Crear un nuevo evento en la agenda.

2. **Actores**

   Usuario registrado en la aplicación.

3. **Precondición**

   El usuario ha accedido a la web y ha iniciado sesión.

4. **Postcondición si éxito**

   Se crea un nuevo evento en la agenda.

5. **Postcondición si fallo**

   Se muestra un mensaje de error indicando que no se ha podido crear el evento.

6. **Entradas**

   Formulario con los siguientes campos:

   - Título del evento.
   - Descripción del evento.
   - Día, mes y año del evento.
   - Hora y minutos del evento.

7. **Salidas**

   El nuevo evento.

8. **Secuencia normal**

   a) Rellenar los campos del formulario.

   b) Comprobar que los campos tienen el formato correcto. Si algún campo no tiene el formato correcto, S-1.

   c) Se crea el evento de la agenda.

9. **Secuencia alternativa**

   - S-1. Se muestra un mensaje de error informando que algún campo tiene un formato incorrecto.

Caso de uso: borrado de datos

1. **Objetivo**

   El usuario elimina datos de su cuenta. Los datos que puede borrar son:

   - Congresos.
   - Anuncios.
   - Documentos.
   - Lugares.
   - Organizadores.
   - Ponentes.
   - Eventos de la agenda.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   El usuario es el propietario de los datos que se quieren borrar.

4. **Postcondición si éxito**

   Se eliminan los datos.

5. **Postcondición si fallo**

   Se muestra mensaje de error informando del fallo.

6. **Entradas**

   Ninguna.

7. **Salidas**

   Se elimina el dato.

8. **Secuencia normal**

   a) El usuario pulsa sobre el botón **Delete** en el dato que quiere eliminar. Si no se ha podido eliminar, S-1.

   b) Se elimina el dato.

9. **Secuencia alternativa**

   a) S-1. se muestra mensaje de error informando del fallo.

Caso de uso: cerrar sesión

1. **Objetivo**

   Cerrar la sesión actual del usuario y volver a la página inicial de la web.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   El usuario ha iniciado sesión en la aplicación.

4. **Postcondición si éxito**

   Se cierra la sesión actual del usuario.

5. **Postcondición si fallo**

   Ninguna.

6. **Entradas**

   Ninguna.

7. **Salidas**

   Cierre de la sesión actual del usuario.

8. **Secuencia normal**

   a) El usuario pulsa sobre el botón **Log out** en la barra de navegación.

   b) Se cierra la sesión.

9. **Secuencia alternativa**

   Ninguna.

5.2. Casos de Uso de la Aplicación Móvil

Caso de uso: registrar usuario

1. **Objetivo**

   Coger los datos de email y contraseña de un formulario e introducirlos en la base de datos.

2. **Actores**

   Usuario sin registrar.

3. **Precondición**

   El usuario se ha descargado la app a su dispositivo.

4. **Postcondición si éxito**

   El usuario queda registrado en la base de datos.

5. **Postcondición si fallo**

   Se muestra un mensaje de error con los campos que no cumplen los requisitos para que el usuario los modifique. Si el error es interno se pide al usuario que vuelva a introducir los credenciales.

6. **Entradas**

   Los datos del formulario son: email, contraseña y repetir contraseña. Todos los campos son obligatorios.

7. **Salidas**

   Entrada en la app o error.

8. **Secuencia normal**

   a) Validar los campos según los requisitos: debe ser un email válido y la contraseña debe tener un mínimo de 8 caracteres.
   b) Si hay campos sin rellenar, S-1.
   c) Si la contraseña es demasiado corta, S-2.
   d) Si los campos de contraseña y repetir contraseña no coinciden, S-3.
   e) Si el email ya existe en la base de datos, S-4.
   f) Se loguea al usuario y se entra en la app.

9. **Secuencia alternativa**

   S-1. Se muestra un mensaje de error para que el usuario introduzca los datos faltantes.

   S-2. Se muestra un mensaje de error para que el usuario introduzca una constraseña más larga.

   S-3. Se muestra mensaje de error de que no coinciden y se borran los campos para que se vuelvan a introducir.

   S-4. Se indica al usuario que el email ya existe y que introduzca otro.

Caso de uso: loguear usuario

1. **Objetivo**

   Identificar al usuario en la app a partir de un email y una contraseña.

2. **Actores**

   Usuario registrado en la app.

3. **Precondición**

   El usuario se ha descargado la app y se ha registrado a través de la misma o desde la página web.

4. **Postcondición si éxito**

   Se identifica al usuario y entra en la app.

5. **Postcondición si fallo**

   Se muestra un mensaje de error para que el usuario vuelva a introducir sus credenciales.

6. **Entradas**

   Los datos del formulario son: email y contraseña.

7. **Salidas**

   Entrada en la app o error.

8. **Secuencia normal**

   a) Validar los campos según los requisitos: debe ser un email válido y la contraseña debe tener un mínimo de 8 caracteres.

   b) Si hay campos sin rellenar, S-1.

   c) Si la contraseña es demasiado corta, S-2.

   d) Si el email ya existe en la base de datos, S-3.

   e) Se loguea al usuario y se entra en la app.

9. **Secuencia alternativa**

   S-1. Se muestra un mensaje de error para que el usuario introduzca los datos faltantes.

   S-2. Se muestra un mensaje de error para que el usuario introduzca una constraseña más larga.

   S-3. Se indica al usuario que el email ya existe y que introduzca otro.

Caso de uso: mostrar novedades

1. **Objetivo**

   Mostrar al usuario una lista con las últimas novedades sobre el congreso.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Se han recibido bien los datos desde la API y se han guardado en la base de datos interna.

4. **Postcondición si éxito**

   Se muestra una lista con las novedades, ordenadas por fecha de la más reciente a la más antigua.

5. **Postcondición si fallo**

   Se muestra un mensaje informando de que no existen novedades.

6. **Entradas**

   Ninguna.

7. **Salidas**

   Lista de novedades.

8. **Secuencia normal**

   a) Se hace una llamada a la base de datos. Si error, S-1.

   b) Se muestra la lista.

9. **Secuencia alternativa**

   S-1. No se muestra mensaje de error explícito y no se muestra la lista.

Caso de uso: mostrar el documento Call for Papers

1. **Objetivo**

   Abrir un pdf con la información de los temas a tratar en el congreso o mostrar una lista.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Ninguna.

4. **Postcondición si éxito**

   Se abre el visor de documentos pdf del dispositivo y muestra la información, o se muestra una lista con esta información.

5. **Postcondición si fallo**

   Mensaje de error al no disponer el usuario de una aplicación para visualizar documentos pdf.

6. **Entradas**

   Ninguna.

7. **Salidas**

   Información de los Call for Papers.

8. **Secuencia normal**

   a) Si está en un archivo pdf, se abre la aplicación para visualizarlo. Si error, S-1.

   b) Si no, se hace una llamada a la base de datos. Si error, S-2.

   c) Se muestra la lista.

9. **Secuencia alternativa**

   S-1. Mensaje de error al no disponer de una aplicación para pdfs.

   S-2. No se muestra mensaje de error explícito y no se muestra la lista.

Caso de uso: mostrar la lista del comité organizador

1. **Objetivo**

   Mostrar una lista agrupada con los miembros del comité organizador.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Se han recibido bien los datos desde la API y se han guardado en la base de datos interna.

4. **Postcondición si éxito**

   Se muestra una lista con los miembros del comité, agrupada según el grupo al que pertenezca cada miembro. Se incluye su nombre y su procedencia.

5. **Postcondición si fallo**

   Se muestra un mensaje informando de que no existen miembros.

6. **Entradas**

   Ninguna.

7. **Salidas**

   Lista de miembros.

8. **Secuencia normal**

   a) Se hace una llamada a la base de datos. Si error, S-1.

   b) Se muestra la lista.

9. **Secuencia alternativa**

   S-1. No se muestra mensaje de error explícito y no se muestra la lista.

Caso de uso: mostrar la lista de los ponentes principales

1. **Objetivo**

   Mostrar una lista con los ponentes principales del congreso.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Se han recibido bien los datos desde la API y se han guardado en la base de datos interna.

4. **Postcondición si éxito**

   Se muestra una lista con los ponentes, incluyendo su foto, el nombre y el cargo de cada uno.

5. **Postcondición si fallo**

   Se muestra un mensaje informando de que no existen miembros.

6. **Entradas**

   Ninguna.

7. **Salidas**

   La lista de ponentes principales.

8. **Secuencia normal**

   a) Se hace una llamada a la base de datos. Si error, S-1.

   b) Se muestra la lista.

   c) Si no se ha incluído foto, no se muestra.

9. **Secuencia alternativa**

   S-1. No se muestra mensaje de error explícito y no se muestra la lista.

Caso de uso: mostrar un mapa con los lugares donde se celebra el congreso

1. **Objetivo**

   Mostrar un marcador en la ubicación donde se celebrarán ponencias.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Se han recibido bien los datos desde la API y se han guardado en la base de datos interna.

4. **Postcondición si éxito**

   Se muestra un mapa de *Google Maps* con un marcador por cada ubicación, y también una lista con dichas ubicaciones, que se puede seleccionar y abrir una aplicación externa.

5. **Postcondición si fallo**

   Se muestra un mensaje informando de que no existen ubicaciones para mostrar y el mapa vacío.

6. **Entradas**

   Ninguna.

7. **Salidas**

   El mapa con las ubicaciones y una lista.

8. **Secuencia normal**

   a) Se hace una llamada a la base de datos. Si error, S-1.

   b) Si no se muestra el mapa, S-2.

   c) Se muestra la lista.

9. **Secuencia alternativa**

   S-1. No se muestra mensaje de error explícito y no se muestra la lista. S-2. Ha ocurrido un problema externo a nuestra app (falta de OpenGL en el dispositivo, no dispone de conexión a internet...). No se muestra ningún mensaje de error y aparece el mapa gris.

Caso de uso: mostrar información sobre la ciudad donde se celebra el congreso

1. **Objetivo**

   Mostrar un texto y/o imágenes sobre la ciudad donde se celebra el congreso.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Se han recibido bien los datos desde la API y se han guardado en la base de datos interna.

4. **Postcondición si éxito**

   Se muestra el texto con o sin imágenes sobre la ciudad, explicando brevemente sus detalles o costumbres de interés general.

5. **Postcondición si fallo**

   No se muestra nada.

6. **Entradas**

   Ninguna.

7. **Salidas**

   El texto informativo.

8. **Secuencia normal**

   a) Se hace una llamada a la base de datos. Si error, S-1.

   b) Se muestra el texto.

9. **Secuencia alternativa**

   S-1. No se muestra mensaje de error explícito y no se muestra nada.

Caso de uso: mostrar el programa del congreso

1. **Objetivo**

   Mostrar una lista con los horarios de las diferentes conferencias.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Se han recibido bien los datos desde la API y se han guardado en la base de datos interna.

4. **Postcondición si éxito**

   Se muestra una lista de las conferencias, incluyendo el horario, el nombre de la conferencia y una descripción si se incluye. La lista está ordenada con cabeceras de cada día y por hora.

5. **Postcondición si fallo**

   Se muestra un mensaje de que no se ha introducido ningún programa.

6. **Entradas**

   Ninguna.

7. **Salidas**

   La lista con los horarios de las ponencias.

8. **Secuencia normal**

   a) Se hace una llamada a la base de datos. Si error, S-1.

   b) Se muestra la lista.

9. **Secuencia alternativa**

   S-1. No se muestra mensaje de error explícito y no se muestra la lista.

Caso de uso: mostrar una lista de enlaces

1. **Objetivo**

   Mostrar una lista con enlaces a páginas web referentes al congreso.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Se han recibido bien los datos desde la API y se han guardado en la base de datos interna.

4. **Postcondición si éxito**

   Se muestra una lista con los diferentes enlaces y un icono que los identifica.

5. **Postcondición si fallo**

   Se muestra un mensaje de que no hay enlaces disponibles.

6. **Entradas**

   Ninguna.

7. **Salidas**

   La lista de enlaces.

8. **Secuencia normal**

   a) Se hace una llamada a la base de datos. Si error, S-1.

   b) Se muestra la lista.

9. **Secuencia alternativa**

   S-1. No se muestra mensaje de error explícito y no se muestra la lista.

Caso de uso: mostrar información sobre el congreso

1. **Objetivo**

   Mostrar el título del congreso y una breve descripción sobre el mismo.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Se han recibido bien los datos desde la API y se han guardado en la base de datos interna.

4. **Postcondición si éxito**

   Se muestra el título del congreso y un texto con información sobre el evento y los temas que se tratarán en él.

5. **Postcondición si fallo**

   Ninguna.

6. **Entradas**

   Ninguna.

7. **Salidas**

   El texto informativo.

8. **Secuencia normal**

   a) Se hace una llamada a la base de datos.

   b) Se muestra el texto.

9. **Secuencia alternativa**

   Ninguna.

Caso de uso: actualizar la información

1. **Objetivo**

   Hacer una llamada a la API para obtener todos los datos de la base de datos global.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   Existe conexión a internet.

4. **Postcondición si éxito**

   Se descargan todos los datos de la base de datos y se guardan en la base de datos interna del dispositivo.

5. **Postcondición si fallo**

   Se muestra un mensaje de información de que ha ocurrido un error y que vuelva a intentar actualizar pasado un momento.

6. **Entradas**

   Ninguna.

7. **Salidas**

   Información nueva en la app.

8. **Secuencia normal**

   a) El usuario pincha sobre el botón de actualizar que se muestra en la `ActionBar`.

   b) Se descargan los datos a partir de una llamada a la API. Si error, S-1.

   c) Se guardan los datos en la base de datos interna. Si error, S-2.

9. **Secuencia alternativa**

   S-1, S-2. Se muestra un mensaje de que ha ocurrido un error.

Caso de uso: desloguear al usuario

1. **Objetivo**

   Desconectar la cuenta del usuario de la app y salir.

2. **Actores**

   Usuario registrado.

3. **Precondición**

   El usuario está logueado en la app.

4. **Postcondición si éxito**

   Se desconecta del servicio al usuario y se finaliza la app. Al volver a abrir la app deberá introducir sus credenciales.

5. **Postcondición si fallo**

   Ninguna.

6. **Entradas**

   Ninguna.

7. **Salidas**

   Salida de la app.

8. **Secuencia normal**

   a) El usuario picha sobre el menú de la `ActionBar` y después en "Logout".

   b) Se finaliza la ejecución de la app.

9. **Secuencia alternativa**

   Ninguna.

Capítulo 6

Pruebas y Resultados

En este capítulo describiremos un primer prototipo de aplicación móvil que creamos al comienzo y la prueba de la aplicación realizada en un congreso.

6.1. Prototipo

Durante la primera fase de desarrollo, cuando se decidió qué tipo de aplicación móvil queríamos desarrollar, las directoras de proyecto nos propusieron realizar un prototipo para las Jornadas de Gerencia Universitaria [19] que se realizaron en Noviembre del 2013. De esta forma podríamos obtener opiniones de usuarios reales para saber hacia dónde enfocar más nuestro trabajo.

Decidimos realizar una aplicación para dispositivos Android básica, sin ningún servicio web por detrás. Simplemente cogimos toda la información que se encontraba en su página web y la presentamos de manera sencilla en una aplicación con 3 secciones: programa, ponentes e información.

- Programa. Presentaba la información de la agenda de las Jornadas del mismo modo que decidimos hacerlo para nuestra aplicación final, con las diferentes conferencias separadas por día, con su horario y una breve descripción su fuese necesario.
- Ponentes. Una lista de los ponentes principales con una fotografía, su nombre completo y su puesto de trabajo dentro de la universidad.
- Info. Incluía la dirección de las localizaciones donde se realizaban las conferencias con un botón para abrir la aplicación de *Google Maps* para verlas en un mapa y poder obtener direcciones, y un email de contacto y teléfonos para obtener cualquier información.

En la figura 6.1 se pueden ver capturas de pantalla de la aplicación.

Tuvimos una reunión junto con nuestras directoras con un organizador de las Jornadas de Gerencia para presentarle la aplicación y dar el visto bueno para utilizarla durante el evento.

(a) Pantalla Inicial (b) Programa

Figura 6.1: Pantallas del prototipo para las Jornadas de Gerencia Universitaria Capturas del prototipo

Aunque fuese un prototipo bastante simple, nos dio una imagen sobre cómo debíamos continuar y en qué aspectos debíamos mejorar, tanto de la propia aplicación como de nuestra organización y forma de trabajar. A partir de este momento comenzamos con el diseño de nuestra aplicación final.

6.2. Prueba de la aplicación en el PIC 2014

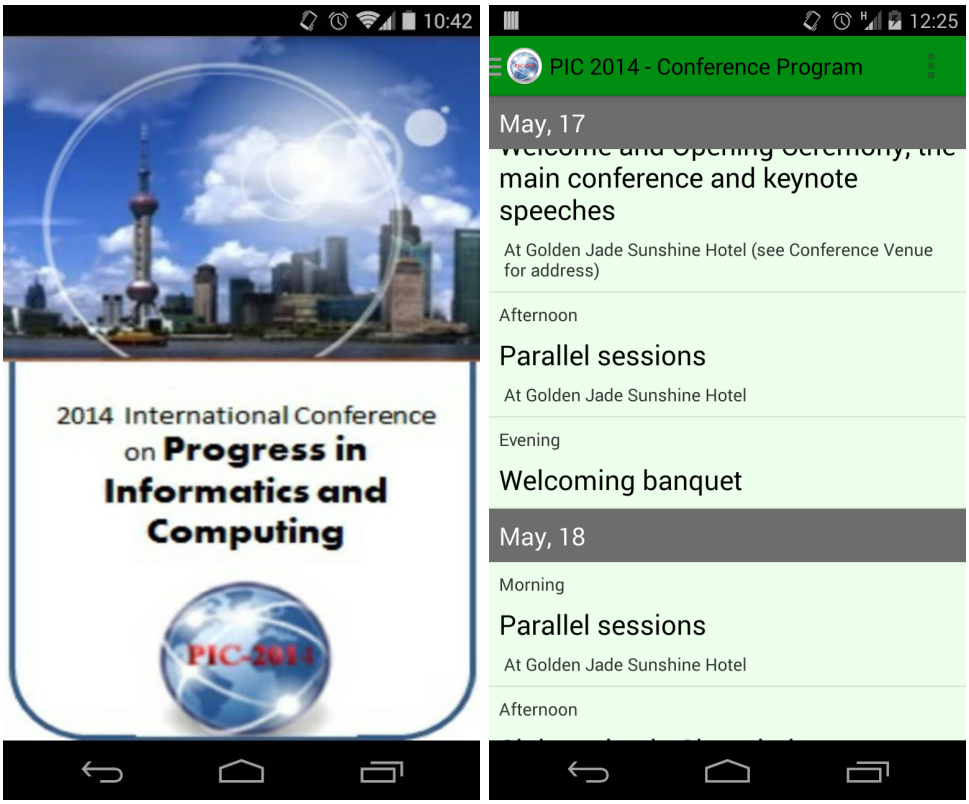
A mediados de Febrero las directoras nos ofrecieron la posibilidad de dar a conocer nuestro proyecto en un congreso real de caracter internacional, el PIC 2014 (*International Conference on Progress in Informatics and Computing*) [20], que se realizó entre el 16 y el 18 de Mayo del presente año. Esto nos obligaba a tener la versión final del proyecto a finales de Abril, para poder probar todo y enviar los resultados antes de que diera comienzo el congreso.

Aunque esto nos supuso más presión añadida de lo que ya es un proyecto de fin de carrera pudimos cumplir los plazos y entregar una versión final tanto del servicio web como de la aplicación móvil. En este caso, y al ser la primera prueba, fuimos nosotros quienes utilizamos el formulario web para introducir toda la información

y también nos dimos cuenta de algunas carencias. Había aspectos que se podían mejorar y que más adelante se modificaron para la entrega final de este proyecto, pero era precisamente para eso la importancia de probarlo en un congreso real, para que los usuarios pudieran probarlo y ver qué errores sucedían para corregirlos.

Cuando finalizó tuvimos una reunión para ver qué era necesario modificar para hacer todavía más sencillo el trabajo de los organizadores a la hora de introducir los datos. Dichos cambios se modificaron y se empezó la planificación para probarlo en un segundo congreso que se celebraría durante el mes de Julio, ya dejando todo en manos de los organizadores.

En la figura 6.2 se muestran capturas de pantalla de la aplicación.



(a) Pantalla Inicial (b) Agenda

Figura 6.2: Capturas de la aplicación para el PIC 2014

Capítulo 7

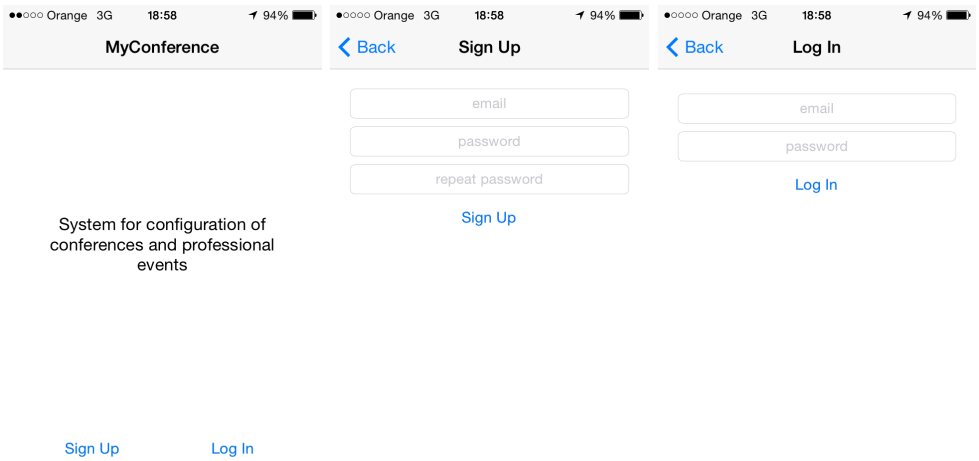
Trabajo Futuro

En este capítulo se explica el trabajo que queda pendiente para el futuro.

7.1. Aplicación Móvil para iOS

La idea inicial era desarrollar la aplicación móvil tanto para sistemas Android como para sistemas iOS, pero debido a falta de tiempo no se ha podido desarrollar la aplicación para iOS, aunque sí se comenzó con su desarrollo.

Actualmente, la aplicación iOS solo tiene desarrollada la base del proyecto para que pueda funcionar. Esta base incluye la configuración del proyecto (versiones del sistema que admite, dispositivos en los que se puede ejecutar la aplicación, etc), funcionalidad que permite registrar o iniciar sesión en la aplicación y un diseño básico de la interfaz para que se adapte correctamente a todos los tipos de dispositivo que admite la aplicación.



(a) Pantalla Principal (b) Registro (c) Identificación

Figura 7.1: Capturas de pantalla de la aplicación iOS

A continuación se detallan las partes de la aplicación que faltan por implementar:

- Mostrar la lista de congresos del usuario en cuanto este inicie sesión en la aplicación.
- Cuando el usuario pulse sobre un congreso se mostraría un vista con la información resumida acerca del congreso que acaba de seleccionar y un menú que permita poder navegar entre las distintas secciones del congreso. La idea es que el menú sea el nativo que proporciona el SDK de iOS, que es una barra que aparece en la parte inferior de la pantalla con distintos items de navegación.
- Implementar cada una de las secciones del congreso, es decir, la sección de usuarios invitados, los anuncios del congreso, los documentos útiles, los lugares de interés, los organizadores y ponentes del congreso y la agenda.
- Implementar el sistema de cierre de sesión para informar a la API de que el usuario no va a seguir usando la sesión actual.
- Internacionalizar la aplicación para que pueda soportar varios idiomas. Actualmente la aplicación solo está disponible en inglés.
- Diseñar una base de datos interna que almacene los datos que se recogen de la API para mejorar el rendimiento de la aplicación.

La figura 7.1 muestra capturas de la aplicación iOS.

7.1.1. Requisitos

Figura 7.2: Logo de Xcode

El software utilizado para el desarrollo de la aplicación iOS es Xcode, un entorno de desarrollo utilizado para implementar aplicaciones tanto para iOS como para MacOS. Se puede descargar directamente desde el App Store y solo requiere tener creada una cuenta en iTunes para tener un Apple ID con el que registrar la descarga.

Xcode incluye todos los SDKs nativos de iOS necesarios para poder desarrollar la aplicación. También incluye simuladores para poder probar la aplicación si no se dispone de un iPhone o un iPad. Hay simuladores para todos los tipos de iPhone y de iPad incluyendo todas las versiones disponibles del sistema iOS para cada tipo de dispositivo.

Es necesario disponer de un Mac ya que Xcode solo está disponible para estos ordenadores. También es recomendable disponer de un iPhone o un iPad para realizar las pruebas sobre dispositivos reales porque, aunque los simuladores proporcionados por Xcode funcionan bastante bien, son limitados con algunas características del sistema y no simulan al 100 % un dispositivo real.

Apéndice A

Guía de Despliegue

Para poder continuar con el desarrollo o desplegar el proyecto en su forma actual, hace falta realizar una serie de pasos en cada una de las piezas que forman MyConference.

Tanto para la API como para la Web es necesario disponer de un servidor web capaz de ejecutar aplicaciones escritas para Node.js (Sección 2.2.1), e idealmente ambas usarán diferentes servidores. Una de las opciones más sencillas es utilizar alguna *Plataforma como Servicio*, como por ejemplo Heroku (Sección 2.1.1), pero también es posible configurarlo para funcionar en un servidor privado. Para ambas aplicaciones es necesario disponer de una base de datos MongoDB (Sección 2.3.1). Aunque las colecciones usadas son disjutas y por tanto técnicamente es posible usar una única base de datos, es conveniente por motivos de seguridas utilizar dos bases de datos independientes, una para la Web y otra para la API.

Para la app de Android solo es necesario disponer de un IDE compatible con Android, como por ejemplo Eclipse (Sección 2.5.2). Una vez creada la aplicación, si se desea publicar en Google Play o en algún otro servicio similar, hará falta firmar y alinear (mediante `zipalign`) el paquete generado, generalmente mediante la opción de Exportar APK del IDE elegido.

A.1. Despliegue de la API

La API, descrita en el capítulo 3, es la columna vertebral de MyConference, y por tanto será la primera parte del sistema que se instalará y configurará.

En primer lugar es necesario tener instalado Node.js, versión `0.12` o superior. A continuación debemos, usando un terminal y desde la carpeta raíz del proyecto, ejecutar:

```
$ npm install
```

Esto descargará e instalará las dependencias necesarias para ejecutar la API en sutotalidad. Para hacerlo, ejecutaremos:

```
$ npm start
```

Sin embargo, en este punto probablemente veamos algún error en la consola, debido a que la aplicación no ha sido configurada en absoluto. Para su configuración local se utilizará Foreman (Véase 2.6.1), mientras que para el despliegue final utilizaremos Heroku.

A.1.1. Configuración con Foreman

En primer lugar deberemos descargar e instalar `foreman`. Dependiendo de la plataforma el proceso puede variar, por lo que no entraremos en detalles.

Tras la instalación, se necesitan dos ficheros: `Procfile` y `.env`. El fichero `Procfile` define los prcesos que se ejecutarán con la aplicación, y ya está creado. El fichero `.env` contiene las variables de configuración en forma de variables de entorno con la siguiente sintaxis:

```
VAR1=valor 1
VAR2=valor 2
```

Este fichero **nunca** debe añadirse a ningún sistema de control de versiones, y será diferente en cada máquina (desarrollo, pruebas, producción).

Una vez definidos estosdos ficheros, lanzaremos la aplicación usando el comando

```
$ foreman start
```

Por defecto, Foreman establece la variable de etorno `PORT` a 5000, e iráincrementando este valor si lanzamosvariosprocesos. Para cambiar el valor inicial de esta variable, podemos utilizar la opción `-p`. Si queremos que se utilice el puerto 80 para obtener conectividad HTTP directa, deberemos lanzar el servidor con privilegios de superusuario:

```
# foreman start -p 80
```

A.1.2. Configuración de MongoDB

Para que la API funcione correctamente, necesita conectividad con una base de datos MongoDB.

Para configurar qué base de datos utilizar, se deben establecer la variable `MONGO_URI` a una URI con esquema (protocolo) de tipo `mongodb:`, como por ejemplo `mongodb://root:root@example.org/myconference`. También se aceptará este valor si se utiliza la variable `MONGOLAB_URI`, como se explicará en el punto A.1.5.

La base de datos en sí no necesita más configuración. Una vez la API sepa qué host, usuario y contraseña debe utilizar, la base de datos está completamente configurada.

A.1.3. Configuración de Mandrill

Para permitir el envío de correos electrónicos, MyConference utiliza Mandrill, un servicio de envío de correo electrónico en la nube.

Para configurar la cuenta de usuario de Mandrill que se utilizará, se deben establecer las variables `MANDRILL_USERNAME` y `MANDRILL_APIKEY` con el nobre de usuario y la API-key de Mandrill respectivamente.

Una vez la API tiene acceso a la cuenta de Mandrill, es necesario crear una plantilla en el panel de control de Mandrill llamada `invitation-code`, que contendrá el correo enviado a los usuarios cuando se les invite a un congreso de MyConference. Esta plantilla utiliza las siguientes *merge variables*:

- `CONFNAME`: El nombre completo del congreso.
- `JOINURL`: La URL de la página en la que el usuario puede unirse a un congreso
- `JOINCODE`: El código que el usuario debe introducir en dicha URL

Nótese que mediante `*|JOINURL|*?code=*|JOINCODE|*` podemos obtener la URL completa de la página en la que el usuario se unirá a la conferencia, con el código ya introducido para que el usuario simplemente acepte.

A continuación se muestra un código básico que se puede usar en esta plantilla.

```
<h1>You have been invited to join *|CONFNAME|* on MyConference</h1>
<p>
  <a href="*|JOINURL|*?code=*|JOINCODE|*">Click Here</a>
  to join or enter the code *|JOINCODE|* in the
  <a href="*|JOINURL|*">Join Conference</a>
  view on the MyConference Web or Mobile App.
</p>
```

A.1.4. Configuración General

Además de la configuración anterior, existen otras variables que debemos configurar para que la API funcione correctamente:

- `HOST`: El nombre de dominio (host) completo de la API, como por ejemplo `api.myconference.com`. Nótese que no deben aparecer separadores de ruta (`/`) ni prefijos de esquema (`http://`).
- `NODE_ENV`: Debe contener `production` en entornos de producción y `development` en desarrollo.
- `WEB_URL`: Debe contener la URL completa a la aplicación web de MyConference, por ejemplo `http://myconference.com`. Nótese que no debe aparecer el separador final de ruta (`/`) pero sí el prefijo de esquema (`http://`).

Tras realizar toda la configuración, hay que realizar un paso más: Crear las entradas de las aplicaciones por defecto. Los siguientes comandos, ejecutados en orden, nos mostrarán los IDs de dichas aplicaciones, que se habrán creado si no existían:

```
$ cat .env | eval
$ export MONGO_URI MONGOLAB_URI
$ node scripts/get_or_create_apps.js
$ unset MONGO_URI MONGOLAB_URI
```

Debemos apuntar estos datos puesto que serán necesarios para configurar las aplicaciones Web, Android e iOS

Con esto, la API está lista para funcionar.

A.1.5. Heroku

El desarrollo original de MyConference ha sido probado y puesto en marcha en Heroku (Véase 2.1.1) desde el primer día, y todo el sistema de configuración se integra a la perfección con Heroku.

Para desplegar la API en Heroku, el primer paso es registrarse y crear una aplicación. Tras esto pasamos a configurar algunos add-ons necesarios:

MongoLab es el proveedor de MongoDB elegido.

Puesto que MongoLab no requiere ninguna configuración, bastará con añadir a la aplicación una instancia *Sandbox* gratuita.

Se ha de tener en cuenta que el código actualmente está preparado para leer la variable de configuración `MONGOLAB_URI` que establece MongoLab automáticamente, por lo que si se utilizara otro servicio habría que modificar dicho código o establecer la variable `MONGO_URI` manualmente.

Mandrill by MailChimp es el servicio elegido para el envío de correo electrónico.

Debemos añadir a la aplicación una instancia (puede ser *Starter*, gratuita) de Mandrill. Al hacerlo se añadirán a la aplicación las variables `MANDRILL_USERNAME` y `MANDRILL_APIKEY`.

Para que este add-on se integre correctamente, se debe realizar la configuración de Mandrill tal como se describe en el punto A.1.3.

A continuación faltará realizar la configuración descrita en el punto A.1.4.

Heroku tiene su propia forma de realizar la configuración en lugar de utilizar un fichero `.env`. En primer lugar debemos instalar la utilidad de Heroku para línea de comandos, tal como se explica en `https://devcenter.heroku.com/articles/heroku-command`. Una vez hecho esto, procederemos a realizar la configuración.

Para establecer variables de configuración utilizaremos el siguiente comando:

```
$ heroku config:set VAR1="valor 1" VAR2="valor 2"
```

En este caso, debemos ejecutar un comando similar a este:

```
$ heroku config:set \
      NODE_ENV=production \
      HOST=api.myconference.com \
      WEB_URL=http://myconference.com
```

Una vez tenemos la configuración y los add-ons, es hora de desplegar la aplicación. Para ello, usando Git, desde la raíz de nuestro proyecto, ejecutaremos:

```
$ git push heroku master
```

Debemos haber configurado el código para utilizar `git`, y haber apuntado el *remote* `heroku` a la URL del repositorio de nuestra aplicación en Heroku (por ejemplo, `git@heroku.com:myconference-api.git`).

Una vez termine el despliegue de la aplicación, falta configurar los datos iniciales de la aplicación, mediante el siguiente comando:

```
$ heroku run 'node scripts/get_or_create_apps.js'
```

De nuevo obtendremos datos que debemos apuntar, puesto que serán necasios para el resto de aplicaciones.

A.2. Despliegue de la Web

La web de MyConference, descrita en el capítulo 3.3, es la única aplicación que permite la introducción de datos en el sistema, más allá de la creación de nuevos usuarios.

Su despliegue es muy similar al de la API, descrito en el punto A.1, puesto que utilizan las mismas tecnologías básicas.

De igual manera que en la API, lo primero que hemos de hacer es instalar las dependencias, ejecutando el siguiente comando:

```
$ npm install
```

Tras esto pasaremos a la configuración con Foreman y, posteriormente, Heroku.

Para una breve introducción al uso de Foreman, véase la sección A.1.1 Configuración con Foreman.

A.2.1. Configuración de MongoDB

Para que la web funcione correctamente, necesita conectividad con una base de datos MongoDB. A diferencia de la base de datos de la API, esta base de datos se utiliza úncamente para persistir información de la sesión de usuario, nunca para almacenar datos de MyConference.

Para configurar qué base de datos utilizar, se deben establecer la variable `MONGO_URI` a una URI con esquema (protocolo) de tipo `mongodb:`, como por ejemplo `mongodb://root:root@example.org/myconference`. También se aceptará este valor si se utiliza la variable `MONGOLAB_URI`, como se explicará en el punto A.2.4.

La base de datos en sí no necesita más configuración. Una vez la web sepa qué host, usuario y contraseña debe utilizar, la base de datos está completamente configurada.

A.2.2. Configuración de Transloadit

Para que el sistema de subida de archivos funcione correctamente, la web necesita conectividad con una cuenta de Transloadit (Véase 2.2.6).

Transloadit utiliza las siguientes variables de configuración:

- `TRANSLOADIT_AUTH_KEY`: Clave de autorización de la cuenta de Transloadit.
- `TRANSLOADIT_SECRET_KEY`: Clave secreta de la cuenta de Transloadit.
- `TRANSLOADIT_URL`: URL de la API de Transloadit (`https://api2.transloadit.com`).
- `TRANSLOADIT_TEMPLATE_SPEAKERS`: Identificador de la plantilla usada para la carga de imágenes de *keynote speakers*, creada más adelante.

Una vez establecidas las variables, debemos configurar Transloadit. Para ello necesitamos crear la plantilla que usaremos para la carga de imágenes. Simplemente crearemos una nueva plantilla e introduciremos lo siguiente:

```
{
  "steps": {
    "resize_xxhdpi": {
      "robot": "/image/resize",
      "use": ":original",
      "width": 240,
      "height": 300
    },
    "export": {
      "robot": "/ftp/store",
      "use": "resize_xxhdpi",
      "host": "ftp.example.org",
      "user": "myconference-user",
      "password": "myconference-password",
      "path": "/${file.id}.${file.ext}",
      "url_template":
        "http://example.org/path/to/${file.id}.${file.ext}"
    }
  }
}
```

Los datos han de ser ajustados para utilizar una cuenta de FTP existente, así como para que la URL permita una descarga directa de la imagen.

Una vez tengamos la platilla creada, añadiremos el ID generado a la variable de configuración `TRANSLOADIT_TEMPLATE_SPEAKERS`.

A.2.3. Configuración General

Además de la configuración anterior, existen otras variables que debemos configurar para que la web funcione correctamente:

- `NODE_ENV`: Debe contener `production` en entornos de producción y `development` en desarrollo.
- `API_URL`: Debe contener la URL completa a la API de MyConference, por ejemplo `http://api.myconference.com`. Nótese que no debe aparecer el separador final de ruta (`/`) pero sí el prefijo de esquema (`http://`).
- `APPLICATION`: Debe contener el ID de aplicación asociado a la web, obtenido previamente al configurar la API en el paso de creación de aplicaciones.

Con esto, la web está completamente lista para funcionar.

A.2.4. Heroku

El desarrollo original de MyConference ha sido probado y puesto en marcha en Heroku (Véase 2.1.1) desde el primer día, y todo el sistema de configuración se integra a la perfección con Heroku.

Para desplegar la web en Heroku, el primer paso es registrarse y crear una aplicación. Tras esto pasamos a configurar algunos add-ons necesarios:

MongoLab es el proveedor de MongoDB elegido.

Puesto que MongoLab no requiere ninguna configuración, bastará con añadir a la aplicación una instancia *Sandbox* gratuita.

Se ha de tener en cuenta que el código actualmente está preparado para leer la variable de configuración `MONGOLAB_URI` que establece MongoLab automáticamente, por lo que si se utilizara otro servicio habría que modificar dicho código o establecer la variable `MONGO_URI` manualmente.

Transloadit es el servicio elegido para la gestión de subidas de imágenes.

Debemos añadir a la aplicación una instancia de Transloadit. Al hacerlo se añadirán a la aplicación las variables `TRANSLOADIT_URL`, `TRANSLOADIT_AUTH_KEY` y `TRANSLOADIT_SECRET_KEY`.

Para que este add-on se integre correctamente, se debe realizar la configuración de Transloadit tal como se describe en el punto A.2.2, incluyendo la variable de configuración `TRANSLOADIT_TEMPLATE_SPEAKERS`.

A continuación faltará realizar la configuración descrita en el punto A.2.3.

Para ello utilizaremos la herramienta de línea de comandos de Heroku, descrita en el punto A.1.5.

En este caso, debemos ejecutar un comando similar a este:

```
$ heroku config:set \
    NODE_ENV=production \
    APPLICATION=webapplicationid \
    API_URL=http://api.myconference.com
```

Una vez tenemos la configuración y los add-ons, es hora de desplegar la aplicación. Para ello, usando Git, desde la raíz de nuestro proyecto, ejecutaremos:

```
$ git push heroku master
```

Debemos haber configurado el código para utilizar `git`, y haber apuntado el *remote* `heroku` a la URL del repositorio de nuestra aplicación en Heroku (por ejemplo, `git@heroku.com:myconference-web.git`).

A.3. Despliegue de la App Android

La app de MyConference, descrita en el capítulo 4, es la encargada de mostrar todos los datos introducidos a partir de la web.

Para su desarrollo solo hace falta descargarse el proyecto del repositorio alojado en Git [21].

El IDE utilizado para este proyecto ha sido Eclipse. Basta con importar el proyecto al *workspace* y automáticamente lo compilará. Dentro del repositorio ya se incluyen las librerías necesarias para su ejecución.

Para testear la app es necesario un teléfono Android con versión 2.3 o superior, ya que el emulador disponible con el SDK no puede ejecutar nada relacionado con Google Maps.

La aplicación hace uso de algunos valores que pueden cambiar dependiendo de cómo se haya desplegado la API en la sección A.1. Por ello debemos modificar apropiadamente los valores de los siguientes campos en la clase `es.ucm.myconference.util.Constants`:

`APP_ID` : Identificador de la aplicación android obtenido en el despliegue de la API.

`API_URL` : URL base para el acceso a todas las rutas de la API (por ejemplo, `http://api.myconference.com`).

Una vez realizados estos cambios, la aplicación debe compilarse en modo producción. Para ello, hacemos click derecho en el proyecto y, dentro de la opción de menú específica para Android, elegimos *Exportar APK firmado*. Se pedirá un certificado para firmar la aplicación. Este certificado es fundamental y debe ser siempre el mismo para que Google Play nos permita actualizar la aplicación.

Apéndice B

Guía de uso

En este apéndice se explica el funcionamiento del formulario Web que permite la gestión de los congresos. Está separado en tres apartados: el acceso a la aplicación, la visualización de los datos de los congresos creados por el usuario o a los que ha sido invitado, y por último, la creación de congresos y de sus datos.

B.1. Acceso a la aplicación

En esta sección se muestra como un usuario puede acceder a la aplicación, ya sea registrando una cuenta de usuario nueva si este no posee una o iniciando sesión con una cuenta de usuario registrada previamente.

En la figura B.1 se muestra la página inicial de la web.

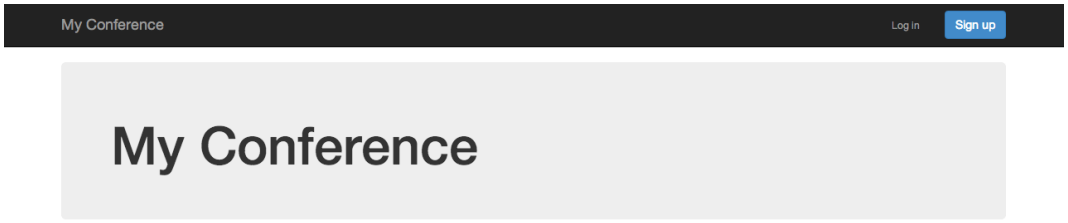


Figura B.1: Página inicial de la web.

B.1.1. Registro

Es necesario tener registrada una cuenta de usuario para poder usar la aplicación. Se puede crear una cuenta nueva pulsando sobre el botón **Sign up** y rellenando un simple formulario con los siguientes campos obligatorios:

- E-mail: debe tener un formato válido y que no haya sido registrado previamente en la aplicación.

- Password: una contraseña que contenga al menos 8 caracteres.
- Repeat password: repetición de la contraseña para evitar fallos. Debe coincidir con la contraseña introducida previamente.

En la figura B.2 se muestra el formulario de registro.

Figura B.2: Formulario de *registro*.

B.1.2. Inicio de sesión

Si ya se posee una cuenta de usuario registrada, se puede iniciar sesión pulsando en el botón **Log in** e introduciendo un e-mail registrado previamente en la aplicación con su correspondiente contraseña.

En la figura B.3 se muestra el formulario de inicio de sesión.

Figura B.3: Formulario de *inicio de sesión*.

B.2. Visualización de congresos

Una vez se ha iniciado sesión en la aplicación, ya se puede acceder a toda la información relacionada con la cuenta. Nada más acceder se muestra una página principal con la lista de todos los congresos que están relacionadas con el usuario, tanto los creados por el propio usuario como a los que ha sido invitado a asistir.

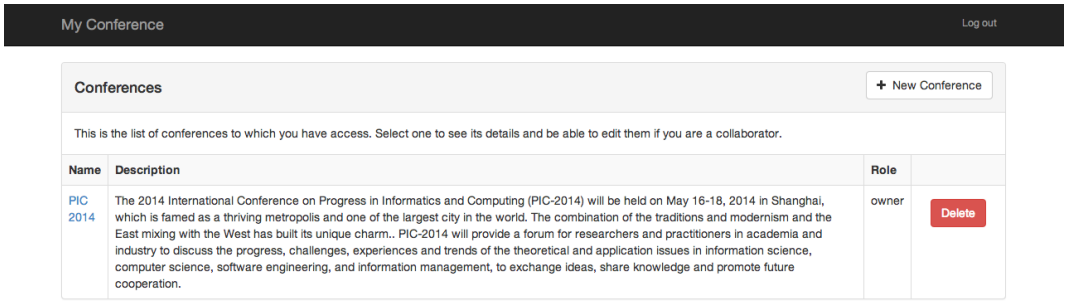


Figura B.4: Página con la lista de los congresos del usuario.

En la figura B.4 se muestra la página principal con la lista de los congresos.

En esta página principal se muestra la siguiente información:

- Nombre del congreso.
- Breve descripción del congreso.
- Papel que tiene el usuario en el congreso (propietario o asistente).
- Botón para poder borrar un congreso creado en caso de que el usuario sea su propietario.

Pulsando en el nombre de un congreso de la lista se accede a su información detallada. Una vez se ha accedido al congreso, en la barra de navegación aparece el nombre del congreso, las diferentes secciones en las que se divide la información y el botón **Log out** para poder cerrar la sesión actual. Pulsando en el nombre del congreso en la barra de navegación se vuelve a la lista de congresos.

En la figura B.5 se muestra la página principal de cada congreso.

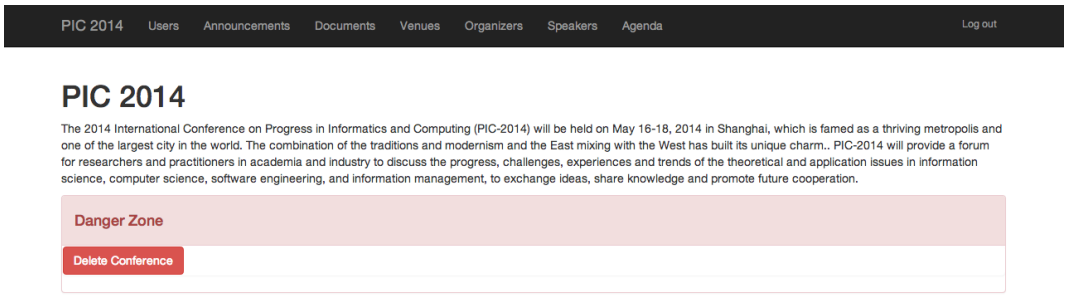


Figura B.5: Página principal de un congreso.

A continuación se describen las distintas secciones.

B.2.1. Usuarios

En esta sección se pueden ver los usuarios que han sido invitados al congreso y han aceptado la invitación. El propietario del congreso es el único que puede mandar invitaciones a otros usuarios. Estos usuarios tendrán acceso a todos los datos del congreso con el papel de asistentes, por lo que no podrán ni añadir ni borrar ningún tipo de información.

Los campos que se pueden ver en esta sección son:

- ID: el ID asignado al usuario. Actualmente los usuarios no poseen un nombre de usuario en sus cuentas, por lo que la única manera de identificar a un usuario es por su ID.
- Role: el papel que ejerce el usuario.

En la figura B.6 se muestra la sección de usuarios de un congreso.

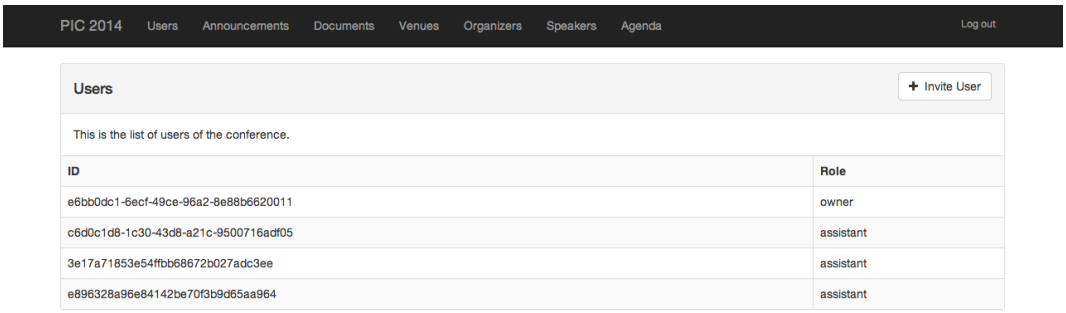


Figura B.6: Sección de *usuarios*.

B.2.2. Anuncios

Esta sección muestra los anuncios que se publican para mantener a los usuarios al día con las novedades del congreso.

Los anuncios llevan la siguiente información:

- Title: título descriptivo del anuncio.
- Body: contenido completo del anuncio.
- Date: fecha en la que ha sido publicado el anuncio.

En la figura B.7 se muestra la sección de anuncios de un congreso.

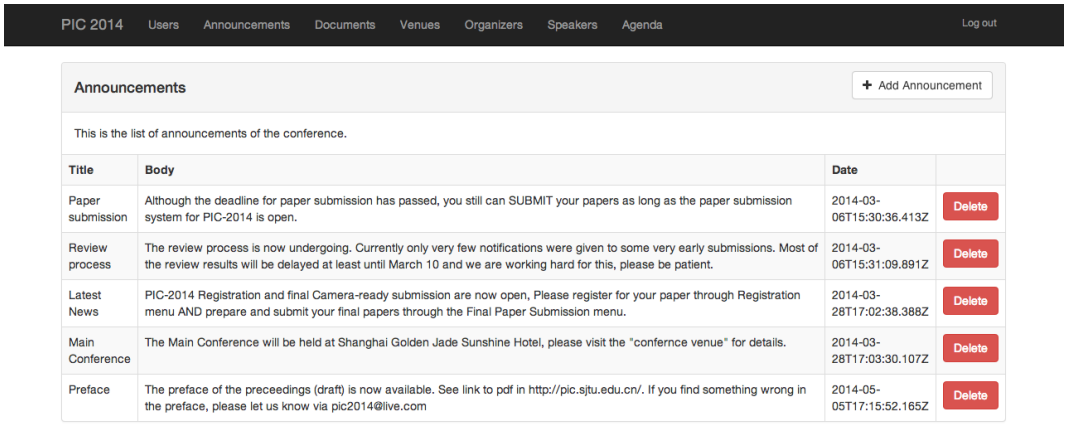


Figura B.7: Sección de *anuncios*.

B.2.3. Documentos

Muestra los documentos importantes del congreso. Los documentos se muestran como enlaces a la web del congreso donde se encuentra dicha información.

Los documentos tienen los siguientes campos:

- Title: título del documento.
- Description: descripción del contenido del documento.
- Data: enlace al documento.

En la figura B.8 se muestra la sección de documentos de un congreso.

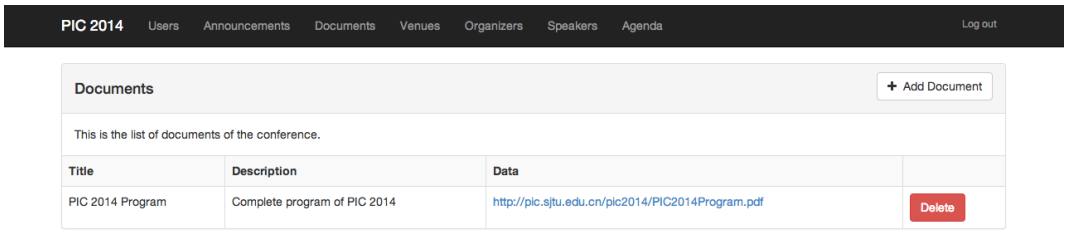


Figura B.8: Sección de *documentos*.

B.2.4. Lugares

Muestra los lugares de interés de cara al congreso, como por ejemplo, los sitios donde se realizarán las ponencias, los hoteles donde se pueden alojar los asistentes, etc.

Un lugar viene indicado por:

- Name: nombre del lugar.
- Location: localización del lugar dada por sus coordenadas geográficas y su dirección.
- Details: detalles de interés del lugar.

En la figura B.9 se muestra la sección con los lugares del congreso.

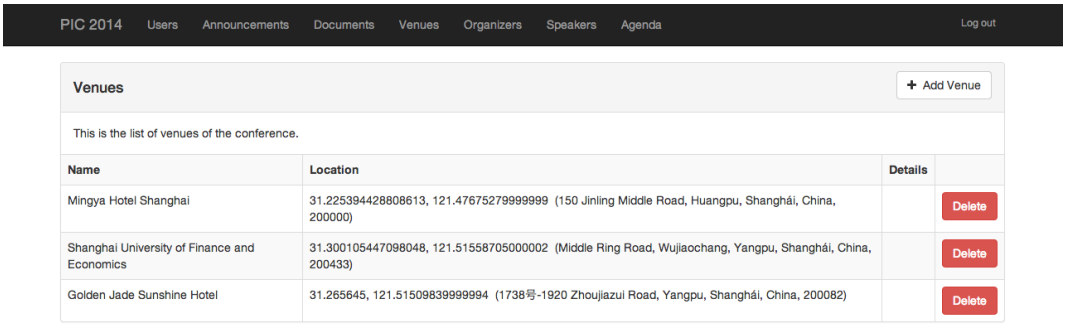


Figura B.9: Sección de *lugares*.

B.2.5. Organizadores

Esta sección muestra las personas encargadas de la organización del congreso.
Cada organizador tiene la siguiente información:

- Name: nombre del organizador.
- Origin: procedencia del organizador.
- Details: detalles de interés del organizador.
- Group: grupo al que pertenece el organizador (comité organizador, comité científico, etc).

En la figura B.10 se muestra la sección con los organizadores del congreso.

B.2.6. Ponentes

Muestra los encargados de dar las ponencias en el congreso.
Cada ponente tiene la siguiente información:

- Name: nombre del ponente.
- Origin: procedencia del ponente.
- Charge: cargo del ponente.
- Description: una breve biografía del ponente.

En la figura B.11 se muestra la sección con los ponentes del congreso.

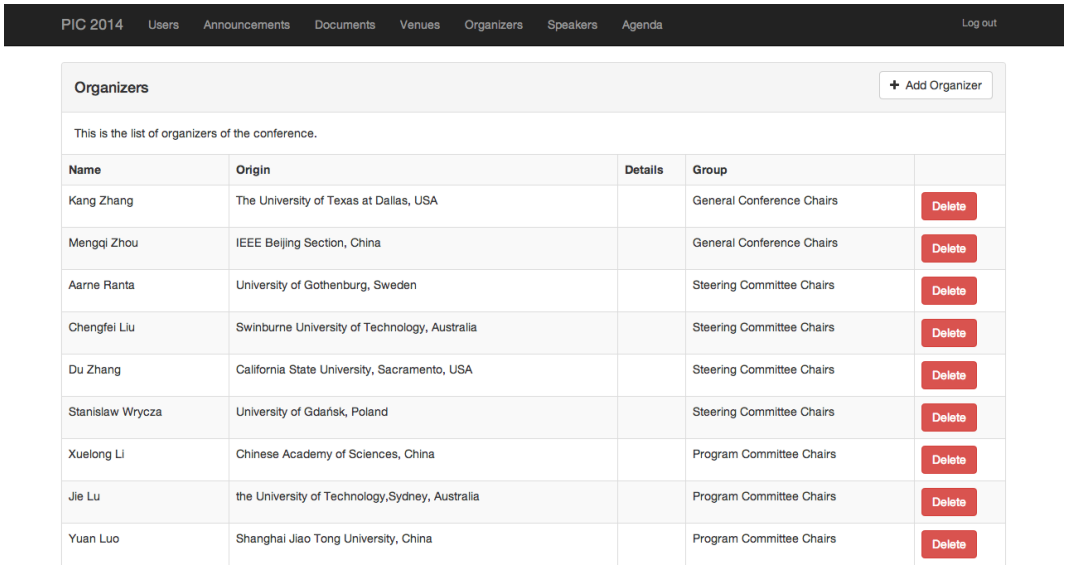


Figura B.10: Sección de *organizadores*.

B.2.7. Agenda

Esta sección muestra la agenda del congreso, es decir, los eventos que se realizarán durante el congreso.

Cada evento se detalla por:

- Title: título del evento.
- Descripción: breve descripción del evento.
- Date & Time: fecha en la que se realizará el evento.

En la figura B.12 se muestra la sección con los eventos de la agenda del congreso.

B.3. Creación de congresos

En esta sección se explica como poder crear un congreso desde cero, creando primero el congreso en sí con su información básica y posteriormente ir añadiendole sus datos correspondientes en cada sección.

Desde la página principal con la lista de congresos se pueden crear nuevos congresos pulsando sobre el botón + **New Conference** e introduciendo los siguientes campos:

- Name: nombre del congreso.
- Descripción: breve descripción del congreso.

En la figura B.13 se muestra el formulario que permite crear nuevos congresos.

Inicialmente, el congreso sólo tendrá esta información. Posteriormente se irá añadiendo nueva información desde cada una de las secciones vistas anteriormente.

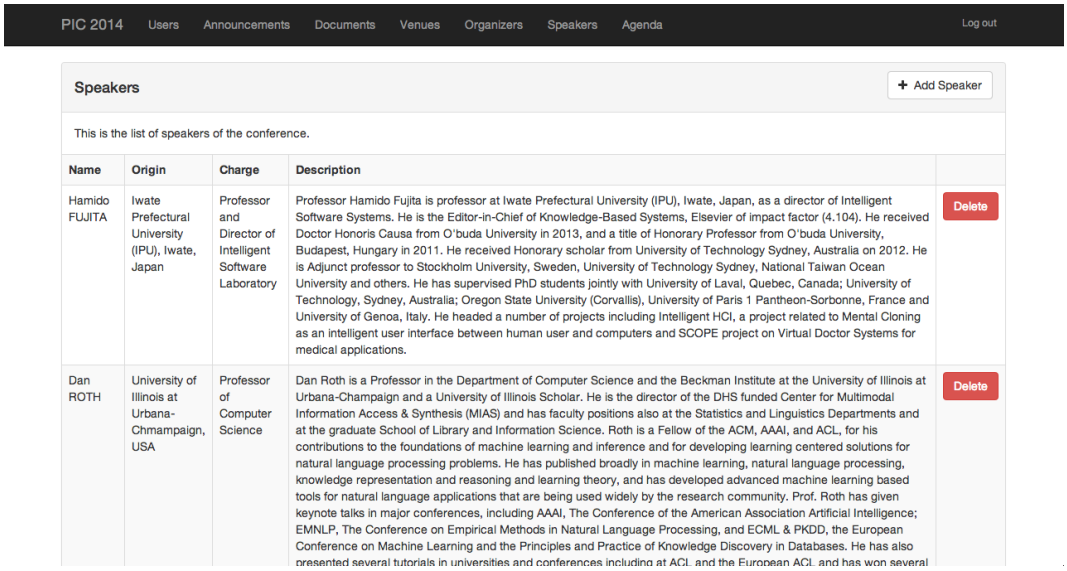


Figura B.11: Sección de *ponentes*.

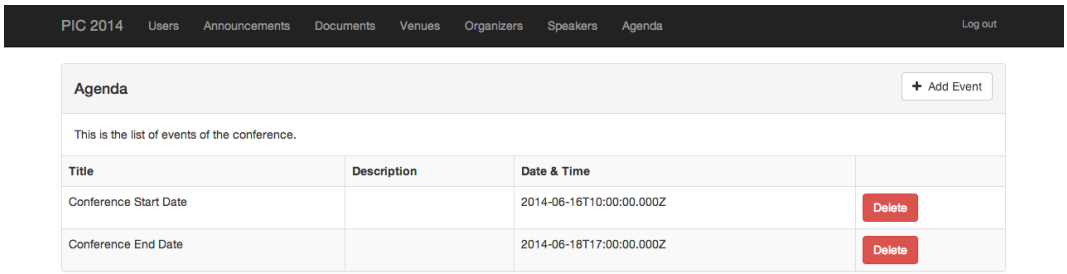


Figura B.12: Sección de la *agenda*.

B.3.1. Usuarios

El propietario del congreso es el único que puede invitar a otros usuarios al congreso. Los usuarios invitados tendrán el papel de asistentes y podrán acceder a toda la información del congreso desde sus propias cuentas, pero no podrán añadir ni borrar ningún tipo de información.

Se pueden enviar invitaciones desde la sección de usuarios pulsando el botón + **Invite User** e introduciendo el nombre y el e-mail del usuario al que se quiere invitar.

En la figura B.14 se muestra al formulario que permite enviar invitaciones a otros usuarios.

Una vez se ha mandado la invitación, al usuario le llegará un correo con un código de activación que permite vincular el congreso a su cuenta de usuario. En cuanto el usuario introduce el código de activación, éste podrá acceder de manera inmediata a

Figura B.13: Formulario de *creación de nuevos congresos.*

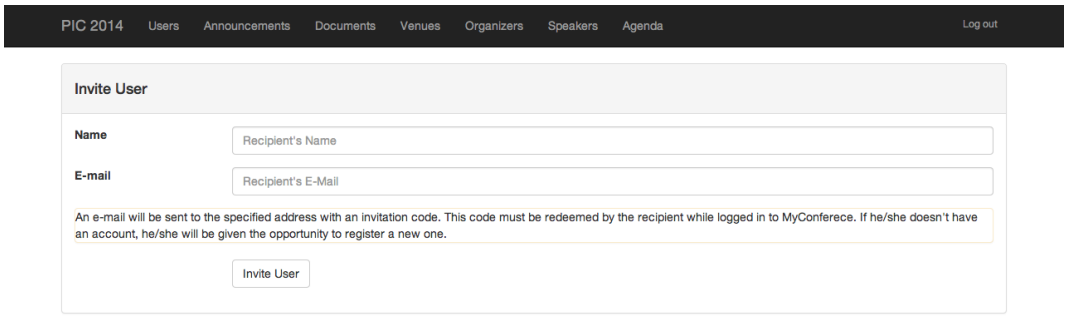


Figura B.14: Formulario de *invitación de usuarios.*

la toda la información y aparecerá automáticamente en la lista de usuarios asistentes al congreso.

En la figura B.15 se muestra el e-mail que le llega al usuario que ha recibido una invitación.

Figura B.15: Correo con el *código de activación.*

B.3.2. Anuncios

Se pueden añadir nuevos anuncios pulsando sobre el botón + **Add Annoucement** en la sección de anuncios y rellenando el siguiente formulario:

- Title: título del anuncio.
- Body: contenido completo del anuncio.

La fecha del anuncio se añade automáticamente y será la fecha actual en la que se ha publicado el anuncio.

En la figura B.16 se muestra el formulario que permite añadir nuevos anuncios al congreso.

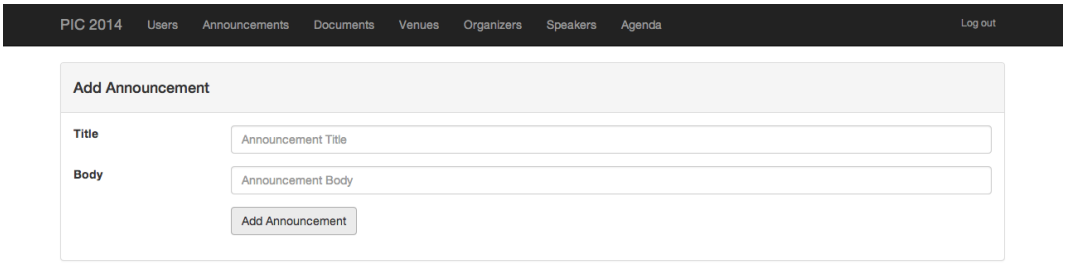


Figura B.16: Formulario para añadir nuevos *anuncios*.

B.3.3. Documentos

Para añadir nuevos documentos al congreso hay que pulsar el botón + **Add Document** en la sección de documentos e introduciendo los siguientes campos:

- Title: título del documento.
- Description: descripción del documento.
- Data: enlace al documento.

En la figura B.17 se muestra el formulario que permite añadir nuevos documentos al formulario.

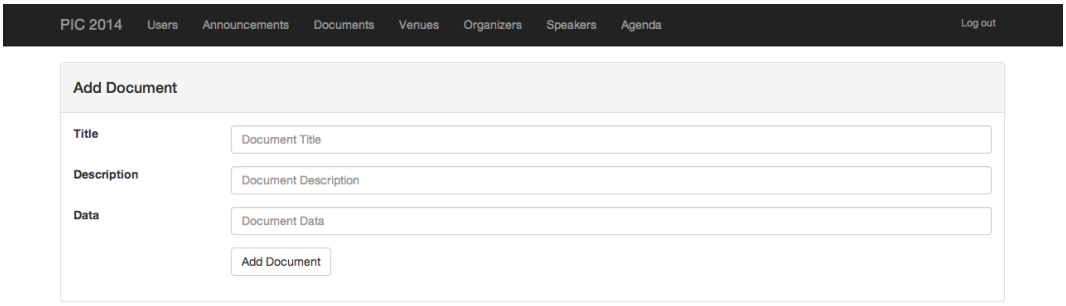


Figura B.17: Formulario para añadir nuevos *documentos*.

B.3.4. Lugares

Los lugares de interés para el congreso se añaden pulsando el botón + **Add Venue** en la sección de lugares e rellenando el siguiente formulario:

- Name: nombre del lugar.
- Details: detalles de interés del lugar.
- Location: localización del lugar. Este campo se puede rellenar de dos maneras, introduciendo las coordenadas geográficas (latitud y longitud) o introduciendo la dirección del lugar. Cuando se introducen las coordenadas o la dirección, se muestra en un mapa un marcador con la localización para poder comprobar que esta es correcta.

En la figura B.18 se muestra el formulario que permite añadir nuevos lugares al congreso.

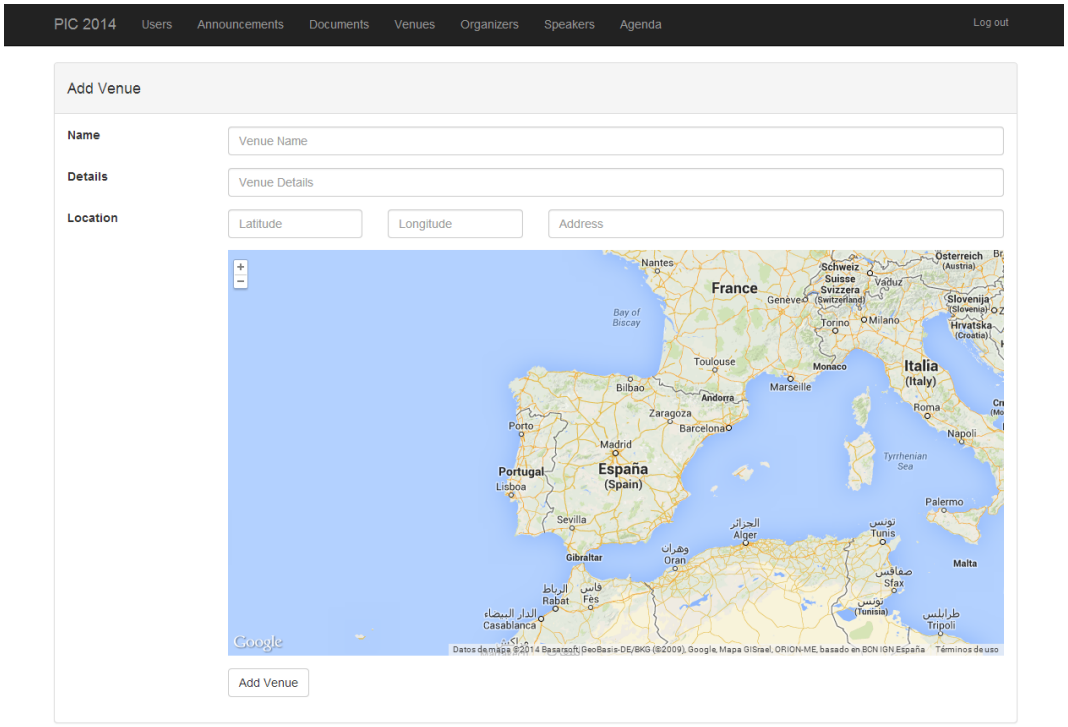


Figura B.18: Formulario para añadir nuevos *lugares*.

B.3.5. Organizadores

Para añadir a los organizadores del congreso hay que pulsar sobre el botón + **Add Organizer** en la sección de organizadores y completar la siguiente información:

- Name: nombre del organizador.
- Origin: procedencia del organizador.
- Details: detalles relevantes del organizador.

- Group: grupo al que pertenece el organizador (comité organizador, comité científico, etc).

En la figura B.19 se muestra el formulario que permite añadir nuevos organizadores al congreso.

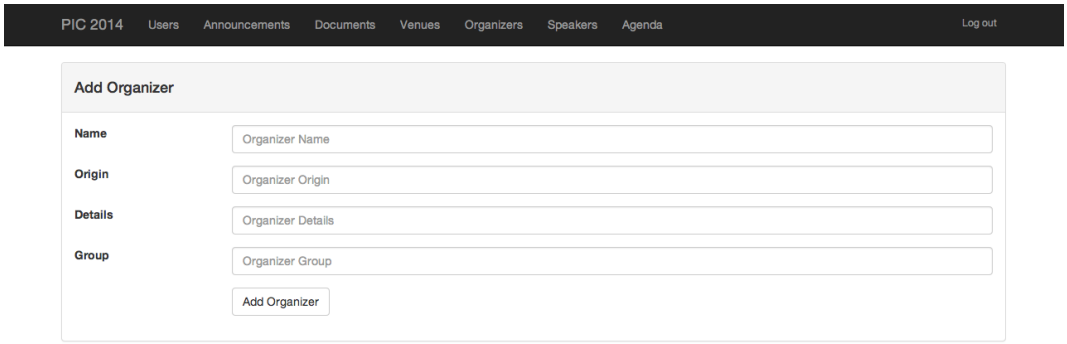


Figura B.19: Formulario para añadir nuevos *organizadores*.

B.3.6. Ponentes

Los ponentes que van a hablar en el congreso se añaden en la sección de pontentes, pulsando el botón + **Add Speaker** y rellenando el siguiente formulario:

- Name: nombre del ponente.
- Origin: procedencia del ponente.
- Charge: cargo del ponente.
- Description: breve biografía del ponente.
- Picture: foto del pontente. La foto tiene que tener un tamaño de 160x200 píxeles.

En la figura B.20 se muestra el formulario que permite añadir nuevos ponentes al congreso.

B.3.7. Agenda

Para añadir nuevos eventos en la agenda del congreso hay que pulsar el botón + **Add Event** en la sección de la agenda e introduciendo los siguientes campos en el formulario:

- Title: título del evento.

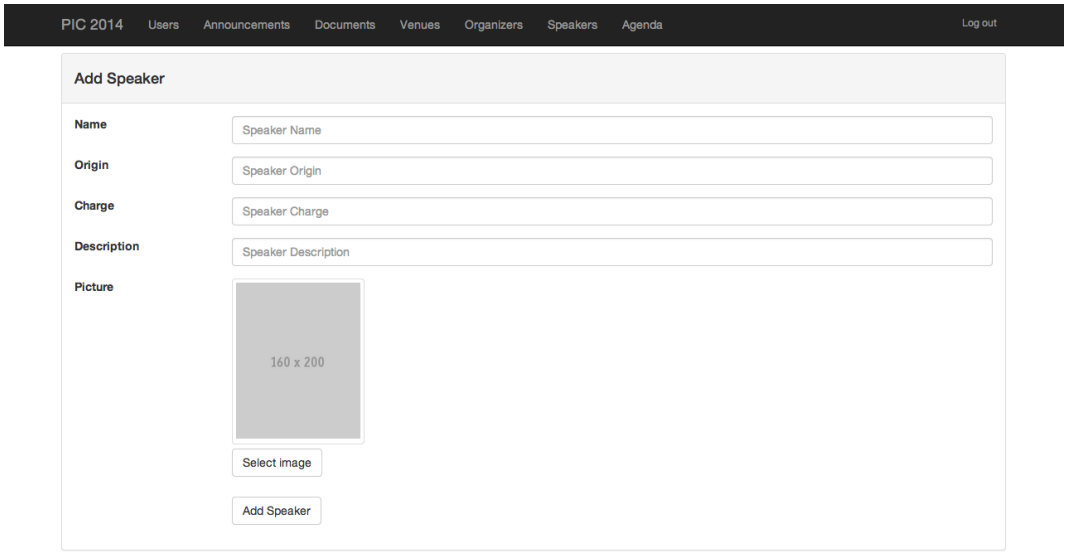


Figura B.20: Formulario para añadir nuevos *ponentes*.

- Description: breve descripción del evento.
- Day, month, year: día, mes y año para generar la fecha del evento.
- Hour, minutes: hora y minutos para generar la hora a la que se realizará el evento.

En la figura B.21 se muestra el formulario que permite añadir nuevos eventos en la agenda del congreso.

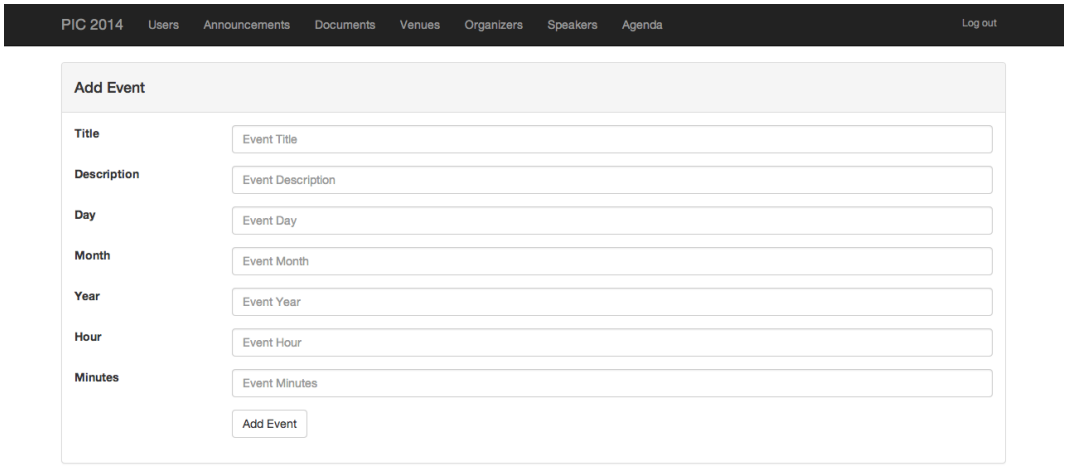


Figura B.21: Formulario para añadir nuevos *eventos* en la agenda.

Bibliografía

[1] ActionBarSherlock. http://actionbarsherlock.com/.

[2] AmazingListView. https://code.google.com/p/android-amazing-listview/.

[3] Android. http://developer.android.com/about/index.html.

[4] Aplicación de TedConnect en Google Play. https://play.google.com/store/apps/details?id=com.ted.android.conference&hl=es.

[5] Bootstrap project page. http://getbootstrap.com/.

[6] Distribución de versiones de Android. https://developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net.

[7] Eclipse ide. http://www.eclipse.org/.

[8] Express project page. http://expressjs.com/.

[9] Foreman github repository. http://github.com/ddollar/foreman.

[10] Git. http://git-scm.com/.

[11] GitHub. https://github.com/.

[12] Google Maps javascript api v3. https://developers.google.com/maps/documentation/javascript/.

[13] Guía de diseño para Android. https://developer.android.com/design/patterns/navigation-drawer.html.

[14] Heroku cloud application framework. http://heroku.com.

[15] Jade project page. http://jade-lang.com/.

[16] MongoDB. http://mongodb.org/.

[17] Mongoose project page. http://mongoosejs.com/.

[18] Node.js project page. http://nodejs.org/.

[19] Página web de las Jornadas de Gerencia Universitaria. http://www.ucm.es/jornadasgerencia.

[20] Página web del congreso PIC 2014. http://pic.sjtu.edu.cn/.

[21] Repositorio de la app de Android. https://github.com/MyConference/MyConference-Android.

[22] Restify project page. http://mcavage.me/node-restify/.

[23] Sass, systactically awesome stylesheets. http://sass-lang.com/.

[24] SQLite. http://sqlite.org/.

[25] Sublime Text. http://www.sublimetext.com/.

[26] Transloadit. https://transloadit.com/.

[27] Tutorial para incluir Google Maps en Android. https://developers.google.com/maps/documentation/android/start#installing_the_google_maps_android_v2_api.

[28] Tutorial para la autenticación en Android. http://udinic.wordpress.com/2013/04/24/write-your-own-android-authenticator/.

[29] Tutorial para la creación de la base de datos en Android. http://developer.android.com/guide/topics/providers/content-provider-creating.html.