Grado en Ingeniería Informática

Trabajo Final de Grado

Desarrollo de aplicaciones web híbridas Metodología y caso práctico

Autor:
Daniel Chía Aguilar

Supervisor:
Sergio Aguado González
Tutor académico:
Óscar Belmonte Fernández

Fecha de lectura: 26 de Junio de 2017
Curso académico 2016/2017

Resumen

Las tecnologías en el ámbito del desarrollo web han avanzado mucho en los últimos años, lo que ha propiciado la aparición y el auge de nuevos tipos de aplicación, como son las aplicaciones híbridas, aquellas que son posibles de compilar a dispositivos móviles y funcionar de la misma forma que una aplicación nativa.

Uno de los *frameworks* que ha cobrado más interés recientemente es Ionic, especialmente a partir de su versión 2, puesto que ha integrado Angular 2 y pueden usarse los conocimientos en Angular para desarrollar aplicaciones web híbridas de forma sencilla. Ionic proporciona toda una serie de componentes preparados para ser usados en entornos móviles y opera sobre Cordova para conseguir que aplicaciones web funcionen en dispositivos móviles.

En este documento se detallan los pasos para conseguir una aplicación de estas características con un proyecto real, tratando de emplear las tecnologías más recientes en el entorno web.

Palabras clave

Aplicación web híbrida, Ionic, Angular

Keywords

Hybrid web application, Ionic, Angular

Índice general

Índice de figuras

Capítulo 1

Introducción

Este capítulo es una introducción del proyecto al lector. Se detalla el contexto y la motivación que han llevado a la creación del proyecto, la empresa en la que se ha realizado y una breve introducción al concepto de aplicación híbrida. También se detallan los objetivos del proyecto y la estructura que sigue esta memoria.

1.1. Contexto y motivación del proyecto

En esta sección se presenta el proyecto, la empresa en la que se ha realizado y el concepto de aplicación híbrida, importante para comprenderlo. También se dan ciertas pinceladas sobre el contexto en el que se encuentran las aplicaciones híbridas.

1.1.1. El proyecto

El proyecto desarrollado consiste en una aplicación híbrida de gestión de eventos y congresos, compuesta por una parte servidor y una parte cliente, diseñada para ser funcional en sistemas *Android* 4.0 y superiores, y en sistemas *iOS* 9.0 y superiores. Se han escogido estas versiones al considerarse que el software funcional en dichas versiones es el que puede ser ejecutado en el mayor número de dispositivos. Escoger una versión inferior habría supuesto el sacrificio de funcionalidades de la aplicación, y una de las últimas versiones habría supuesto que estuviese disponible sólo en los terminales más modernos y caros.

En este sentido, se busca que el resultado obtenido sea lo más similar posible al desarrollo nativo de la misma aplicación en cada plataforma, si no idéntico, por lo menos a nivel visual. Siendo un caso práctico real, se busca probar los límites de estas tecnologías y comparar el mismo caso práctico con el desarrollo nativo en ambas plataformas.

El proyecto en sí trata de demostrar la viabilidad práctica de una serie de tecnologías muy recientes, como son las usadas para el desarrollo de aplicaciones híbridas, con el fin de poder desarrollar y aplicar un nuevo paradigma de desarrollo dentro de la empresa en la que se realiza.

1.1.2. La empresa

El proyecto cuya propuesta se presenta en este documento se ha desarrollado en la empresa *Soluciones Cuatroochenta*, empresa especializada en el desarrollo integral de aplicaciones para smartphones y tablets y programación a medida para mejorar procesos de trabajo [8].

Además del desarrollo de aplicaciones, también experimenta en sectores como la realidad virtual, el desarrollo de videojuegos y centra parte de sus recursos en estos sectores y en la investigación de nuevas tecnologías y procesos que se puedan aplicar al resto de la empresa, o mejorar los ya existentes.

1.1.3. Aplicaciones híbridas

1.1.3.1. En qué consiste una aplicación híbrida

Una aplicación híbrida es aquella que ha sido desarrollada empleando tecnologías web, para después ser portada a sistemas operativos móviles o de escritorio, pudiendo funcionar en cualquiera de estas plataformas con un rendimiento superior al que tendría únicamente como aplicación web. Antes de alcanzarse este modelo se tenían aplicaciones web que podría decirse que "simulaban"por su apariencia ser nativas, o que como mínimo estaban adaptadas para el uso en móviles (aplicaciones *responsive*), pero en ningún caso podían instalarse.

Este tipo de aplicaciones supera, además, otros inconvenientes como la dependencia de internet para su uso, o la necesidad de desarrollar la misma aplicación más de una vez si se quiere hacer multiplataforma para móviles, siendo necesario en este caso una versión para *iOS*, otra para *Android* y otra para *Windows Phone* si se quiere tenerla en las tres plataformas.

Se podría decir que una aplicación híbrida combina las virtudes del desarrollo en plataformas nativas con las virtudes del desarrollo web. En la figura 1.1 puede verse un gráfico que compara las tres plataformas.

Características de la App	App Nativa	App Web	App Híbrida
Gráficos	API Nativa	HTML, Canvas, SVG	HTML, Canvas, SVG
Rendimiento	Muy rápido	Lento	Rápido
Look&Feel	Nativo	Emulado	Emulado
Distribución	App Store/Google Play	Web	App Store/Google Play
Coste	Medio-Alto, Alto	Bajo	Medio
Interfaz de Usuario	Muy buena	Muy mala	Buena
Tiempo de Desarrollo	Alto	Bajo	Medio
Funcionalidades del Teléfono			
Cámara	Si	No	Si
Notificaciones	Si	No	Si
Contactos	Si	No	Si
Calendario	Si	No	Si
Almacenamiento Offline	Almacenamiento seguro de ficheros	Solo Cache, IndexedDb y SQL compartida	Almacenamiento seguro de ficheros y SQL compartida
Geolocalización	Si	Si	Si
Gestos			
Swipe	Si	Si	Si
Pinch	Si	No	Si
Spread	Si	No	Si
Contectividad	Online y Offline	Mayormente Online	Online y Offline
Lenguaje	Objective C y Java	HTML, CSS, Javascript	HTML, CSS, Javascript

Figura 1.1: Comparativa de los tipos de aplicación. Fuente [1]

1.1.3.2. Contexto actual de las aplicaciones híbridas

En los últimos años ha habido un gran desarrollo tecnológico en el contexto del desarrollo web, tanto en *frontend* (parte cliente), con *frameworks* como *Angular* [9], *React* [10], *Ember* [11], entre otros, como en *backend* (parte servidor), fundamentalmente con *NodeJS* [12] y con toda la multitud de *frameworks* que han surgido para PHP [13], como son Symfony [14] y Laravel [15], entre muchos otros.

En especial el avance en el lado del *frontend*, junto a otras tecnologías como *Cordova* [16], han facilitado el avance y la aparición de *frameworks* enfocados al desarrollo de aplicaciones híbridas a lo largo de todos estos años, como son *PhoneGap* [17] o *SenchaTouch* [18], pero el que sin duda está adquiriendo más popularidad y desarrollo, especialmente desde la salida de *Angular 2*, es *Ionic* [19], más concretamente a partir de su versión *Ionic 2*.

La reciente irrupción de *Ionic* se debe a la reconstrucción que se hizo del *framework* para adoptar y utilizar *Angular 2*, lo que permite aprovechar los conocimientos en esta tecnología y hacer

posible que, con un esfuerzo relativamente pequeño, proyectos existentes en *Angular 2* puedan ser adaptados en versiones móviles.

Como fuentes de su éxito, también cabe destacar la comunidad y el equipo de desarrolladores que tiene detrás, bastante activo.

1.2. Objetivos del proyecto

El objetivo principal del proyecto consiste en el desarrollo de la aplicación web híbrida propuesta, para la gestión de eventos y congresos, y asegurar su funcionamiento en sistemas *Android* 4.0 y superiores, así como en sistemas *iOS* 9.0 y superiores.

Este objetivo principal puede desglosarse en el siguiente listado de objetivos parciales:

Parte cliente

- Diseño y desarrollo de los componentes gráficos de la aplicación.
- Desarrollo de los modelos de datos usados en la parte cliente.
- Desarrollo de una capa de servicios en la parte cliente.
- Diseño y desarrollo de las pantallas de la aplicación.
- Diseño y desarrollo de la infraestructura de persistencia de datos en la parte cliente.
- Aseguramiento de calidad de la parte cliente.
- Despliegue de la parte cliente.

Parte servidor

- Desarrollo de la estructura base del servidor.
- Desarrollo de los servicios web.
- Diseño y desarrollo de la infraestructura de persistencia de datos en la parte servidor.
- Aseguramiento de calidad de la parte servidor.
- Despliegue de la parte servidor.
- Aseguramiento de calidad de la aplicación completa, incluyendo integración y aseguramiento de calidad a nivel de usuario (e2e [20]).

Cabe destacar que el desarrollo de la parte servidor, su despliegue, y el aseguramiento de calidad de la parte servidor son necesarios para el funcionamiento de la aplicación completa, pero quedan en un segundo plano.

1.3. Estructura de la memoria

En el capítulo 2 se presenta una descripción más amplia del proyecto, de la funcionalidad de la aplicación, el alcance que tiene y las restricciones principales que se han encontrado en el proyecto.

El capítulo 3 muestra la metodología que se ha usado, explicando por qué se ha escogido esa metodología y su funcionamiento, y la planificación y el seguimiento que se ha realizado del proyecto.

En el capítulo 4, se detalla el análisis del sistema que se ha realizado, la arquitectura final del sistema, y el diseño de la interfaz que hay detrás.

En el capítulo 5 se detalla cómo ha sido la implementación del proyecto, paso a paso, y los mecanismos de verificación y validación que se han empleado a lo largo del mismo.

Por último, en el capítulo 6 se detallan las conclusiones, tanto a nivel técnico, como a nivel personal, y se trata de dar unas directrices generales aplicables a cualquier proyecto de las características del que se ha presentado, tomando el proyecto como un ejemplo práctico en el que se pueden seguir dichas directrices.

Capítulo 2

Descripción del proyecto

Este capítulo describe en qué consiste el proyecto y las funcionalidades que este proporciona, también se exponen las ventajas que se obtienen con su consecución, la situación inicial desde la que se parte, el alcance que tiene y las restricciones que ha sufrido. Se concluye el capítulo con un pequeño listado de las tecnologías que lo conforman, que son expuestas con más detalle en el capítulo 4.

2.1. El proyecto

Como se ha introducido en el capítulo 1, el proyecto consiste en el análisis, diseño e implementación de una una aplicación híbrida de gestión de eventos y congresos, formada por una parte cliente y de una parte servidor, cuyo funcionamiento está asegurado en sistemas *Android* 4.0 y superiores así como en sistemas *iOS* 9.0 y superiores.

Se ha dado preferencia al desarrollo de la parte cliente frente a la parte servidor puesto que el proyecto surge por la decisión que Soluciones Cuatroochenta tomó: demostrar la viabilidad de *Ionic* en un caso real, con el fin de incorporar y utilizar esta tecnología en la empresa.

2.1.1. Funcionalidad

La aplicación permite publicar toda la información referente a un congreso, dividida y localizada en las secciones correspondientes y hacerla accesible a todos los usuarios que la descarguen, distinguiendo entre usuarios registrados y no registrados, permitiéndoles navegar por dichas secciones y consultar la información.

La información que se muestra a los usuarios no registrados consiste en:

- El cartel del congreso, en la pantalla principal.
- Los eventos que se realizan y la información de los mismos.

- La información de inscripción.
- Los ponentes que asisten e información sobre ellos.
- Documentación pública del congreso, como puede ser la programación.
- Un listado con las localizaciones de interés relacionadas con el congreso.
- Un listado con los servicios que se ofertan dentro del mismo, como hoteles.
- Información de los patrocinadores.
- Información legal del congreso.

De haber algún cambio en el congreso, el usuario podrá verlo en una sección dedicada a ello en forma de notificación. Se distingue entre notificaciones públicas, que pueden ver todos los usuarios, incluídos los no registrados y notificaciones privadas, propias de cada usuario.

La aplicación también gestiona las inscripciones al congreso, pidiendo y gestionando los datos necesarios para el registro, pudiendo enlazarse a una plataforma de pagos en el futuro y generando un usuario y una acreditación.

El usuario podrá autentificarse en la aplicación y esta le ofrecerá acciones específicas para ese usuario, como:

- La edición de su perfil.
- La opción de activar una funcionalidad que le permita crear una red de contactos con los asistentes al congreso que hayan activado esta opción.
- Acceder y consular los datos de su acreditación.
- La opción de emplear una funcionalidad de realidad aumentada e interactuar con objetos del congreso que ofrezcan esta funcionalidad.

También deberá poder acceder a toda la información a la que tiene acceso un usuario no registrado, a excepción de la opción de inscripción, puesto que ya la ha realizado.

2.1.2. Ventajas que se obtienen

Una aplicación con estas funcionalidades permite:

1. Ahorrar tiempo en la gestión de los congresos a los organizadores de los mismos.
2. Centralizar la gestión de dichos congresos en un único punto.
3. Facilitar a los asistentes la información de los congresos y la inscripción a los mismos.
4. Ofrecer información en tiempo real del congreso, por ejemplo, un cambio en la planificación de algún evento.

5. Ofrecer información adicional de los patrocinadores del congreso y de la localización de los servicios cercanos al congreso, como hoteles, por ejemplo, entre otro tipo de información.
6. Permitir la comunicación entre los asistentes del congreso y los ponentes mediante un canal de comunicación destinado para ello, basado en la realización de preguntas y respuestas.
7. Ayudar a los asistentes a los congresos a crear una red de contactos (*networking*) entre los propios asistentes del congreso que lo deseen.
8. Realizar sugerencias a los organizadores del congreso.

2.2. Situación inicial

El desarrollo del proyecto parte de cero. No obstante, se cuenta desde el inicio del mismo con un diseño preliminar de las interfaces, creadas con el programa *Sketch* [21] lo que ahorra el proceso de creación de *mockups*, puesto que ya se proporcionan de antemano, es el diseño que desea el cliente y está bien formulado, es decir, no existen inconsistencias en el mismo. También se proporcionan los iconos usados en este diseño preliminar, puesto que pueden obtenerse del fichero original.

2.3. Alcance

El alcance de la aplicación propuesta incluye la creación de la parte cliente de la aplicación descrita que ha de ofrecer toda la funcionalidad expuesta. Dentro de la parte cliente se incluye el diseño de la aplicación, la programación y el resto de elementos necesarios que la hagan funcionar.

También se incluye la creación de una parte servidor que ofrezca los servicios web que la parte cliente necesite, pero muy reducida, ofreciendo únicamente los servicios web necesarios y de manera concisa, para centrar los esfuerzos en la parte cliente.

No se incluye la creación de la sección de administración (*backoffice*) de los congresos por parte de los organizadores. Se incluye únicamente la parte cliente, la que usarán los usuarios finales.

A nivel:

- **Organizativo**: La aplicación final será utilizada por los organizadores del congreso para gestionar la información que se muestra, por los usuarios del congreso para acceder a la información del mismo y gestionar sus propios datos, y mantenida por personal especializado.
- **Funcional**: Ha de cubrir toda la funcionalidad necesaria para la gestión de los congresos, desde el acceso a la información a los registros, y también deberá poder ser usada para facilitar la gestión de la entrada al congreso, mostrando la acreditación.
- **Informático**: Es un sistema nuevo e independiente, por lo que no se sustituye ni se cambia ninguno de los existentes.

2.4. Restricciones

Fundamentalmente se tiene, a nivel:

- **Temporal**: Se tiene un plazo fijo de 300h para realizar el proyecto, y no puede ser prorrogado, por lo que cualquier retraso en la planificación puede llegar a ser un inconveniente serio.
- **De RRHH**: El proyecto lo lleva una única persona, puesto que se trata de un proyecto con fines exploratorios para el que no es necesario más que una persona.
- **Económico**: Al ser un proyecto realizado en la estancia en prácticas, no se disponen de recursos económicos presupuestados para realizarlo.

2.5. Tecnologías empleadas

Se distingue entre las tecnologías compartidas por todo el proyecto, las utilizadas en la parte cliente y las usadas en la parte servidor. A continuación se proporciona un listado a modo de resumen, este listado se expone con detalle en el capítulo 4.

2.5.0.1. Compartidas en todo el proyecto

- TypeScript
- WebPack

2.5.0.2. De la parte cliente

- HTML5
- SASS
- Angular
- Ionic
- NgRx
- Redux
- RxJS
- Apache Cordova

2.5.0.3. De la parte servidor

- NodeJS
- ExpressJS
- Body-parser
- Cors
- Mongoose
- MongoDB
- Winston

2.5.0.4. Aseguramiento de calidad

- Jasmine
- Karma
- Codecov

Capítulo 3

Detalle de las tecnologías empleadas

En este capítulo se detallan todas las tecnologías que se han empleado a lo largo del proyecto, especificando el ámbito en el que son empleadas, la función que realizan y, de existir más opciones, por qué se ha optado por esa y no por otras tecnologías similares.

3.1. Lenguajes

En esta sección se detallan los lenguajes empleados en todo el proyecto, las características de cada uno y, de haberlas, la versión empleada en cada caso.

3.1.1. HTML5

HTML [22] *(HyperText Markup Language)* en su versión 5 es uno de los lenguajes claves en el entorno web y el lenguaje principal utilizado para el diseño de la estructura de las vistas en Ionic. Es utilizado por Angular, que a su vez es usado por Ionic.

Su uso se limita a la creación de la estructura de los componentes, cabe destacar que, en el diseño de componentes, únicamente se incluye el contenido que va dentro de la etiqueta *body* de un documento HTML, a excepción del documento HTML principal, encargado de cargar los *scripts* de Angular, que es un documento HTML completo.

3.1.2. JavaScript

JavaScript [23] es, a día de hoy, junto a HTML y CSS [24], uno de los lenguajes que mueve la web, y ha cobrado especial importancia en los últimos años. Una prueba de ello puede verse en githut [2], que es una web en la que se muestra la actividad de GitHub [25] clasificada por lenguajes. Una gráfica puede verse en la figura 3.1.

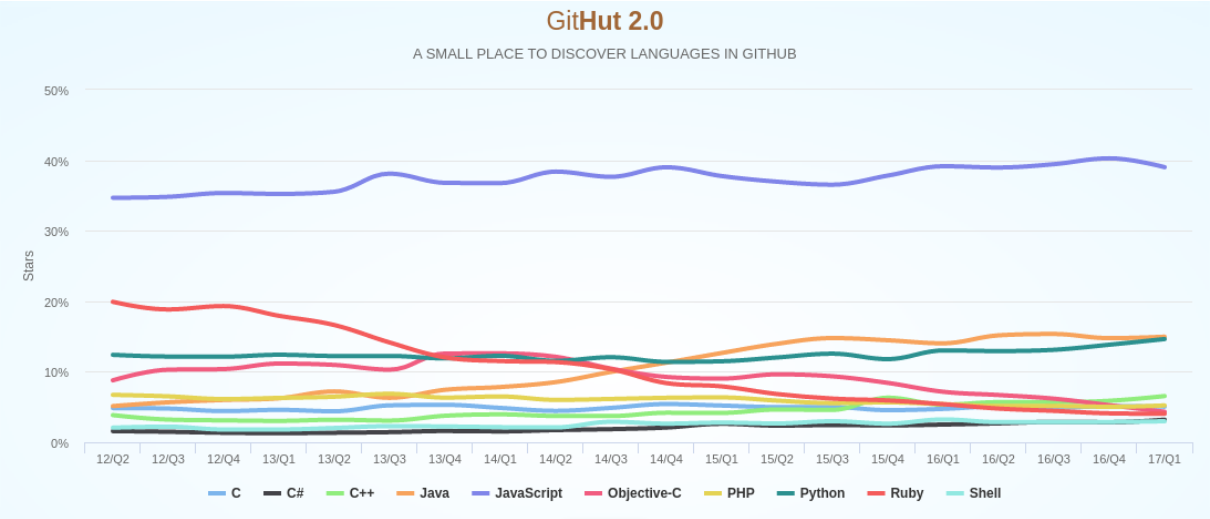

Figura 3.1: Uso de lenguajes en GitHub. Fuente [2]

El código del proyecto es escrito en TypeScript, y posteriormente *traspilado* a código JavaScript, que es el código que es ejecutado en un navegador o compilado por Ionic para obtener una aplicación móvil. El código final del proyecto, el código JavaScript, se tiene en ECMAScript 5 [26].

Esta traspilación es un proceso similar al de la compilación, en el que la diferencia fundamental radica en el resultado, puesto que no se obtiene código máquina o un ejecutable, sino que se obtiene código en otro lenguaje, en este caso JavaScript, con la misma funcionalidad.

3.1.3. TypeScript

Superset de JavaScript, TypeScript [27] amplía las opciones del lenguaje, añadiendo tipado entre otras muchas funcionalidades. Esto es especialmente útil cuando el proyecto alcanza un tamaño considerable.

Todo el proyecto está escrito utilizando TypeScript, empleando la sintaxis equivalente a ECMAScript 6 en JavaScript, incluyendo el tipado y otras funcionalidades que aporta TypeScript. TypeScript es *traspilado* a JavaScript, utilizando su propio compilador, obteniendo así el código funcional que será usado finalmente.

En el proyecto se ha usado la versión 2.2.1.

3.1.4. SASS

SASS [28] *(Syntactically Awesone StyleSheets)* es una extensión a CSS que añade soporte a variables, anidamiento, *mixins* (similares a los métodos) y herencia, entre otras funcionalidades. Al igual que ocurre con TypeScript, las hojas de estilo escritas en SASS son compiladas a CSS.

Ionic 2 lo incluye por defecto, y también lo trata por defecto, al construir el proyecto, por lo que en este sentido su incorporación y su uso no ha requerido ningún esfuerzo extra.

3.2. Tecnologías compartidas

En esta sección se introducen las tecnologías que son utilizadas tanto en la parte cliente como en la parte servidor.

3.2.1. Webpack

Este componente de software se encarga de empaquetar todos los ficheros JavaScript (o TypeScript), agrupándolos en uno o más módulos, a partir de los que se generan uno o más ficheros, agrupando todos los componentes necesarios y sus dependencias en el orden correcto para que el resultado sea funcional.

Esto tiene diversos propósitos, pero en general sirve para poder cargar la aplicación con todas sus dependencias a partir de un único script (o varios, si hay varios módulos), que esté funcionando bien, que tenga únicamente las dependencias que necesita, y que sea construido automáticamente, por lo que, al añadir nuevas dependencias, no tengamos que realizar esfuerzo extra.

Webpack [29] es lo que se conoce en inglés como un *module bundler*. Otra opción a Webpack es Browserify [30], en este sentido se ha optado por Webpack principalmente porque ya viene incluído y configurado en Angular e Ionic, por lo que su uso no ha requerido ningún esfuerzo. En la parte servidor se ha usado principalmente en el entorno de testing, puesto que, en el caso del servidor, se ha usado directamente el código funcional en una consola de NodeJS.

3.3. Parte cliente

En esta sección se detallan todas las tecnologías que se han empleado en la parte cliente del proyecto, tanto en la creación de las vistas como en las necesarias para el funcionamiento interno de esta parte del proyecto.

3.3.1. Angular

La base de la parte cliente, que está escrita utilizando Angular [9] como base, permite crear y hacer funcionar toda la parte gráfica en un entorno web.

Crea lo que se conoce como SPA [31] *(Single Page Application)*, que ofrecen la ventaja de ser cargadas y usadas por el cliente, pudiendo reducir el número de llamadas al servidor. Sigue un modelo de diseño por componentes, lo que lo hace altamente modular (los componentes son fácilmente reutilizables y agrupables) y escalable (es fácil organizar los componentes en módulos).

A partir de la versión 2 incorpora y emplea TypeScript y WebPack.

Otra opción habría sido el uso de ReactJS [10]. En este sentido no había elección posible, puesto que Ionic trabaja sobre Angular, escoger Ionic implica trabajar sobre Angular.

En el proyecto se ha usado la versión 2 en primera instancia y se ha realizado una actualización a la versión 4.

3.3.2. Ionic

Ionic [19] es la tecnología clave en este proyecto. Trabaja sobre Angular y se encarga de hacer el renderizado nativo en los dispositivos móviles, a partir del código de una aplicación web, utilizando los componentes que Ionic aporta, diseñados para ello, escritos en Angular 2, lo que lo hace muy fácil de usar.

Esto hace que, al ser compilada con las herramientas que también aporta Ionic, una aplicación web, que puede estar funcionando como una página web en un entorno de escritorio, pueda ser instalada como una aplicación móvil y se vea con el estilo propio del sistema operativo del móvil, teniendo así una aplicación híbrida.

Funciona en conjunto con Cordova, que es el componente software encargado de realizar la compilación a sistemas móviles.

Otra alternativa a Ionic podría haber sido React Native [32], que emplea React como base.

En el proyecto se ha utilizado la versión 2 en las fases iniciales del proyecto y se ha actualizado a la versión 3.

3.3.3. Apache Cordova

Apache Cordova [16] permite compilar una aplicación web (código HTML, CSS y JavaScript) y generar una aplicación móvil, permitiendo realizar aplicaciones híbridas, es la tecnología empleada por Ionic encargada de compilar el proyecto para obtener un archivo ejecutable instalable en un sistema operativo Android, iOS, Windows Phone, Blackberry, FirefoxOS, Ubuntu, entre otros.

Además, proporciona una multitud de *plugins* que permiten interactuar con los componentes físicos del teléfono, por ejemplo, la cámara, algo imposible de hacer únicamente con una aplicación web. Estos *plugins* se distribuyen en una especie de *market*, lo que implica que son de libre acceso y que los usuarios pueden crear y subir su propios *plugins* que realicen acciones concretas.

3.3.4. Redux

Redux [33] es un componente software que permite centralizar el estado de la aplicación en un único punto, llamado la *store*, y hacer que esta *store* sólo sea modificada por el código localizado en los llamados *reductores*, que responden a *acciones* y tienen en cuenta el estado anterior del estado antes de modificarla.

Todo esto hace que el estado de la aplicación sea mucho más fácil de gestionar, puesto que está centralizado, puede ser accedido desde cualquier punto de la aplicación y únicamente puede ser modificado por los *reductores* que responden ante *acciones* concretas y sólo ante esas *acciones*.

Las *acciones*, al igual que los *reductores*, también son definidas y pueden aportar datos que el *reductor* utilice.

El estado de la aplicación puede repartirse entre los diversos *reductores*, consiguiendo así un estado modular, cada *reductor* puede aportar nuevas variables al estado, sin apenas tener que realizar cambios en el código.

En el proyecto se ha empleado la versión 3.6.0.

3.3.5. RxJS

RxJS [34] es una librería de *programación reactiva* [35] [36] para JavaScript y TypeScript que implementa este paradigma y aporta y permite utilizar objetos como el *Observable*.

La gestión de llamadas asíncronas puede ser todo un quebradero de cabeza, especialmente cuando se anidan, (el llamado *callback hell*), y no es extraño que se de este caso cuando el uso de llamadas asíncronas alcanza cierta complejidad. Esta librería ayuda enormemente a evitar estos casos proporcionando herramientas con las que gestionar la asincronía, tratando la información como flujos de datos, que pueden variar con el tiempo y proporcionando operadores para trabajar con el flujo en sí.

En el proyecto se ha empleado la versión 5.1.1.

3.3.6. NgRx

NgRx [37] es un componente software formado por una serie de módulos independientes pensado para ser usado en proyectos Angular.

Hace posible utilizar la *store* de Redux en proyectos Angular, en combinación con RxJS instalando los dos módulos en cuestión. También ofrece más módulos con más funcionalidades como, por ejemplo, el de una base de datos indexada que emplea RxJS, pero en el proyecto sólo se han usado esos dos.

En el proyecto se ha empleado la versión 1.2.0 en el *core*.

3.4. Parte servidor

En esta sección se detallan las tecnologías empleadas para construir y poner en marcha el servidor. También se incluyen y se detallan las librerías o componentes tecnológicos más importantes que se emplean dentro del mismo y que son fundamentales para que de el servicio que da.

3.4.1. NodeJS

La base del servidor, NodeJS [12], utiliza el motor de JavaScript V8 [38] de Chrome, por lo que funciona y es programado en JavaScript. Tiene un diseño modular, lo que significa que prácticamente todas las funciones que ofrezca han de ser añadidas como módulos, y son usadas por este núcleo. Esto lo hace liviano y eficiente, puesto que únicamente tiene y utiliza lo que necesita.

En el proyecto se ha empleado la versión 7.10.0.

3.4.2. ExpressJS

ExpressJS [39] es un módulo para NodeJS que ofrece las funcionalidades de un servidor HTTP y que funciona sobre el propio módulo HTTP de NodeJS, aportando muchas funcionalidades ya construidas y probadas sobre este módulo para no tener que preocuparnos constantemente de detalles de bajo nivel que pueden omitirse.

Esto permite construir y ofrecer los servicios web con una sintaxis muy sucinta, de manera muy rápida y sin perder rendimiento en ningún momento, por lo que no es de extrañar que a día de hoy la inmensa mayoría de servidores HTTP construidos con NodeJS usen ExpressJS.

En el proyecto se ha empleado la versión 4.14.1.

3.4.3. Body-parser

Body-parser [40] es una librería utilizada dentro del servidor, concretamente en el código encargado de dar respuesta a las peticiones realizadas desde la parte cliente.

Es lo que en NodeJS se conoce como *middleware*, su cometido es el de transformar, de forma automática, los objetos *Request*, que contienen los datos de la petición hecha por el cliente, y ofrecer el contenido del *body* de dicha petición como un objeto JSON, lo que hace mucho más fácil su uso y manipulación.

En este sentido actúa como una librería de utilidad, el servidor podría funcionar sin su uso, no obstante, su uso dentro del servidor es muy extendido y relevante.

En el proyecto se ha empleado la versión 1.17.1.

3.4.4. Cors

Cors [41] se trata también de otra librería, y también actúa como *middleware*, en este caso, para aplicar las cabeceras CORS [42] (Cross Origin Resource Sharing) de forma automática en todas las respuestas.

Es también otra utilidad, no necesaria para el funcionamiento del servidor, pero muy usada.

En el proyecto se ha empleado la versión 2.8.3.

3.4.5. Mongoose

Mongoose [43] es una librería que permite al servidor interactuar con la base de datos.

Es una pieza muy importante dentro del servidor, puesto que permite transformar métodos concretos usados dentro del servidor en acciones concretas dentro de la base de datos, obtener la respuesta de haberla si la acción se ha podido ejecutar correctamente, o un código de error si ha fallado, y transformar esta información en un objeto dentro del servidor con el que poder interactuar. Actúa, por tanto, como nexo de unión entre el servidor y la base de datos.

En el proyecto se ha empleado la versión 4.9.2.

3.4.6. MongoDB

Mongo [44] es la base de datos usada en el proyecto. Es la implementación de una base de datos noSQL basada en documentos (base de datos documental [45]), con documentos con una estructura JSON.

Se ha optado por esta base de datos por la fácil integración que tiene con NodeJS y por estar muy extendida, lo que implica que hay más documentación accesible, lo que es beneficioso para usuarios noveles en este tipo de tecnologías, como es el caso.

Otra opción habría sido emplear una base de datos SQL como *PostgreSQL*, con la que el autor de este documento tiene más experiencia, u otra implementación de una base de datos noSQL, como puede ser CouchDB [46], que es también una base de datos noSQL basada en documentos, o bien otra base de datos de otro tipo, como es Apache Cassandra [47], por ejemplo, de clave/valor.

Finalmente se ha optado por emplear mLab [48], que es un servicio externo que ofrece una base de datos Mongo por lo que, en este sentido no ha sido necesario preocuparse por la instalación.

3.4.7. Winston

Winston [49] es una librería que actúa como *logger*. Tiene la particularidad de ser asíncrona y poder operar sobre un número enorme de salidas, que incluyen bases de datos de prácticamente cualquier tipo, peticiones HTTP, ficheros e incluso servicios de Amazon, entre otros, y de ser altamente configurable.

Dentro del proyecto se ha empleado con la salida a consola, para mostrar los datos de cada petición realizada al servidor, algo que no es fundamental e imprescindible para su funcionamiento pero que sin duda ayuda mucho y que, con poca configuración más, podría ser de gran utilidad a la hora de detectar errores o poder obtener datos para, por ejemplo, realizar estadísticas de uso.

En el proyecto se ha empleado la versión 2.3.1.

3.5. Aseguramiento de calidad

En esta sección se introducen todos los componentes software empleados para asegurar la calidad del proyecto tanto en la parte cliente como en la parte servidor y para probar y tener una prueba empírica de que el código hace lo que queremos que haga.

3.5.1. Jasmine

Jasmine [50] es una librería que permite escribir tests unitarios en JavaScript. Muy similar a JUnit, que podría considerarse su equivalente en entornos Java.

En el proyecto se ha empleado la versión 2.5.2.

3.5.2. Karma

Karma [51] es lo que se conoce como un *test runner*. Permite ejecutar baterías de tests, además de automatizar la ejecución de los mismos, entre otras opciones, lo que permite aplicar mucho más fácilmente metodologías *TDD*. Es altamente configurable, lo que lo hace adaptable a un gran número de proyectos y entornos.

Ofrece el resultado de la ejecución de los tests por consola y mediante un navegador, en el que se muestra un informe en HTML.

Cuenta con su propia CLI (*Command Line Interface*) y cuenta con un gran número de *add-ons* propios que se pueden añadir para, por ejemplo, cambiar el formato de la salida que se utiliza, o cambiar de tipo de navegador, por ejemplo de *Chrome* a uno de tipo consola, como *PhantomJS*, entre muchos otros.

En el proyecto se ha realizado esto mismo, se ha cambiado a un navegador en consola para poder ejecutar la batería de tests en sistemas de integración continua.

Se ha empleado la versión 1.6.0.

3.5.3. Codecov

Codecov [52] es un componente software, integrado dentro de Angular, pero que puede ser empleado por separado, y que permite realizar un análisis de cobertura del código realizado por los tests.

Genera un informe muy detallado en el que no sólo detalla el porcentaje de código cubierto en cada archivo del proyecto, sino que además detalla qué líneas son cubiertas por los tests y cuales no, lo que es especialmente útil para encontrar errores que se hayan podido cometer en los tests, o condiciones o circunstancias que no se habían tenido en cuenta durante el diseño de los mismos.

Además es fácilmente usable e integrable en sistemas de integración continua, desde un proyecto Angular basta con añadir una opción al comando de test para generar un informe, y desde

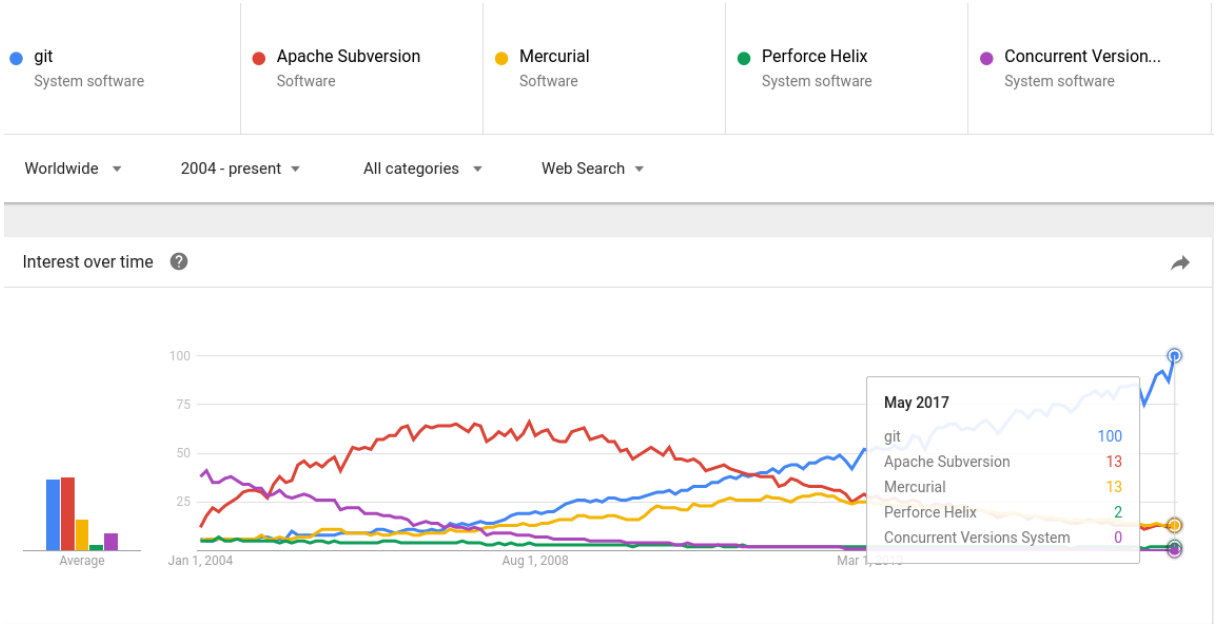

Figura 3.2: Búsquedas de Git en Google en los últimos años. Fuente [3]

un sistema de integración continua puede controlarse, por ejemplo, el porcentaje mínimo de cobertura de código que debe existir para integrar o no la subida de código, que puede ser especificado fácilmente.

En el proyecto se ha empleado la versión 2.1.0.

3.6. Control de versiones

En esta sección se habla del sistema de control de versiones que se emplea en el proyecto, por qué se ha escogido el sistema de control de versiones en cuestión y qué ofrece.

3.6.1. Git

Git [53] es un software libre de control de versiones muy extendido y usado a día de hoy. Esto puede verse reflejado en una comparación de las búsquedas realizadas en Google en los últimos años, en la figura 3.2, y un gráfico por países que muestran el término más buscado en cada uno, en la figura 3.3.

Figura 3.3: Término más buscado en cada país. Fuente [3]

Una de las principales características de Git es que es un sistema de control de versiones distribuido, lo que implica que existen múltiples copias y que el repositorio es compartido. Además, es gratuíto y de código abierto, lo que ha propiciado que sea usado en muchos servicios de control de versiones, como son GitHub, Bitbucket, GitLab, etc...

3.6.2. Bitbucket

Bitbucket [54] es una plataforma de desarrollo que emplea Git y que aporta una interfaz para trabajar con los repositorios.

Podía haberse optado por el uso de GitHub [25] o GitLab [55], por ejemplo, en lugar de Bitbucket. Se ha usado Bitbucket porque es la plataforma que se estaba usando en la empresa.

3.7. Integración continua

En esta sección se introduce el sistema de integración continua (CI) que se ha usado en el proyecto. Un sistema de integración continua [4] permite la construcción automática del proyecto y, normalmente, la ejecución de diversos comandos, como puede ser el usado para la ejecución de los tests o el usado para realizar el despliegue continuo.

3.7.1. Bitbucket pipelines

Bitbucket pipelines [56] es un sistema de integración continua integrado en el propio Bitbucket, lo que facilita mucho su uso, puesto que puede ser configurado para ejecutarse al subir nuevos cambios en el repositorio y no es necesario instalar nada o integrar un servicio externo.

Su funcionamiento, en la práctica, consiste en la ejecución de una serie de órdenes en consola en una máquina, normalmente remota, que tiene los últimos cambios subidos. Estas órdenes pueden ser definidas de diversas formas, según el tipo de sistema de integración continua, en los

modernos suelen usarse ficheros con la extensión *.yml* [57] [58].

En este caso Bitbucket pipelines requería un fichero con esta extensión para funcionar, en el que se definen dichas órdenes, normalmente, la construcción del proyecto y la ejecución de los test. Si una de estas órdenes no se ejecuta correctamente (por ejemplo, fallan los test), no se ejecutará la siguiente (por ejemplo, despliegue del proyecto). También puede especificarse el tipo de imagen de la máquina Linux que ejecute las órdenes, en este caso, una con NodeJS para ejecutar el proyecto.

Otra característica muy beneficiosa, especialmente cuando se trabaja en grupo, es la posibilidad del envío de mensajes a Slack [59] cuando la ejecución de la integración continua ha finalizado, cuando va a empezar o cuando está ejecutándose. Se ha realizado esta integración en el proyecto y ha sido muy beneficiosa.

Había muchas opciones disponibles en lugar de Bitbucket pipelines, como son Travis CI [60], Circle CI [61], Codeship [62], Semaphore CI [63], Jenkins [64], Solano CI [65]... La mayoría son de pago y bastante similares, Jenkins es software libre pero ha de ser instalado y desplegado, al final se optó por el uso de Bitbucket pipelines porque venía ya integrada con Bitbucket y su plan gratuito encajaba muy bien con el uso que se le iba a dar en el proyecto.

3.8. Despliegue

En esta sección se detalla el mecanismo de despliegue, es decir, puesta en producción, que se ha empleado en el proyecto y por qué.

3.8.1. Heroku

Heroku [66] es una plataforma en la nube que ofrece servicios y herramientas para el despliegue de aplicaciones e integraciones para realizar un despliegue continuo. Su uso simplifica mucho el despliegue, ya que no hay que preocuparse por el control de la infraestructura y ofrecen muchas herramientas que permiten, por ejemplo, escalar los recursos de la aplicación, entre otras muchas cosas.

Lo especialmente interesante para el proyecto, es que ofrece un servicio gratuito con características más que suficientes para el despliegue del proyecto.

También tiene integración con Slack, que funciona exactamente igual que Bitbucket pipelines, pero mostrando información de la máquina usada en Heroku. Se ha realizado también la integración con Slack de Heroku en el proyecto.

3.8.2. Despliegue continuo

El despliegue continuo consiste en el paso automatizado del software de desarrollo a producción, una vez pasa unos requisitos mínimos, como son que pueda construirse y que pase todos los test. Esto implica que, por ejemplo, si se está desarrollando una web, y se añade un botón con una nueva funcionalidad, este botón aparecerá en la versión en producción, la que usan los

usuarios, si no hay ningún tipo de problema al construir el proyecto y todos los test pasan.

Una figura de todo el proceso puede verse en la figura 3.4.

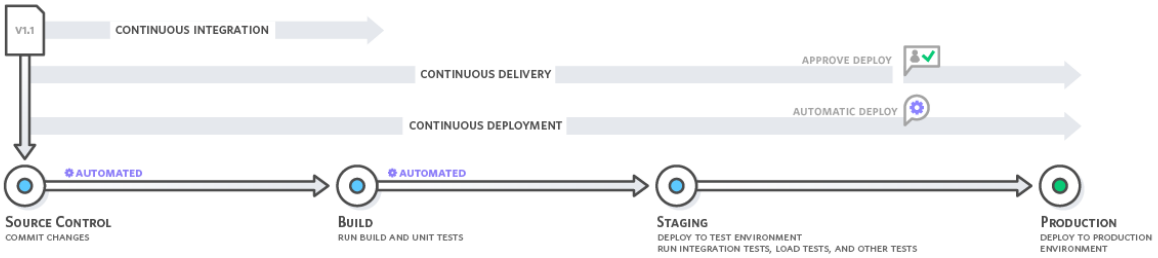

Figura 3.4: Diagrama de la integración continua, el despliegue continuo y la integración continua. Fuente [4]

Heroku ofrece herramientas para realizar el despliegue continuo, cada proyecto tiene asociado un repositorio asociado, desde el que se construye y despliega el proyecto. Pasar los cambios a producción implica subir los cambios a este repositorio.

El proceso de integración continua y despliegue sigue los siguientes pasos:

1. Instalación de las dependencias del proyecto
2. Construcción del proyecto
3. Ejecución de los test
4. Subida de los últimos cambios (HEAD) al repositorio de Heroku
5. La máquina encargada del despliegue, en Heroku, repetirá el punto 1 y 2, si tiene éxito, el despliegue estará completo.

Todo esto se ejecuta en la máquina encargada de la integración continua, después de la subida de cualquier cambio. Cualquier fallo en uno de estos puntos parará el proceso.

3.9. Herramientas para la planificación

En esta sección se introducirá la herramienta que se ha utilizado para planificar el proyecto, se explica lo que ofrece dicha herramienta y qué opciones se tenían.

3.9.1. Jira

Jira [67] es una herramienta web de pago que permite la gestión de proyectos, incidencias y errores, el seguimiento y control del tiempo, dentro de un proyecto, de cada uno de los

participantes, entre otras muchas funciones. Tiene un módulo *Agile* que ofrece una pizarra *Kanban* y diversas herramientas que permiten gestionar *sprints*.

Se ha usado esta herramienta para gestionar el proyecto y el tiempo que se ha empleado, fundamentalmente porque es la que se emplea en la empresa en la que se ha realizado. No se ha utilizado el módulo *Agile* porque no está incluído en la versión que se emplea en la empresa.

Alternativas que sean tan completas como Jira hay pocas, aunque una muy interesante es Youtrack [68].

Capítulo 4

Planificación del proyecto

En este capítulo se introduce la metodología de planificación empleada en el proyecto, por qué se ha seleccionado dicha metodología en cuestión y no otra, la planificación realizada, la estimación de recursos y costes del proyecto y, por último, el seguimiento que se ha realizado en el proyecto en cada revisión de *sprints*.

4.1. Metodología

Dado que:

1. Las tecnologías que se han empleado son novedosas, y existía la posibilidad de que alguna de ellas pudiese cambiar, o decidirse que debería sustituirse por otra a lo largo del proyecto.
2. Los requisitos concretos de la aplicación podían cambiar a lo largo de su desarrollo, siendo necesario añadir o eliminar alguna sección de las que se han propuesto.

Se ha optado por el uso de una metodología ágil para la gestión del proyecto, más concretamente se ha optado, en la medida de lo posible, por seguir **Scrum**.

Se ha optado por *Scrum* frente a otras opciones porque:

1. La aplicación tiene diversas secciones, lo que facilita la división del proyecto en *sprints*, permitiendo, de esta forma, obtener partes funcionales del proyecto lo antes posible.
2. Era posible una comunicación fluída y constante con el representante de la empresa, lo que facilita mucho el uso de esta metodología.
3. Algunas partes del desarrollo de la aplicación eran similares a otras, por lo que las *retrospectivas* podían funcionar como un proceso de mejora constante, aprendiendo de los fallos cometidos.

4. Bastante clara y fácil de seguir, además, el autor cuenta ya con cierta experiencia en este tipo de metodologías.

4.1.0.1. Por qué Scrum y no eXtreme Programming (XP)

Scrum es un método para equipos en el que se recomiendan equipos de 3 a 9 personas. No obstante, en este caso el proyecto lo ha llevado una persona, que ha actuado de *Scrum master* y del equipo. No parece tener sentido haber aplicado Scrum, ya que, por ejemplo, las reuniones diarias no tendrían ningún sentido con una única persona.

Cabe destacar, por ello, que de Scrum se ha aplicado el uso de sprints, planificaciones de sprints, revisiones del sprint y retrospectivas, y no las reuniones diarias.

Otra opción era usar **eXtreme Programming**, pero a fin de cuentas es menos adecuada que Scrum en este caso, ya que también hay una serie de roles que en este caso no se podían cumplir de forma adecuada. En esta metodología también se recomienda la programación por pares, imposible con una persona.

En este sentido sí que se ha tratado de aplicar, en lo posible, el resto de principios de eXtreme Programming, como son el uso de ATDD (*Aceptance Test Driven Development*), TDD (*Test Driven Development*) y la refactorización continua a fin de obtener un producto de calidad, buena parte de estos principios también suelen aplicarse en Scrum.

4.1.1. Funcionamiento de la metodología de trabajo propuesta

Antes de exponer el funcionamiento de *Scrum* es importante presentar los **roles** que encontramos en esta metodología, los **artefactos** y los **eventos** que son definidos en esta metodología.

4.1.1.1. Roles de Scrum

Tenemos al **propietario del producto**, o en su defecto un representante del mismo. Es quien determina las prioridades en las funcionalidades que hay que desarrollar, además de realizar un seguimiento a lo largo del desarrollo.

En segundo lugar tenemos a los **interesados en el producto**. Toman un rol secundario, observando y asesorando junto al propietario del producto en algunas de las fases en las que está presente.

En tercer lugar tenemos al **scrum master**, encargado de gestionar y facilitar el seguimiento de *Scrum*.

Por último tenemos al **equipo de desarrollo**, quienes construyen el producto.

4.1.1.2. Artefactos de Scrum

En primer lugar tenemos la **pila del producto**, que es un listado de todos los requisitos que ha de cumplir el sistema, ordenados por prioridad, y abierta a todas las personas que participan en el proyecto. Normalmente los requisitos están expresados en *historias de usuario*, aunque esto no es imprescindible.

En segundo lugar tenemos la **pila del sprint**, que es un subconjunto de requisitos de la *pila del producto* que van a ser desarrollados en un *sprint*.

En tercer lugar tenemos un **incremento** o un **entregable**, que es una pieza del producto final utilizable y probada, producto de un *sprint*.

Por último tenemos la ***burndown chart***, que es una gráfica en la que se mide la cantidad de requisitos de la pila del producto que quedan por resolver al comienzo de cada sprint, por lo que se puede observar el estado del proyecto. La pendiente de la línea será descendente al ir resolviendo requisitos y ascendente en el caso de que se añadan nuevos. Se va actualizando al ir completando sprints.

4.1.1.3. Eventos de Scrum

En primer lugar, tenemos la **planificación del sprint**, reunión en la que asisten el propietario del producto, el equipo de desarrollo y el scrum master.

En esta reunión el propietario del producto expone las prioridades y el equipo de desarrollo mide el esfuerzo de los requisitos y crea la pila del sprint. En caso de decidirse que va a haber más de un sprint a la vez, se crean las pilas de sprint que sean necesarias.

En segundo lugar, tenemos el **sprint**, que es un ciclo de desarrollo en el que el equipo desarrolla el listado de funcionalidades de la pila del sprint. Pueden haber diversos sprint a la vez, cada uno con su pila del sprint. La duración del sprint ha de ser inferior a 6 semanas, se recomienda que sea inferior a un mes.

Por cada sprint, además, tenemos la **reunión diaria**, que, como su nombre indica, es una reunión que se celebra diariamente, con la particularidad de estar limitada a una duración máxima de 15 minutos, en la que cada miembro del equipo resume lo que va a hacer, lo que ha hecho, y si ha tenido o cree que va a tener algún tipo de problema. También se actualiza la pila del sprint.

En tercer lugar tenemos la **revisión del sprint**, que es una reunión informativa, de máximo 4 horas, celebrada al terminar un sprint, en la que se presentan los resultados, se plantean sugerencias y se anuncia el próximo sprint.

Por último tenemos la **retrospectiva**, que es una reunión en la que el equipo de desarrollo y el scrum master analizan cómo han trabajado, y tratan de mejorar y corregir los errores que puedan estar cometiendo. Se celebra después de cada revisión del sprint.

4.1.1.4. Pasos a seguir en Scrum

Teniendo en mente los roles, artefactos y eventos de *Scrum*, su funcionamiento, que es iterativo, puede resumirse en los siguientes pasos:

1. Se realiza una reunión inicial en la que el propietario del producto, el equipo de desarrollo y el scrum master definen la pila del producto.
   En esta reunión también se priorizan las tareas de la pila del producto, se resuelven las dudas que puedan surgir, y el equipo de desarrollo estima el esfuerzo necesario para cada una de las tareas de la pila del producto. Por último, se definen uno o más sprints, cada uno con su pila del sprint y un equipo asociado.
   Cada sprint funcionará de manera independiente y se seguirán los puntos que se listan a continuación en cada uno de ellos.

2. El equipo celebra una reunión diaria, en el que se va actualizando el estado de la pila del sprint. El propietario del producto y los interesados pueden asistir a esta reunión como oyentes.

3. Al terminar el sprint se realiza una reunión de revisión del sprint, a la que asiste el propietario del producto, los interesados, el equipo y el scrum master.

4. El equipo y el scrum master realizan una retrospectiva.

5. Se planifica un nuevo sprint, se repite el punto 1 a excepción de la definición de la pila del producto y de la estimación del esfuerzo, a menos que se añadan nuevas tareas a la pila del producto, de ser así se cambiará la pila del producto y se estimará el esfuerzo de los nuevos requisitos. Se define un nuevo sprint como resultado de la reunión.

6. Se repiten los pasos desde el punto 2 hasta el punto 5 hasta que no queden más funcionalidades por añadir al producto, momento en el que no se definirán más sprints. El producto estará terminado en el momento en el que terminen los sprints restantes, se realicen las reuniones de revisión del sprint correpondientes y se realice la entrega del mismo.

Una síntesis de este proceso, para un único sprint, puede verse en la figura 4.1.

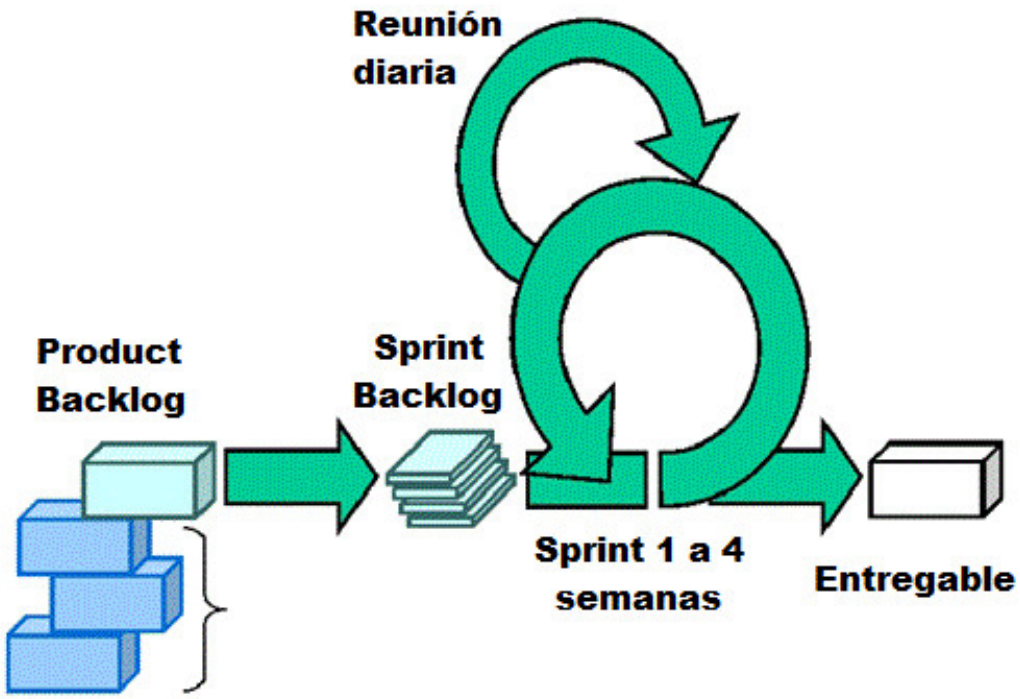

Figura 4.1: Esquema del proceso seguido en Scrum. Fuente [5]

4.1.2. Historias de usuario

Las historias de usuario [69] son un método para sintetizar los requisitos del usuario.

El objetivo de su uso es el de administrar los requisitos del usuario sin elaborar extensos documentos formales, cuya administración suele conllevar mucho tiempo, además de ser poco adaptables al cambio. Es por ello que las historias de usuario tratan de ser lo más sucintas posible y tienen la particularidad de escribirse usando el lenguaje del usuario.

Suelen usarse mucho en las metodologías ágiles, que buscan precisamente el ser flexibles y adaptarse mejor a los cambios en los requisitos.

Las historias tienen que responder siempre a tres preguntas:

- ¿Quien se beneficia?
- ¿Qué es lo que se quiere lograr?
- ¿Cual es el beneficio obtenido?

Para responder a estas tres preguntas, se suele usar la siguiente estructura en su elaboración:

Como [Rol] quiero [algo] para poder [beneficio].

Además de responder a las tres preguntas, cada historia de usuario ha de cumplir también los

siguientes requisitos:

- **Ser independientes unas de otras**: Cada historia de usuario ha de hacer referencia a un requisito y sólo a uno. Las que no lo cumplan han de subdividirse o reformularse.
- **Ser negociables**: Ha de discutirse el alcance de las mismas con los usuarios, y el resultado ha de quedar explícito en las pruebas de validación.
- **Ser valoradas por los clientes**: Han de tener más importancia para los clientes que para el desarrollador.
- **Ser estimables**: Es necesario para poder estimar el tiempo total que durará el proyecto.
- **Ser pequeñas**: Permite reducir mucho la complejidad especialmente a la hora de planificar.
- **Ser verificables**: Ha de poder existir un método para poder verificar el resultado que se tiene que obtener. Este método debería poder automatizarse cuando sea posible. Uno de los métodos que existen son las *pruebas de aceptación*, que aseguran que el componente en cuestión cumple con lo que el usuario quiere que haga.

4.1.3. Pruebas de aceptación

Las pruebas de aceptación son un tipo de prueba que aseguran que el sistema cumpla con aquello que el usuario quiere que haga. Cobran especial interés con el uso de las historias de usuario, puesto que las pruebas de aceptación suelen derivarse de los escenarios existentes en cada una de ellas.

Suelen emplearse en metodologías de desarrollo como *ATDD* [70] (*Acceptance Test-Driven Development*).

Tienen una estructura como la que se muestra a continuación:

Dado (un estado inicial del sistema)
Cuando (ocurre un evento)
Entonces (respuesta del sistema)

4.2. Planificación

La planificación que se ha seguido puede resumirse en el listado que se muestra a continuación:

1. Periodo de formación
2. Creación de la pila del producto inicial
3. Planificación inicial de los *sprint*
4. Desarrollo de los *sprint*

Periodo de formación

Las dos primeras semanas del proyecto se han dedicado a tareas de formación principalmente en *Angular* e *Ionic*, puesto que el desarrollador únicamente tenía experiencia en *Angular*, a un nivel muy básico.

Una vez terminado este periodo de formación, cuando ha sido necesario formarse en otro tipo de tecnología, a lo largo del proyecto, se ha hecho en la fase de planificación inicial del *sprint* y en los primeros días del mismo, teniendo en cuenta este periodo de formación en el *sprint* incluyendo menos carga de trabajo, según la estimación del tiempo que se fuese a dedicar a formación, para que todo cuadrase.

En el término de este periodo de formación se ha realizado la propuesta técnica del proyecto.

Creación de la pila del producto inicial

La planificación del primer sprint también ha incluido, en primer lugar, la creación de la pila del producto inicial, en forma de historias de usuario.

Planificación inicial de los *sprint*

Se ha decidido dividir el proyecto en 5 sprints, todos de dos semanas, repartiendo todas las historias de usuario de la pila del producto en estos 5 sprints

Como en este caso el desarrollo lo ha llevado una persona, todas las historias de usuario se han asignado a la misma persona. De haber sido necesario desarrollarlas en un determinado orden, se ha tenido en cuenta en la planificación.

Desarrollo del *sprint*

En el desarrollo se ha tratado de seguir TDD en la medida de lo posible, en algunas fases se ha optado por realizar pruebas mínimas por la falta de tiempo para revisar los test de forma más exhaustiva y para formarse en determinadas herramientas para *testing*, como *protractor*.

4.2.1. Diagrama de Gantt de la planificación

El resultado de la planificación puede resumirse en la figura 4.2.

4.3. Estimación de recursos y costes del proyecto

4.3.1. Estimación de recursos

El listado de recursos del proyecto incluye todo aquello que ha sido necesario para la consecución del mismo, excluyendo, por simplicidad, todos los componentes software citados en los puntos 3.1, 3.2, 3.3, 3.4 y 3.5 de este documento, puesto que todos son gratuítos y de libre acceso, no ha sido necesario comprar ninguna licencia ni realizar ningún registro.

También se omiten los gastos derivados del lugar de trabajo en el tiempo que ha durado el proyecto, como es el coste del suministro eléctrico.

El listado se cita a continuación:

- 1 ordenador portátil
- 1 IDE WebStorm con licencia de estudiante
- 1 navegador web Chrome
- 1 cuenta de usuario en Jira
- 1 cuenta de usuario en Bitbucket
- 1 licencia en Bitbucket Pipelines
- 1 cuenta de usuario en Codecov
- 1 cuenta de usuario en Slack
- 1 cuenta de correo corporativa en Gmail
- 1 cuenta de usuario en mLab
- 1 cuenta de usuario en Heroku
- 1 desarrollador

4.3.2. Estimación de costes

El coste total del proyecto es la suma del coste de todos los recursos empleados en el proyecto mas la ganancia que se quiera obtener.

Siendo un proyecto de prácticas, no se va a presupuestar un margen de ganancias, por lo que el coste del proyecto equivaldrá al coste de los recursos. Sí que se incluirá el margen de ganancias en los cálculos, pero con un valor de 0.

La estimación de costes del proyecto es la siguiente:

- Se ha usado un ordenador portátil de 700€con una vida útil estimada de unos 5 años (60 meses). Teniendo el proyecto una duración estimada aproximada de unos 3 meses, el coste amortizado del ordenador (vida útil del mismo que se ha empleado) ha sido de 35€

Figura 4.2: Diagrama Gantt del proyecto.

- La licencia de estudiante del IDE WebStorm es gratuíta mientras se sigan cursando los estudios, por lo que el coste ha sido de 0€
- El navegador web Chrome es de uso gratuíto, por lo que el coste ha sido de 0€
- La empresa ya tenía contratada una licencia de Jira, el añadir un nuevo usuario no ha tenido ningún coste, puesto que estas licencias operan sobre un número máximo de usuarios y no se ha alcanzado este límite, por tanto el coste ha sido de 0€
- La licencia de Bitbucket tiene un funcionamiento similar a la de Jira para proyectos privados en Bitbucket, como no se ha alcanzado el límite de usuarios, el coste ha sido de 0€
- La licencia de Bitbucket pipelines era gratuíta para un número máximo de minutos de uso de la máquina remota, cumplía más que suficiente los requisitos del proyecto y tampoco ha habido ningún coste en esto.
- Codecov es gratuíto para un proyecto privado, como no se tenía ninguno más añadido, no ha tenido coste.
- Se ha optado por el uso de un canal de Slack gratuíto, puesto que el límite de 10.000 mensajes no afectaba en absoluto.
- La cuenta de correo de Gmail corporativa se ha utilizado para el registro en el resto de servicios, como Jira, mLab y Heroku. Ya estaba contratada en la empresa y no ha tenido coste.
- Se ha optado por emplear el plan gratuíto que ofrece mLab, con un límite de 500MB, puesto que al principio el sistema tendrá muy pocos datos.
- Heroku ofrece también un plan gratuíto con un límite de uso de la máquina remota al mes, era más que suficiente para el uso actual del sistema.
- El desarrollador ha empleado un total de 300h. Partiendo de una estimación de 20€a la hora, incluyendo dentro de este coste todo tipo de gastos (seguridad social, impuestos, etc...), el coste total es: 20 * 300 = 6000€

Se estima que se quiere obtener una ganancia del 25 % del total de los costes, por lo que se sumará este valor al total.

El **coste total del proyecto** asciende a: 35 + 6000 + 0.25 * 6035 = 7543,75€

4.4. Seguimiento del proyecto

El uso de una metodología ágil como *Scrum* facilita en parte el seguimiento del proyecto, puesto que se realiza de forma continuada en cada *sprint*.

El seguimiento del proyecto se ha ido realizando al término de cada uno de los *sprints* y antes de comenzar el *sprint* siguiente.

Se ha comprobado, al terminar cada uno de los *sprints* si la planificación se estaba cumpliendo

conforme a lo previsto, es decir, que se habían realizado todas las historias de usuario que se habían asignado a dicho *sprint* y no se habían hecho menos o más. También se ha comprobado, antes de planificar el *sprint* siguiente, que la pila del producto no había cambiado.

En el caso de haber terminado las historias de usuario antes de tiempo, se han tomado más de la pila del producto para seguir trabajando. Cuando han habido retrasos, las historias de usuario que han quedado se han pasado al *sprint* siguiente, y se ha reducido la carga del mismo para mantenerla igual que el resto.

Se ha dado un caso en el que la pila del producto ha cambiado. En este caso, se han añadido las historias nuevas y se ha ajustado la prioridad de cada una de ellas, para saber cuales hacer primero, como resultado se ha implementado toda la parte servidor y no se ha implementado la sección de realidad virtual, la de networking ni la de interacción con los ponentes.

Capítulo 5

Definición de requisitos

En esta sección se va a exponer la definición de requisitos. Se comienza introduciendo todas las historias de usuario, son muchas, y es algo que complica mucho la visión del sistema a alto nivel, por lo que se continúa con la definición de módulos, donde se detalla cómo se han clasificado y agrupado todas las historias de usuario.

La definición de los módulos da lugar a los diagramas de casos de uso, donde se expone gráficamente todo el sistema a alto nivel. Por último, se concluye con la estimación de la pila del producto, en la que se toma el listado con todas las historias de usuario y se les asigna un valor en puntos de historia.

5.1. Historias de usuario

En esta sección se definen todas las historias de usuario del proyecto, incluyendo también sus escenarios, de los que se derivan las pruebas de aceptación.

Una tabla resumen de todas las historias de usuario se muestra a continuación:

Código	**Nombre de la historia de usuario**
HU01	Pantalla de carga
HU02	Pantalla principal (no autenticado)
HU03	Pantalla principal (autenticado)
HU04	Pantalla principal (más información)
HU05	Menú (no autenticado)
HU06	Menú (autenticado)
HU07	Listado de noticias
HU08	Noticia en detalle
HU09	Sección de la agenda
HU10	Detalles de un evento de la agenda
HU11	Sección de ponentes
HU12	Detalles de un ponente
HU13	Sección de documentos (no autenticado)
HU14	Sección de documentos del evento (autenticado)
HU15	Sección de mapas
HU16	Detalles de un mapa
HU17	Sección de servicios cercanos al congreso (listado)
HU18	Sección de servicios cercanos al congreso (mapa)
HU19	Mostrar un servicio en el mapa
HU20	Detalles de un servicio
HU21	Sección de patrocinadores
HU22	Sección 'Acerca de'
HU23	Sección de aspectos legales
HU24	Retroalimentación a los organizadores
HU25	Sección de networking (sin activar)
HU26	Sección de networking (activa)
HU27	Listado de contactos
HU28	Pantalla de gestión de un emparejamiento
HU29	Sección de preguntas al ponente
HU30	Detalles de la pregunta a un ponente
HU31	Crear una pregunta para un ponente
HU32	Comentar una pregunta a un ponente
HU33	Sección de notificaciones (sin log)

Cuadro 5.1: Listado de historias de usuario junto con su nombre. (I)

Código	Nombre de la historia de usuario
HU34	Sección de notificaciones (logueado)
HU35	Sección de realidad aumentada (no autenticado)
HU36	Sección de realidad aumentada (autenticado)
HU37	Escaneo de un objeto de realidad aumentada
HU38	Ver los datos de inscripción
HU39	Inscripción por medio de linkedIn
HU40	Formulario de inscripción
HU41	Ver mi acreditación
HU42	Ver mi perfil
HU43	Editar mi perfil

Cuadro 5.2: Listado de historias de usuario junto con su nombre. (II)

A continuación se detallan todas las historias de usuario, incluyendo los escenarios de cada una de ellas. Por simplicidad, no se han incluido las pruebas de aceptación, puesto que están escritas usando datos de ejemplo en cada caso, lo que las hace muy extensas y dificultaría al lector el seguimiento de las historias de usuario.

5.1.1. HU01 - Pantalla de carga

Como un <usuario> quiero ver una pantalla de carga al inicial la aplicación para saber que está iniciándose.

Escenarios:

- Dado un usuario inscrito o no inscrito, cuando inicie la aplicación, entonces me mostrará una pantalla de carga hasta que esté lista.

5.1.2. HU02 - Pantalla principal (no autenticado)

Como un <usuario no autenticado> quiero ver una pantalla principal al iniciarse la aplicación con el cartel del evento, un acceso al menú de opciones y otro para realizar la inscripción para poder seleccionar cualquiera de estas opciones de forma rápida y cómoda.

Escenarios:

- Dado un usuario no autenticado, cuando la aplicación esté en la pantalla principal, entonces se mostrará el cartel del evento y un botón para realizar la inscripción.

5.1.3. HU03 - Pantalla principal (autenticado)

Como un <usuario autentificado> quiero ver una pantalla principal al cargar la aplicación en la que aparezca como fondo el cartel del evento y con un acceso al menú de opciones para poder acceder a las opciones de forma fácil y rápida.

Escenarios:

- Dado un usuario autenticado, cuando la aplicación esté en la pantalla principal, entonces se mostrará el cartel del evento.

5.1.4. HU04 - Pantalla principal (más información)

Como un <usuario> quiero que al realizar una acción de scroll vertical hacia arriba en la pantalla principal aparezca la información del congreso para poder leerla de forma cómoda.

Escenarios:

- Dado un usuario no autenticado, cuando la aplicación esté en la pantalla principal y realice scroll vertical, entonces se mostrará la información del congreso.
- Dado un usuario autenticado, cuando la aplicación esté en la pantalla principal y realice scroll vertical, entonces se mostrará la información del congreso.

5.1.5. HU05 - Menú (no autenticado)

Como un <usuario no autenticado> quiero poder ver un menú con acceso a la pantalla principal y a las siguientes secciones: noticias, agenda, ponentes, documentación, mapas, servicios, notificaciones, patrocinadores, acerca de y a un acceso a la inscripción que muestre el precio mínimo del evento, para poder conocer más información del evento antes de inscribirme y tener un acceso rápido para ello.

Escenarios:

- Dado un usuario no autenticado, cuando realice scroll horizontal hacia la derecha, en cualquier pantalla, entonces el sistema mostrará el menú con las secciones públicas y un botón para poder inscribirme.

5.1.6. HU06 - Menú (autenticado)

Como un <usuario autenticado> quiero poder ver un menú con acceso a las siguientes secciones de mi perfil: networking, pregunta al ponente, realidad aumentada y acreditación, además del resto de opciones que tiene un usuario no autenticado a excepción del acceso a la inscripción, para poder acceder y usar cualquiera de ellas de forma rápida y cómoda.

Escenarios:

- Dado un usuario autenticado, cuando realice scroll horizontal hacia la derecha, en cualquier pantalla, entonces el sistema mostrará el menú con las secciones públicas y las secciones privadas.

5.1.7. HU07 - Listado de noticias

Como un <usuario> quiero poder acceder a una sección de noticias en la que aparezca el listado de noticias del evento permitiéndome acceder a la noticia completa pulsando sobre ella para poder verlas y leer la que quiera.

Escenarios:

- Dado un usuario, autenticado o no, cuando entre en la sección de noticias, entonces el sistema cargará y me mostrará un listado de noticias.

5.1.8. HU08 - Noticia en detalle

Como un <usuario> quiero poder acceder a una noticia completa para poder leerla.

Escenarios:

- Dado un usuario, autenticado o no, dentro de la sección de noticias, cuando pulse sobre una de las noticias del listado, entonces el sistema me mostrará la noticia completa.

5.1.9. HU09 - Sección de la agenda

Como un <usuario> quiero poder ver una agenda con un listado de todos los eventos del congreso mostrando el nombre y la hora del evento, permitiéndome acceder a sus detalles al pulsar sobre él y con un calendario desplegable en la parte superior que me permita moverme más rápidamente para poder conocer los eventos del congreso.

Escenarios:

- Dado un usuario, autenticado o no, cuando entre en la sección de la agenda, entonces el sistema cargará y me mostrará los eventos del mes seleccionado.
- Dado un usuario, autenticado o no, dentro de la sección de la agenda, cuando cambie el mes seleccionado, entonces el sistema cargará y me mostrará los eventos del nuevo mes seleccionado.
- Dado un usuario, autenticado o no, dentro de la sección de la agenda, cuando expanda el calendario, entonces el sistema resaltará los días del mes seleccionado en los que hay un evento.

5.1.10. HU10 - Detalles de un evento de la agenda

Como un <usuario> quiero poder acceder a una pantalla con los datos del evento que he seleccionado, que tendrá un título, un texto con la información y podrá o no tener una foto junto al título.

Escenarios:

- Dado un usuario, autenticado o no, dentro de la sección de la agenda, cuando pulse sobre uno de los eventos del listado que no tiene ninguna foto asociada, entonces el sistema cargará y me mostrará el evento en detalle, mostrando un fondo rojo en el espacio e la foto.
- Dado un usuario, autenticado o no, dentro de la sección de la agenda, cuando pulse sobre uno de los eventos del listado en el que no participa ningún ponente, entonces el sistema cargará y me mostrará el evento en detalle, en el que no se muestra la fila en la que debiera aparece el nombre y la foto del ponente.
- Dado un usuario, autenticado o no, dentro de la sección de la agenda, cuando pulse sobre uno de los eventos del listado en el que participa un ponente, entonces el sistema cargará y me mostrará el evento en detalle, mostrando el nombre y la foto del ponente.

5.1.11. HU11 - Sección de ponentes

Como un <usuario> quiero poder ver un listado con los ponentes del evento, viendo una foto y una breve descripción de cada uno de ellos, junto a accesos a sus redes sociales y a la sección de preguntas a un ponente, pudiendo también acceder a una descripción más detallada al pulsar sobre alguno de ellos para poder saber quien va al evento y su información.

Escenarios:

- Dado un usuario, autenticado o no, cuando entre en la sección de ponentes, entonces el sistema cargará y me mostrará un listado de ponentes.

5.1.12. HU12 - Detalles de un ponente

Como un <usuario> quiero poder acceder a los detalles de un ponente viendo la misma información de ellos que en la sección de ponentes pero con el texto ampliado y únicamente la información de ese ponente, con un botón para volver al listado, para poder conocer los detalles del ponente.

Escenarios:

- Dado un usuario, autenticado o no, dentro de la sección de ponentes, cuando pulse sobre uno de los ponentes, entonces el sistema cargará y me mostrará los detalles de ese ponente.

5.1.13. HU13 - Sección de documentos (no autenticado)

Como un <usuario> quiero poder ver un listado de los documentos públicos del evento pudiendo descargarlos pulsando sobre ellos para poder saber qué información hay publicada y tener acceso a ella.

Escenarios:

- Dado un usuario no autenticado, cuando entre en la sección de documentos, entonces el sistema me mostrará un listado de documentos públicos.

5.1.14. HU14 - Sección de documentos (autenticado)

Como un <usuario autentificado> quiero poder ver un listado de mis documentos del congreso, además de los documentos públicos que puede ver un usuario no autentificado, para poder acceder a los datos de mi inscripción entre otras cosas.

Escenarios:

- Dado un usuario autenticado, cuando entre en la sección de documentos, entonces el sistema me mostrará un listado de documentos públicos y otro con mis documentos privados.

5.1.15. HU15 - Sección de mapas

Como un <usuario> quiero poder ver un listado de mapas relacionados con el evento que incluya: un plano del evento explicativo, la localización del espacio donde se celebra, la localización de los patrocinadores y cualquier otro dato que sirva, pudiendo acceder a su información pulsando sobre el elemento para poder conocer dónde se encuentra cada cosa.

Escenarios:

- Dado un usuario autenticado o no, cuando entre dentro de la sección de mapas, entonces el sistema cargará y mostrará un listado de mapas.

5.1.16. HU16 - Detalles de un mapa

Como un <usuario> quiero poder ver un mapa ampliado de una de las localizaciones del evento que muestre los detalles de esta localización y un botón que me diga cómo llegar para poder ir a la localización en cuestión.

Escenarios:

- Dado un usuario autenticado o no, dentro de la sección de mapas, cuando pulse sobre uno de los mapas del listado, entonces el sistema cargará y mostrará la localización del lugar que se ha pulsado sobre un mapa.

- Dado un usuario autenticado o no, dentro de un mapa cargado, cuando pulse sobre el botón 'ir a', entonces el sistema obtendrá mi ubicación y trazará sobre el mapa el camino más corto hasta la ubicación.

5.1.17. HU17 - Sección de servicios cercanos al congreso (listado)

Como un <usuario> quiero poder ver un listado con los servicios cercanos al congreso y poder ver su localización en el mapa al pulsar sobre ellos, o cambiar a la vista de mapa, para saber dónde hay hoteles o restaurantes.

Escenarios:

- Dado un usuario, autenticado o no, cuando entre dentro de la sección de servicios, entonces el sistema cargará y mostrará un listado de los servicios cercanos al congreso.

5.1.18. HU18 - Sección de servicios cercanos al congreso (mapa)

Como un <usuario> quiero poder ver un mapa con todos los servicios cercanos marcados en el mismo y poder conocer los detalles pulsando sobre el icono en el mapa o ir a la vista en forma de listado, para poder saber dónde hay restaurantes u hoteles.

Escenarios:

- Dado un usuario, autenticado o no, dentro de la sección de servicios, cuando seleccione la opción de vista en mapa, entonces el sistema cargará y mostrará el listado de los servicios cercanos al congreso localizados en un mapa.

5.1.19. HU19 - Mostrar un servicio en el mapa

Como un <usuario> quiero poder ver el servicio centrado en el mapa, una opción que me diga cómo llegar hasta el mismo desde la posición del evento y un acceso a los detalles del servicio al pulsar sobre el texto del mismo para poder conocer la información del servicio.

Escenarios:

- Dado un usuario, autenticado o no, dentro de la sección de servicios y dentro de la vista en el mapa, cuando pulse sobre uno de los servicios del mapa, entonces el sistema lo centrará en pantalla y mostrará información del mismo.
- Dado un usuario, autenticado o no, dentro de la sección de servicios y dentro de la vista en el mapa, cuando pulse sobre uno de los servicios del mapa y pulse sobre la opción de 'cómo llegar', entonces el sistema obtendrá mi ubicación, trazará una línea sobre el mapa con el camino más corto, y centrará mi ubicación en pantalla.

5.1.20. HU20 - Detalles de un servicio

Como un <usuario> quiero conocer los detalles de un servicio, viendo una descripción y accesos a la web y ofertas de haberlas además de su posición en el mapa en la parte superior, pudiendo tener en lugar del mapa una foto, para poder conocer exactamente cómo es el servicio y poder identificarlo.

Escenarios:

- Dado un usuario, autenticado o no, dentro de la sección de servicios, cuando pulse sobre uno de los servicios del listado, entonces el sistema mostrará los detalles del servicio.

5.1.21. HU21 - Sección de patrocinadores

Como un <usuario> quiero poder ver un listado de los patrocinadores del evento clasificados por: organizadores, partners institucionales, partners tecnológicos y colaboradores, mostrando

los logos de cada uno de ellos para conocer a los implicados en el evento.

Escenarios:

- Dado un usuario, autenticado o no, cuando entre dentro de la sección de patrocinadores, entonces el sistema cargará y mostrará en pantalla los logos de los patrocinadores del evento.

5.1.22. HU22 - Sección 'Acerca de'

Como un <usuario> quiero poder tener una sección desde la que acceder a los detalles legales del evento o enviar sugerencias a los organizadores del evento para poder acceder a este tipo de detalles desde una sección destinada a ello.

Escenarios:

- Dado un usuario, autenticado o no, cuando entre dentro de la sección 'acerca de', entonces el sistema mostrará en pantalla dicha sección.

5.1.23. HU23 - Sección de aspectos legales

Como un <usuario> quiero poder ver un texto que detalle los aspectos legales del congreso para revolver mis dudas al respecto.

Escenarios:

- Dado un usuario, autenticado o no, dentro de la sección 'acerca de', cuando entre en la sección de aspectos legales, entonces el sistema mostrará en pantalla los aspectos legales del congreso.

5.1.24. HU24 - Retroalimentación a los organizadores

Como un <usuario> quiero poder enviar retroalimentación a los organizadores del evento a partir de un formulario de sugerencias destinado para ello para poder realizar críticas constructivas al evento.

Escenarios:

- Dado un usuario, autenticado o no, dentro de la sección 'acerca de', cuando entre en la sección de retroalimentación a los organizadores, rellene el formulario y lo envíe satisfactoriamente, entonces el sistema enviará los datos al servidor y mostrará en pantalla una confirmación de que todo ha ido bien.

- Dado un usuario, autenticado o no, dentro de la sección 'acerca de', cuando entre en la sección de retroalimentación a los organizadores, rellene el formulario y lo envíe, pero falle la conexión, entonces el sistema mostrará en pantalla un cuadro de error.

5.1.25. HU25 - Sección de networking (sin activar)

Como un <usuario autenticado> quiero ver una pantalla que me permita activar la opción de networking si así lo deseo para asegurarme de que está desactivada y poder activarla si quiero.

Escenarios:

- Dado un usuario autenticado, cuando entre dentro de la sección de networking cuando no está activada, entonces el sistema mostrará en pantalla un mensaje indicando que ha de activarse para poder usarse.

5.1.26. HU26 - Sección de networking (activada)

Como un <usuario autenticado> quiero que el sistema me ofrezca emparejamientos automáticos que yo pueda rechazar o aceptar y un acceso al listado de contactos de mi red de contactos, todo esto mientras tenga el networking activo, para poder crear mi red de contactos con los asistentes al congreso.

Escenarios:

- Dado un usuario autenticado, cuando entre dentro de la sección de networking cuando está activada, entonces el sistema mostrará en pantalla el listado de contactos y, de haber algún emparejamiento pendiente, lo mostrará en primer lugar.

5.1.27. HU27 - Listado de contactos

Como un <usuario autenticado> quiero poder ver un listado con los contactos del congreso con los que he hecho 'match', mostrando el nombre y los apellidos de cada uno de ellos junto a su foto, una pequeña descripción y un acceso para mandarle un correo electrónico o llamar por teléfono, para poder ver los contactos que tengo y contactar con ellos.

Escenarios:

- Dado un usuario autenticado dentro de la sección de networking cuando está activada, cuando pulse sobre el icono con una carta en uno de los contactos de la lista, entonces el sistema abrirá la aplicación para enviar correos electrónicos a esa dirección.
- Dado un usuario autenticado dentro de la sección de networking cuando está activada, cuando pulse sobre el icono con un teléfono en uno de los contactos de la lista, entonces el sistema iniciará una llamada telefónica a ese número.

5.1.28. HU28 - Pantalla de gestión de un emparejamiento

Como un <usuario autenticado> quiero que aparezca una pantalla que me avise de que he hecho 'match' con alguien, permitiéndome enviarle un correo o llamarle desde ahí, para saber que he hecho un nuevo contacto.

Escenarios:

- Dado un usuario autenticado a punto de entrar en la sección de networking, cuando existan emparejamientos por resolver, entonces el sistema los mostrará en pantalla, uno por uno, para que los acepte o rechace.

5.1.29. HU29 - Sección de preguntas al ponente

Como un <usuario autenticado> quiero poder ver todas las preguntas que se le han hecho a un ponente y acceder a la pregunta completa pulsando sobre ella, tener un acceso para crear una nueva o comentar una de las que hay, para poder interaccionar con el ponente.

Escenarios:

- Dado un usuario autenticado dentro de la sección de ponentes, cuando pulse sobre el botón para entrar en la sección de preguntas al ponente, entonces el sistema mostrará el listado de preguntas de ese ponente.

5.1.30. HU30 - Detalles de la pregunta a un ponente

Como un <usuario autenticado> quiero poder ver la pregunta completa que se le ha hecho a un ponente, ver el listado de comentarios debajo de esa pregunta y tener accesos para comentar la pregunta o crear una nueva desde ahí, para poder interaccionar con el ponente en una cuestión en concreto.

Escenarios:

- Dado un usuario autenticado dentro de la sección de preguntas al ponente, cuando pulse sobre una de las preguntas, entonces el sistema mostrará la pregunta completa y los comentarios que se han realizado.

5.1.31. HU31 - Crear una pregunta a un ponente

Como un <usuario autenticado> quiero poder crear una nueva pregunta destinada a un ponente en concreto y que esta pregunta aparezca en el listado de preguntas, para poder preguntarle cualquier cosa.

Escenarios:

- Dado un usuario autenticado dentro de la sección de preguntas al ponente, cuando pulse sobre el botón para crear una pregunta, la cree y la envíe correctamente, entonces el sistema publicará la pregunta y mostrará una pantalla indicando que todo ha ido bien.
- Dado un usuario autenticado dentro de la sección de preguntas al ponente, cuando pulse sobre el botón para crear una pregunta, la cree, pero no se envíe correctamente, entonces el sistema mostrará una pantalla indicando que ha habido un error, sin borrar la pregunta.

5.1.32. HU32 - Comentar una pregunta a un ponente

Como un <usuario autenticado> quiero poder comentar cualquier pregunta que se le haga a un ponente y que mi comentario aparezca debajo de la pregunta completa, para poder interaccionar con el ponente sobre un tema en cuestión.

Escenarios:

- Dado un usuario autenticado dentro de la sección de preguntas al ponente y dentro de una pregunta, cuando pulse sobre el botón para comentar la pregunta, cree el comentario y lo envíe correctamente, entonces el sistema publicará el comentario y mostrará una pantalla indicando que todo ha ido bien.
- Dado un usuario autenticado dentro de la sección de preguntas al ponente y dentro de una pregunta, cuando pulse sobre el botón para comentar la pregunta, cree el comentario, pero no lo envíe correctamente, entonces el sistema mostrará una pantalla indicando que ha habido un error, sin borrar el comentario.

5.1.33. HU33 - Sección de notificaciones (no autenticado)

Como un <usuario> quiero ver un listado con todas las incidencias que han ocurrido en el congreso, como cambios de última hora o eventos nuevos, para poder estar informado de los cambios que se realizan en el congreso.

Escenarios:

- Dado un usuario no autenticado, cuando entre en la sección de notificaciones, entonces el sistema mostrará las notificaciones públicas.

5.1.34. HU34 - Sección de notificaciones (autenticado)

Como un <usuario registrado> quiero ver un listado de notificaciones que me permita saber que se han hecho interacciones conmigo, además de los cambios en el congreso que puede ver un usuario no registrado, para poder conocer esta información.

Escenarios:

- Dado un usuario autenticado, cuando entre en la sección de notificaciones, entonces el sistema mostrará las notificaciones públicas y las notificaciones privadas de ese usuario.

5.1.35. HU35 - Sección de realidad aumentada (no autenticado)

Como un <usuario autenticado> quiero ver una pantalla que me indique que debo activar la realidad aumentada para usarla siendo consciente de ello.

Escenarios:

- Dado un usuario autenticado, cuando entre en la sección de realidad aumentada y no esté activada, entonces el sistema mostrará una pantalla indicando que ha de activarse para poder usarse.

5.1.36. HU36 - Sección de realidad aumentada (autenticado)

Como un <usuario autenticado> quiero ver una sección desde la que poder comenzar las interacciones con objetos del congreso que lo permitan para mejorar mi experiencia.

Escenarios:

- Dado un usuario autenticado, cuando entre en la sección de realidad aumentada y esté activada, entonces el sistema activará la cámara dentro de la aplicación para poder interactuar con los objetos con realidad aumentada.

5.1.37. HU37 - Escaneo de un objeto de realidad aumentada

Como un <usuario registrado> quiero escanear un objeto de realidad aumentada con la cámara de mi dispositivo móvil para poder interaccionar con él.

Escenarios:

- Dado un usuario autenticado dentro de la sección de realidad aumentada, con la cámara activada, cuando fije la cámara sobre un objeto de realidad aumentada, entonces el sistema lo escaneará y mostrará la interacción.

5.1.38. HU38 - Ver los datos de inscripción

Como un <usuario registrado> quiero tener una sección desde la que ver los datos de la inscripción que he hecho para poder consultarlos cuando quiera.

Escenarios:

- Dado un usuario autenticado, cuando entre en la sección de documentos, entonces el sistema mostrará el listado público y privado de documentos, estando en el privado un documento con los datos de inscripción.

5.1.39. HU39 - Inscripción por medio de LinkedIn

Como un <usuario> quiero inscribirme por medio de LinkedIn y que el sistema tome mis datos personales de esta plataforma para no tener que escribirlos a mano.

Escenarios:

- Dado un usuario no autenticado dentro del formulario de inscripción, cuando pulse sobre el botón de inscribirme mediante LinkedIn y rellene los datos de la cuenta, entonces el sistema obtendrá los datos, rellenará el formulario y lo enviará.

5.1.40. HU40 - Formulario de inscripción

Como un <usuario> quiero tener un formulario de inscripción para poder inscribirme en el congreso.

Escenarios:

- Dado un usuario no autenticado, cuando acceda a la inscripción, entonces el sistema mostrará un formulario de inscripción.

5.1.41. HU41 - Ver mi acreditación

Como un <usuario registrado> quiero una sección que muestre mi acreditación del congreso para poder mostrarla a los organizadores del mismo y tener acceso.

Escenarios:

- Dado un usuario autenticado, cuando entre en la sección de acreditación, entonces el sistema mostrará mi acreditación, que constará de mi foto de perfil, nombre, apellidos, cargo y el identificador único de mi cuenta, tendrá un código QR con este identificador para poder ser leído por los organizadores.

5.1.42. HU42 - Ver mi perfil

Como un <usuario registrado> quiero una sección desde la que ver mi perfil para saber qué datos personales tengo registrados en el sistema.

Escenarios:

- Dado un usuario autenticado, cuando acceda a la sección del perfil, entonces el sistema mostrará su perfil, con su foto de usuario, nombre y apellidos, cargo y las opciones de activar o no el networking y el modo 'snooze'.

5.1.43. HU43 - Editar mi perfil

Como un <usuario registrado> quiero una sección desde la que poder editar mi perfil para poder realizar cambios en el mismo si me he equivocado.

Escenarios:

- Dado un usuario autenticado dentro de la sección del perfil, cuando pulse sobre el botón de editar el perfil, entonces el sistema mostrará un formulario similar al de la inscripción, relleno, que podré editar para cambiar el perfil.

5.2. Definición de módulos

Todas las historias de usuario pueden clasificarse en módulos, según su ámbito funcional. Hacer esto facilita el entendimiento del sistema a alto nivel.

Asimismo, también se divide el sistema en dos subsistemas, la parte pública, accesible por cualquier usuario que utilice la aplicación, y la parte privada, accesible únicamente por aquellos usuarios que se hayan inscrito. Los módulos pertenecen a uno de estos dos subsistemas.

Se obtiene el siguiente listado de módulos. Se detalla en cada uno de ellos las historias de usuario que incluye.

Parte pública

- **Página principal**: HU01, HU02, HU03, HU04, HU05, HU06, HU39, HU40
- **Noticias**: HU07, HU08
- **Agenda**: HU09, HU10
- **Ponentes**: HU11, HU12, HU29, HU30
- **Documentación**: HU13, HU14, HU38
- **Mapas**: HU15, HU16
- **Servicios**: HU17, HU18, HU19, HU20
- **Notificaciones**: HU33, HU34
- **Patrocinadores**: HU21
- **Acerca de**: HU22, HU23, HU24

Parte privada

- **Perfil**: HU42, HU43
- **Preguntas al ponente**: HU31, HU32
- **Realidad aumentada**: HU35, HU36, HU37
- **Acreditación**: HU41
- **Networking**: HU25, HU26, HU27, HU28

5.3. Diagramas de casos de uso

Se muestran, a continuación, los diagramas de casos de uso, no han sido estrictamente necesarios para realizar el proyecto, como sí que lo han sido las historias de usuario, no obstante, nos muestran una visión del sistema a alto nivel y son muy útiles para mejorar la comprensión de cada una de las partes y del sistema en su conjunto.

Una visión inicial del sistema se muestra en la figura 5.1.

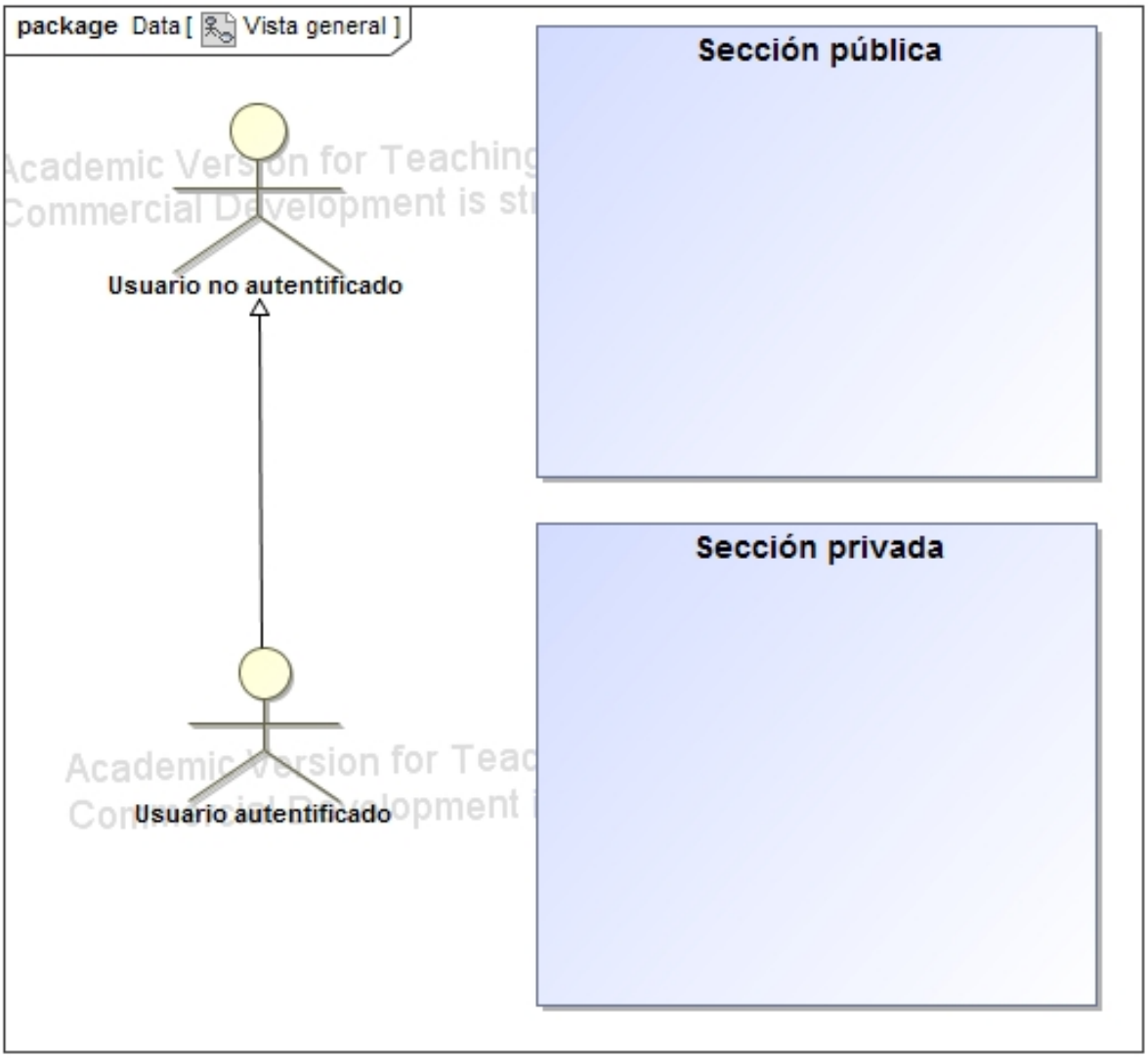

Figura 5.1: Diagrama de casos de uso que muestra los subsistemas en los que se divide el proyecto.

En este diagrama se ha mostrado únicamente los subsistemas en los que se ha dividido el proyecto, la parte pública y la parte privada. En las figuras 5.2 y 5.3 se muestran los diagramas de la parte pública y la parte privada, respectivamente.

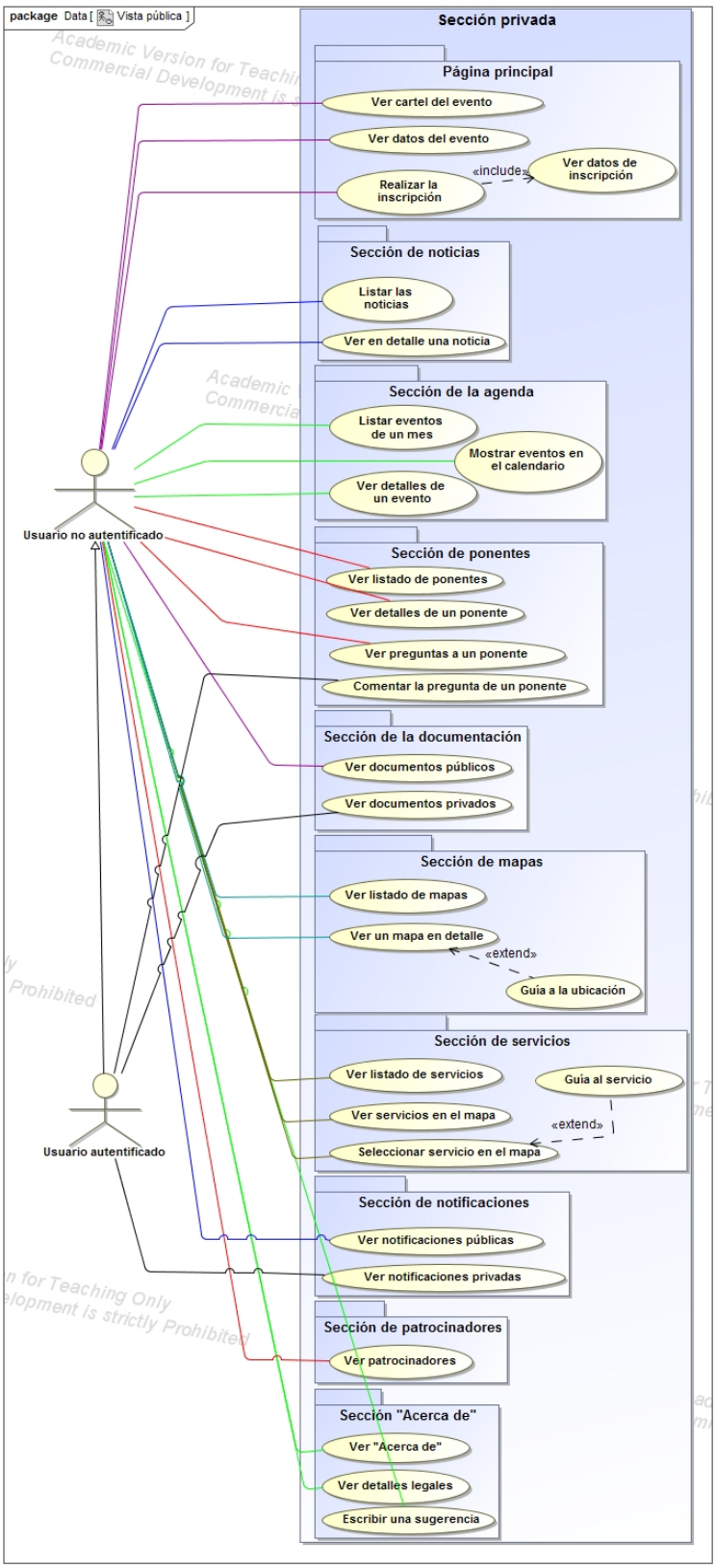

Figura 5.2: Diagrama de casos de uso que muestra el subsistema de la parte pública.

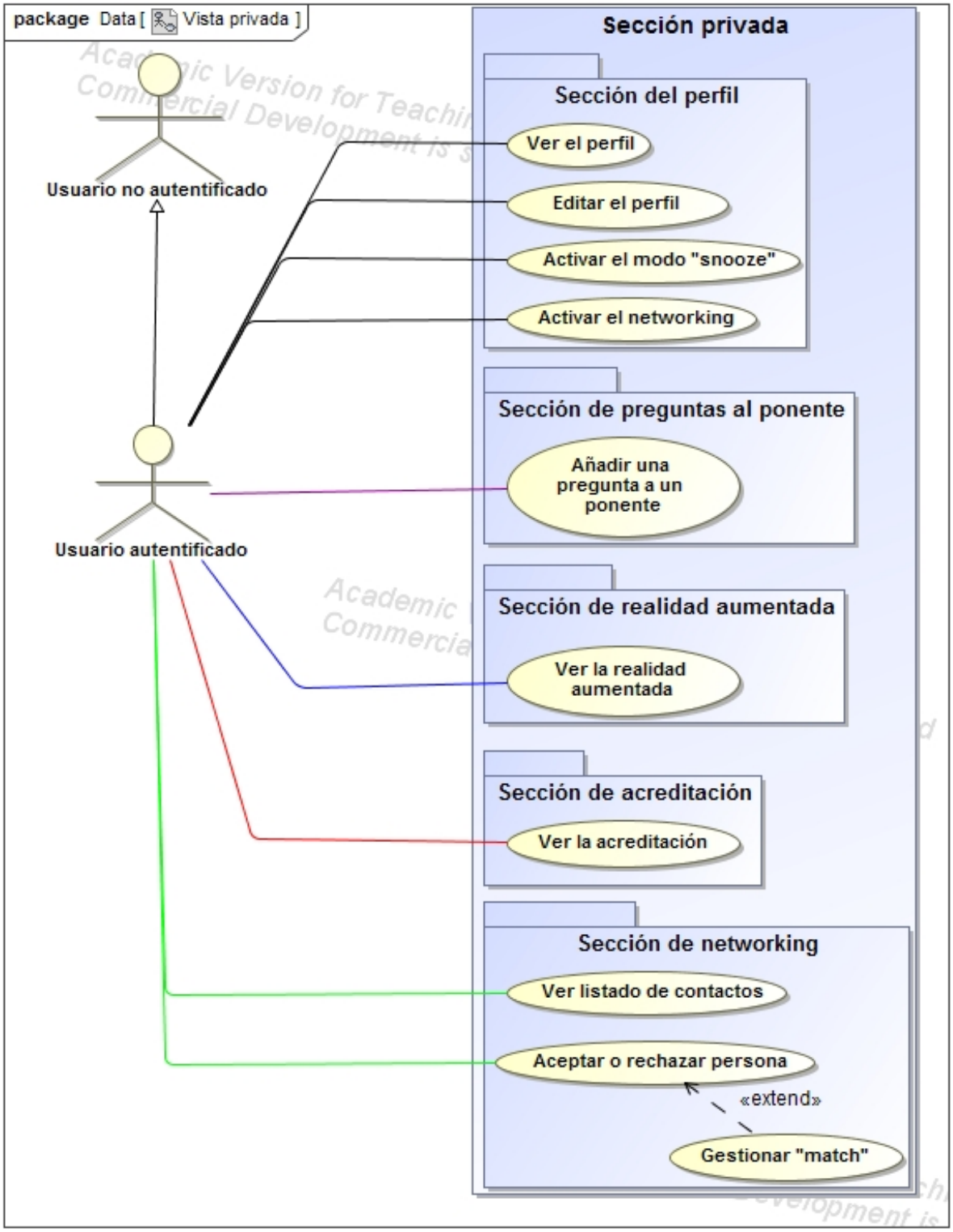

Figura 5.3: Diagrama de casos de uso que muestra el subsistema de la parte privada.

5.4. Estimación de la pila del producto

Una vez definidas todas las historias de usuario, hay que realizar una estimación que indique la cantidad de esfuerzo que conllevará terminar cada una de ellas, a fin de poder asignarlas a *sprints* y conocer al menos una estimación de lo que tardará en realizarse el sistema.

La estimación suele ser una parte complicada, especialmente en equipos grandes, puesto que implica llegar a un acuerdo. Para ello existen métodos como el *planning poker* [71], que tratan de agilizar toda esta fase de estimación. En este caso, como el proyecto lo ha llevado una persona, no ha hecho falta ningún método de este tipo, puesto que no se ha tenido que llegar a ningún tipo de acuerdo.

Otro punto de fricción en toda esta fase de estimación es la unidad de medida a usar. Un día de trabajo no tiene por qué significar lo mismo para dos desarrolladores distintos, y además no se saben los imprevistos que pueden surgir. Es por ello que se suele preferir usar una nueva unidad, los *puntos de historia* [72], que tratan de medir el esfuerzo relativo entre las historias de usuario.

La estimación de las historias de usuario, en puntos de historia, se muestra a continuación:

Código	**Nombre de la historia de usuario**	**Estimación**
HU01	Pantalla de carga	1 PH
HU02	Pantalla principal (no autenticado)	2 PH
HU03	Pantalla principal (autenticado)	2 PH
HU04	Pantalla principal (más información)	1 PH
HU05	Menú (no autenticado)	2 PH
HU06	Menú (autenticado)	2 PH
HU07	Listado de noticias	5 PH
HU08	Noticia en detalle	3 PH
HU09	Sección de la agenda	13 PH
HU10	Detalles de un evento de la agenda	2 PH
HU11	Sección de ponentes	5 PH
HU12	Detalles de un ponente	2 PH
HU13	Sección de documentos (no autenticado)	2 PH
HU14	Sección de documentos del evento (autenticado)	3 PH
HU15	Sección de mapas	3 PH
HU16	Detalles de un mapa	8 PH
HU17	Sección de servicios cercanos al congreso (listado)	2 PH
HU18	Sección de servicios cercanos al congreso (mapa)	8 PH
HU19	Mostrar un servicio en el mapa	2 PH
HU20	Detalles de un servicio	2 PH
HU21	Sección de patrocinadores	1 PH
HU22	Sección 'Acerca de'	2 PH
HU23	Sección de aspectos legales	1 PH
HU24	Retroalimentación a los organizadores	3 PH
HU25	Sección de networking (sin activar)	1 PH
HU26	Sección de networking (activa)	13 PH
HU27	Listado de contactos	5 PH
HU28	Pantalla de gestión de un emparejamiento	5 PH
HU29	Sección de preguntas al ponente	13 PH
HU30	Detalles de la pregunta a un ponente	5 PH
HU31	Crear una pregunta para un ponente	3 PH
HU32	Comentar una pregunta a un ponente	3 PH
HU33	Sección de notificaciones (no autenticado)	3 PH

Cuadro 5.3: Listado de historias de usuario con la estimación. (I)

Código	**Nombre de la historia de usuario**	**Estimación**
HU34	Sección de notificaciones (autenticado)	3 PH
HU35	Sección de realidad aumentada (no autenticado)	1 PH
HU36	Sección de realidad aumentada (autenticado)	13 PH
HU37	Escaneo de un objeto de realidad aumentada	8 PH
HU38	Ver los datos de inscripción	2 PH
HU39	Inscripción por medio de linkedIn	8 PH
HU40	Formulario de inscripción	13 PH
HU41	Ver mi acreditación	8 PH
HU42	Ver mi perfil	8 PH
HU43	Editar mi perfil	5 PH
	Total	197 PH

Cuadro 5.4: Listado de historias de usuario con la estimación. (II)

Capítulo 6

Análisis y diseño del sistema

En este capítulo se detalla toda la fase de análisis del sistema, detallando la arquitectura y mostrando una serie de diagramas que muestran cómo se ha ideado el sistema.

6.1. Análisis del sistema

La fase de análisis del sistema trata de dar solución a los problemas definidos en los requisitos, especificando lo que debe hacer el sistema y su estructura. Para especificar esta estructura se suelen emplear diagramas UML, como el diagrama de clases, por ejemplo.

Cabe destacar que en metodologías clásicas todo esto se realiza antes de empezar con el desarrollo, en las metodologías ágiles no deja de hacerse, sino que se realiza por partes, según se vaya necesitando.

Se expone a continuación el diagrama de clases de toda la fase de análisis, donde se expone la estructura que tiene la solución software que se ha desarrollado. Se expondrá únicamente la parte cliente, la central en este proyecto, puesto que el servidor se ha ido desarrollando en segundo plano bajo demanda.

6.1.1. Diagrama de clases

Antes de exponer el diagrama de clases, y para entender por qué se ha escogido la estructura que se ha creado, hay que tener en cuenta una de las características de Angular, que es la implementación que tiene del patrón de diseño *inyección de dependencias* [73].

Hace posible usarlo de dos formas:

- Global
- Local

Si es global, hay que definir la dependencia en el módulo que contiene a la clase que requiere la dependencia, que hace uso del patrón de diseño *singleton* [74] para crear una instancia única que será inyectada por Angular en todas aquellas clases, dentro del módulo, que incluyan la clase de la dependencia en su constructor.

Si es local, el funcionamiento es similar, solo que en lugar de definir la dependencia en el módulo, hay que hacerlo en el propio componente, dentro de la etiqueta *@Compenent*, usando el elemento *provider*. Haciéndolo de esta forma se creará una nueva instancia de la dependencia para ese componente.

Además, las dependencias deberán emplear la etiqueta *@Injectable()* para poder ser usadas de esta forma.

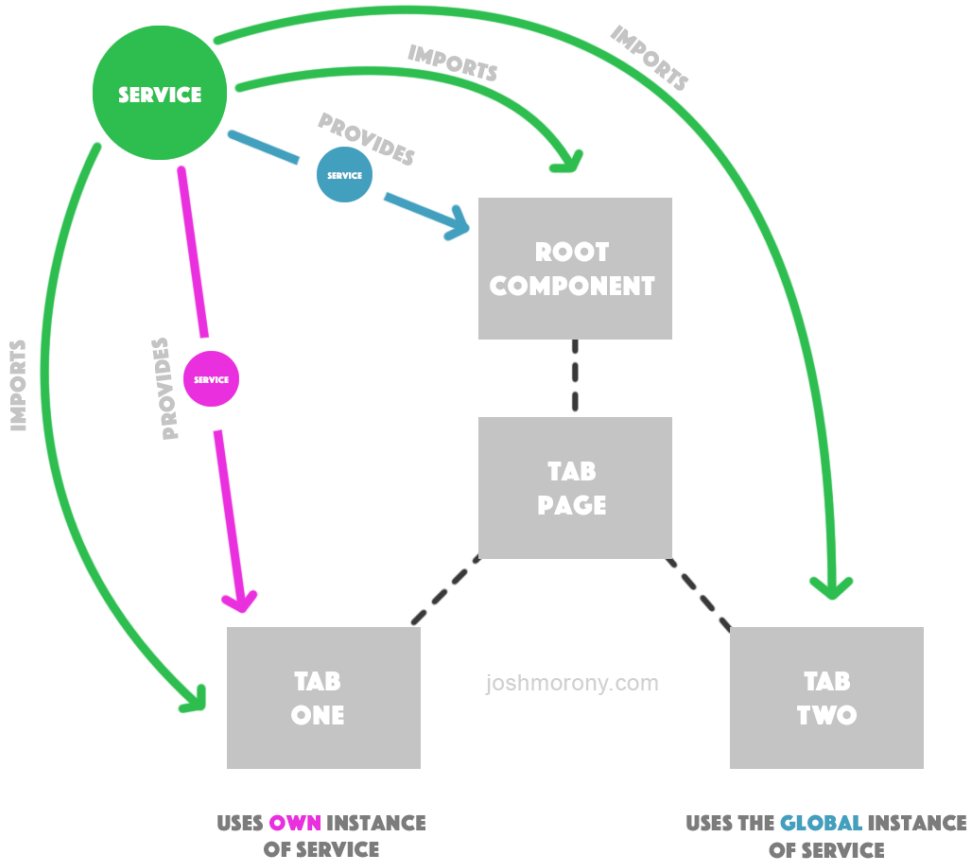

Figura 6.1: Funcionamiento de la inyección de dependencias en Angular e Ionic 2. Fuente [6]

Es, por ello, que la aplicación se divide en tres partes, como se puede ver en un diagrama inicial, en la figura 6.2.

Los componentes hacen uso de los servicios, que obtienen por medio de la inyección de dependencias, y de los modelos. Los modelos se limitan a almacenar información, los servicios

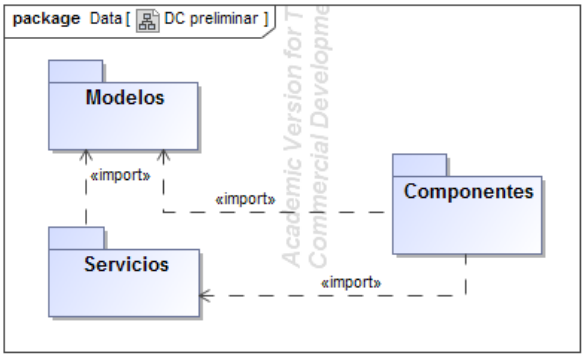

Figura 6.2: Diagrama de la aplicación a nivel de paquetes.

realizan tareas concretas y son reutilizables allá donde sean necesarios.

Para añadir un nuevo componente basta con crearlo y crear también los modelos y servicios que necesite, si los necesita, el único caso en el que se necesita modificar algo es cuando se crea un servicio, hay que añadirlo al módulo si se quiere poder inyectarlo.

El diagrama de clases completo se muestra en la figura 6.3.

Por simplicidad, todas las páginas que están incluidas dentro de una sección se han incluido con el formato: *ComponenteSection*, fundamentalmente porque no aportan información al diagrama.

Lo importante es tener en cuenta que todas las páginas que forman una sección están agrupadas dentro de un módulo si emplean algún tipo de servicio. Esto se ha hecho así para poder aprovechar el *lazy loading* que ofrece Ionic a partir de su versión 3.

Hay algunas secciones que no están representadas de esta forma, dado que no usan ninguno de estos componentes, por simplicidad de cara al diseño se ha decidido incluirlas dentro del *MainModule*.

Por último, las clases *HttpService* y *DBService* son servicios que ya ofrecen Angular e Ionic, lo que facilita mucho las cosas, puesto que basta con usarlos y ahorra mucho tiempo de desarrollo.

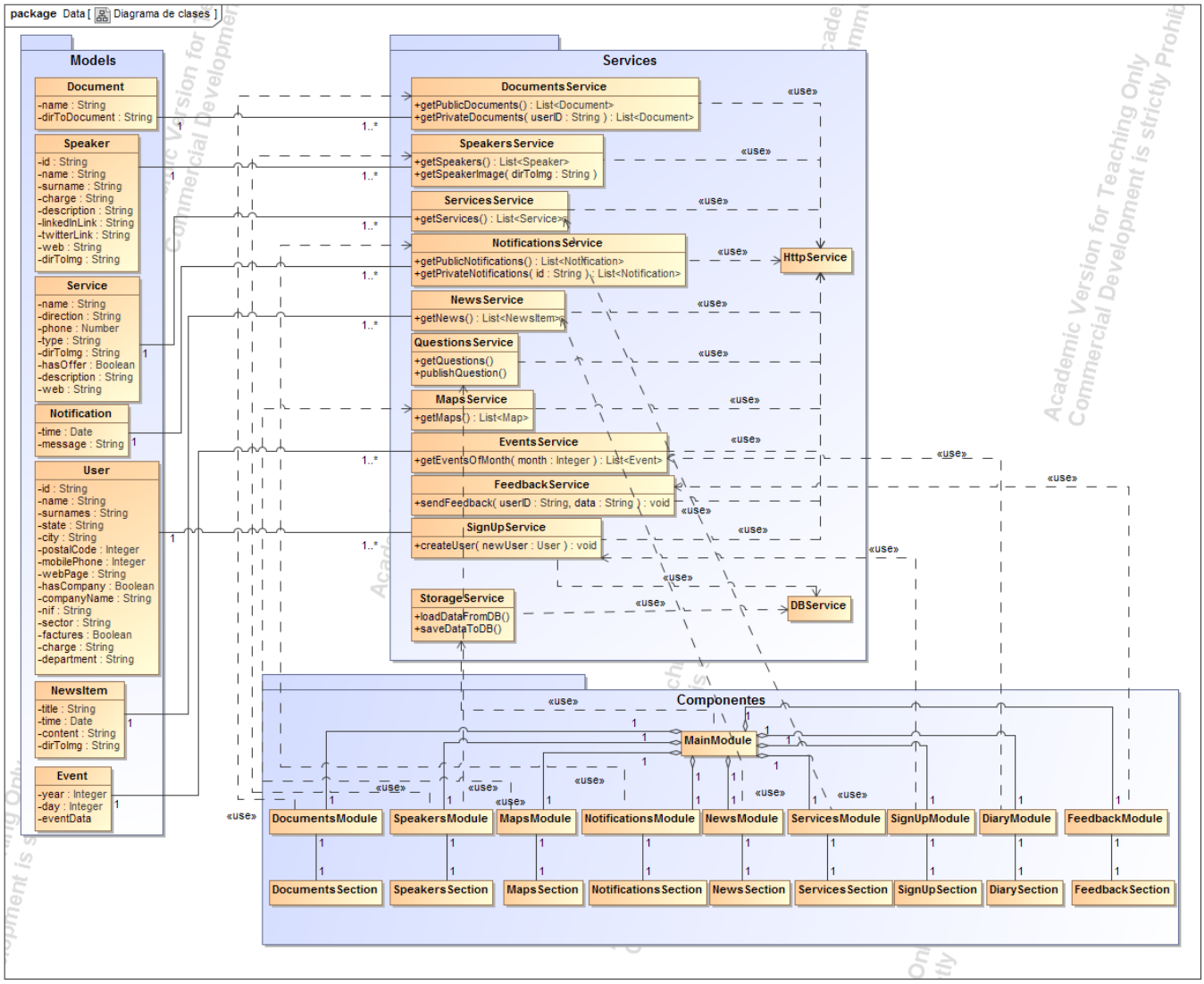

Figura 6.3: Diagrama de clases de la aplicación.

6.2. Diseño de la arquitectura del sistema

En esta sección se va a incluir la arquitectura del sistema, previa exposición del tipo de arquitectura que se está usando.

Se expondrá la arquitectura del sistema a alto nivel, con un diagrama del sistema completo y se detallará la estructura de la base de datos.

6.2.1. Arquitectura cliente-servidor

El paradigma *cliente-servidor* [75] es uno de los más conocidos y utilizados hoy en día, basado en una estructura centralizada, normalmente pasiva (servidor) que ofrece respuestas (servicio) a las peticiones que le llegan por parte de entes activos, que comienzan la interacción (clientes).

Una representación puede verse en la figura 6.4.

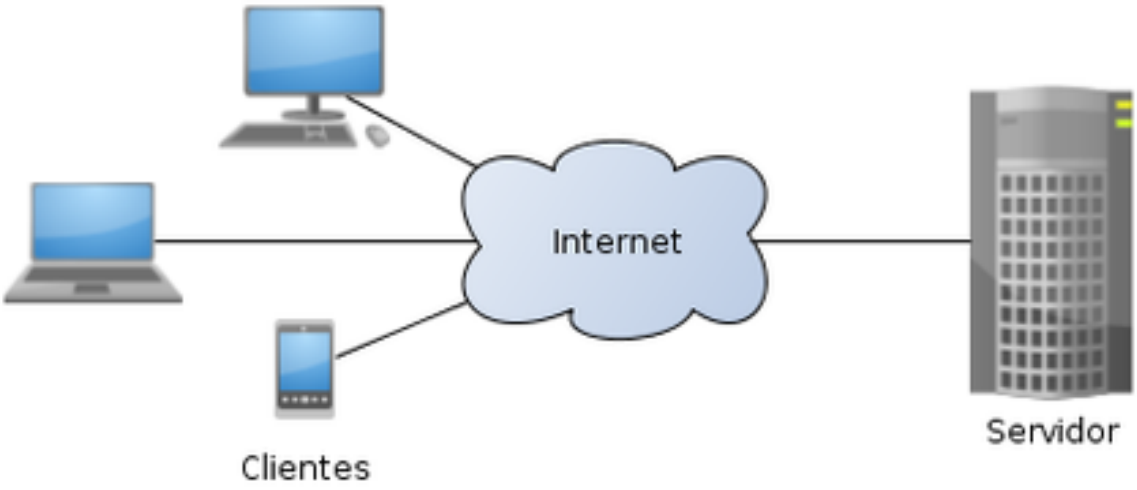

Figura 6.4: Diagrama de la arquitectura cliente-servidor. Fuente [7]

Aplicado al proyecto, la parte cliente se corresponde con los clientes, que demandan recursos/servicios de la parte servidor, encargada de resolver estas peticiones y devolver la información correspondiente.

6.2.2. Arquitectura del sistema a alto nivel

La arquitectura a alto nivel, que resume el funcionamiento de todo el sistema, puede verse en la figura 6.5.

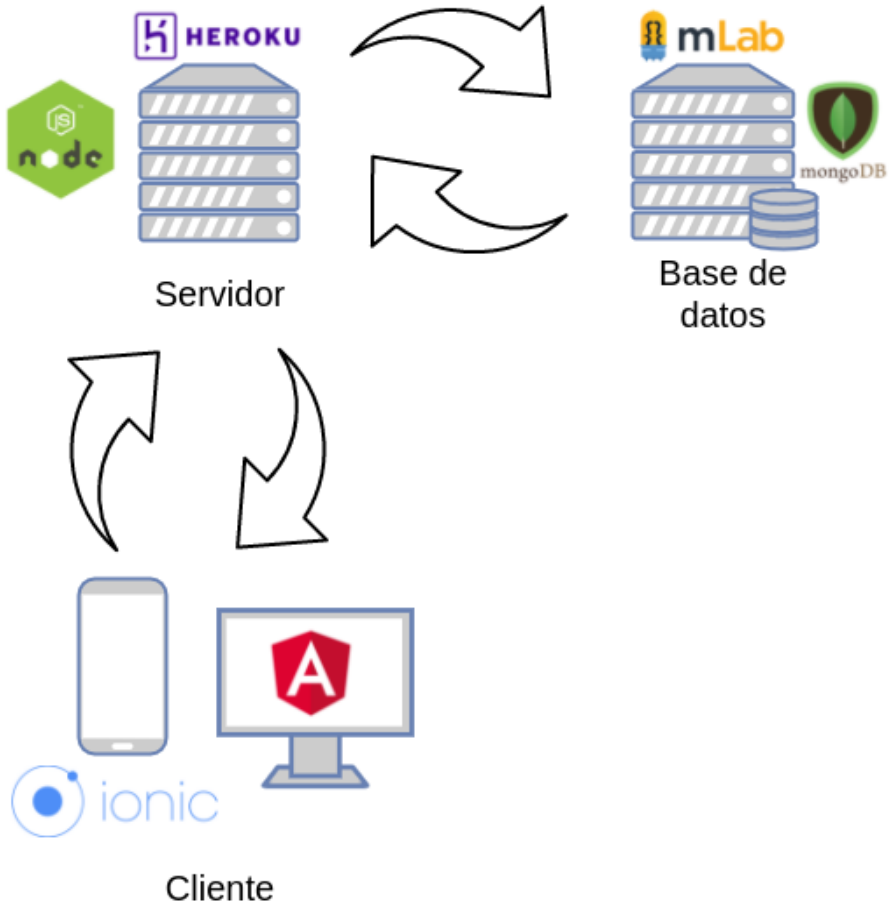

Figura 6.5: Diagrama de la arquitectura del sistema a alto nivel.

El servidor y la base de datos están operando en servidores remotos, en Heroku y mLab, respectivamente. En el caso de la base de datos, el servidor abre una conexión con la base de datos cuando es necesario.

El hecho de que la base de datos sea remota facilita mucho el poder replicar tanto el servidor como la base de datos para prevenir caídas o escalarlos mediante un *sistema distribuido* [76]. Si estuviesen juntos, al caer el servidor también caería la base de datos. Manteniéndolos separados pueden mantenerse réplicas del servidor y la base de datos en lugares distintos.

6.2.3. Diseño de la base de datos

Siguiendo la sintaxis de MongoDB, la base de datos consta de una serie de colecciones (un equivalente a las tablas en el modelo relacional).

Dentro de cada colección se guarda lo que se llaman documentos (registros en el modelo relacional), que tienen la estructura de documentos JSON.

En el caso del proyecto se ha mantenido un diseño muy minimalista. Se muestra a continuación las colecciones de la base de datos y qué se guarda en cada una.

- **Documentos**
  - id
  - tipo
  - documento
- **Eventos**
  - año
  - mes
  - día
  - evento
- **Feedbacks**
  - id
  - feedback
  - id_usuario
  - fecha_creación
- **Mapas**
  - nombre
  - miniatura
  - tipo
  - nombre_dirección
  - dirección
  - latitud
  - longitud
  - recurso (opcional)
- **Noticias**
  - título
  - contenido
  - imagen

    - fecha_creación
- **Notificaciones**
    - fecha_creación
    - mensaje
    - tipo
    - id_usuario (opcional)
- **Preguntas**
    - id
    - pregunta
    - id_usuario
    - id_ponente
    - fecha_creación
- **Servicios**
    - nombre
    - dirección
    - teléfono
    - tipo
    - imagen
    - tiene_oferta
    - descripción
    - web
    - latitud
    - longitud
- **Ponentes**
    - id
    - nombre
    - apellidos
    - cargo
    - descripción
    - dir_linkedin
    - dir_twitter
    - dir_web
    - imagen
- **Usuarios**
    - uuid_dispositivo

- id
- nombre
- apellidos
- país
- comunidad
- ciudad
- codigo_postal
- teléfono
- web (opcional)
- nombre_compañía (opcional)
- nif (opcional)
- sector (opcional)
- puesto (opcional)
- departamento (opcional)
- factura (opcional)

6.3. Diseño de la interfaz

El diseño de la interfaz fue proporcionado por el cliente, estaba ya validado en contenido (incluía todo lo que quería el cliente) y en forma (lo realizó una diseñadora profesional, por lo que no había ningún tipo de inconsistencia), por tanto, no ha habido una fase de diseño de *mockups*.

A continuación se adjuntan algunas de las pantallas que se proporcionaron, las que considero que aportan más información, en las figuras 6.6, 6.7, 6.8, 6.9, 6.10, 6.11, 6.12:

Figura 6.6: Diseño proporcionado para la pantalla de carga.

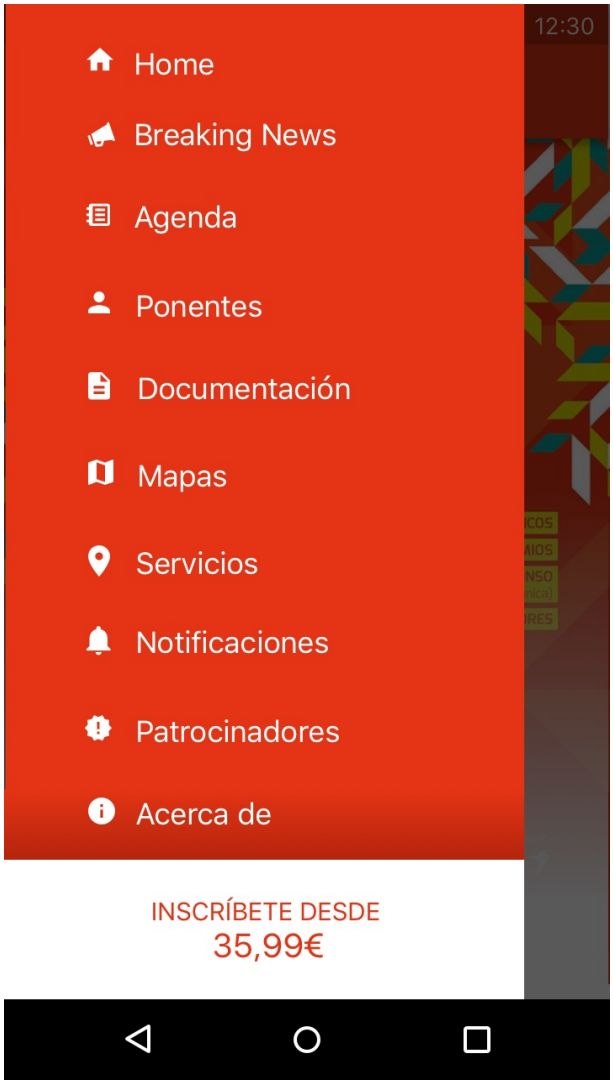

Figura 6.7: Diseño proporcionado para el menú sin un usuario autenticado.

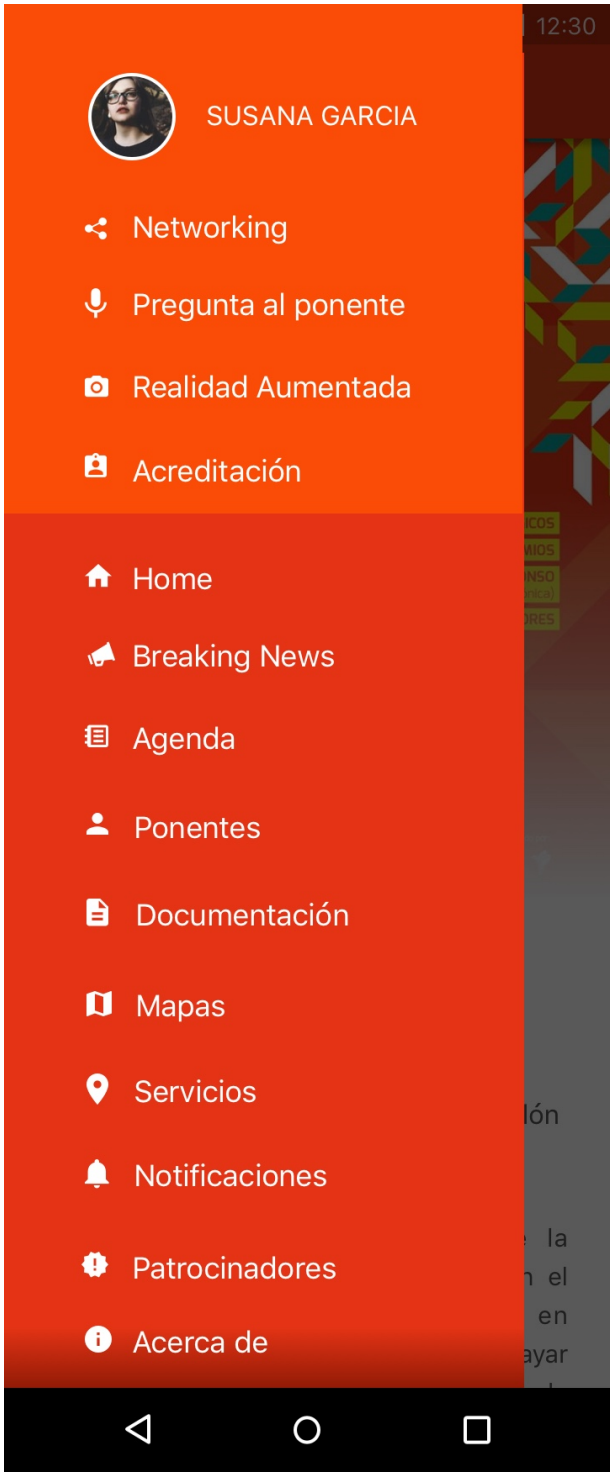

Figura 6.8: Diseño proporcionado para el menú con un usuario autenticado.

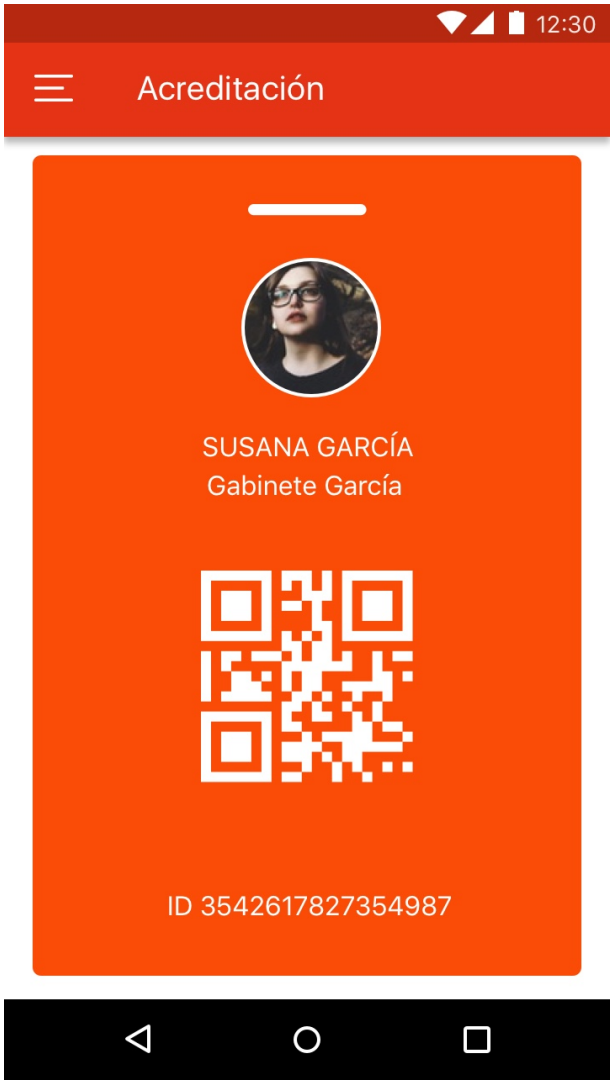

Figura 6.9: Diseño proporcionado para la acreditación.

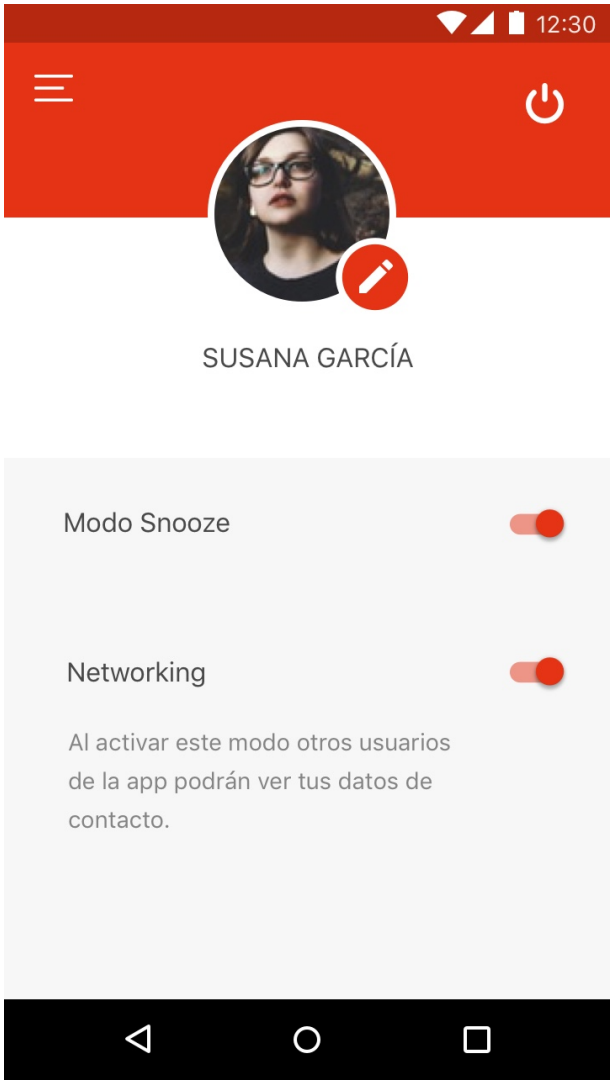

Figura 6.10: Diseño proporcionado para el perfil.

Figura 6.11: Diseño proporcionado para la agenda con el calendario desplegado.

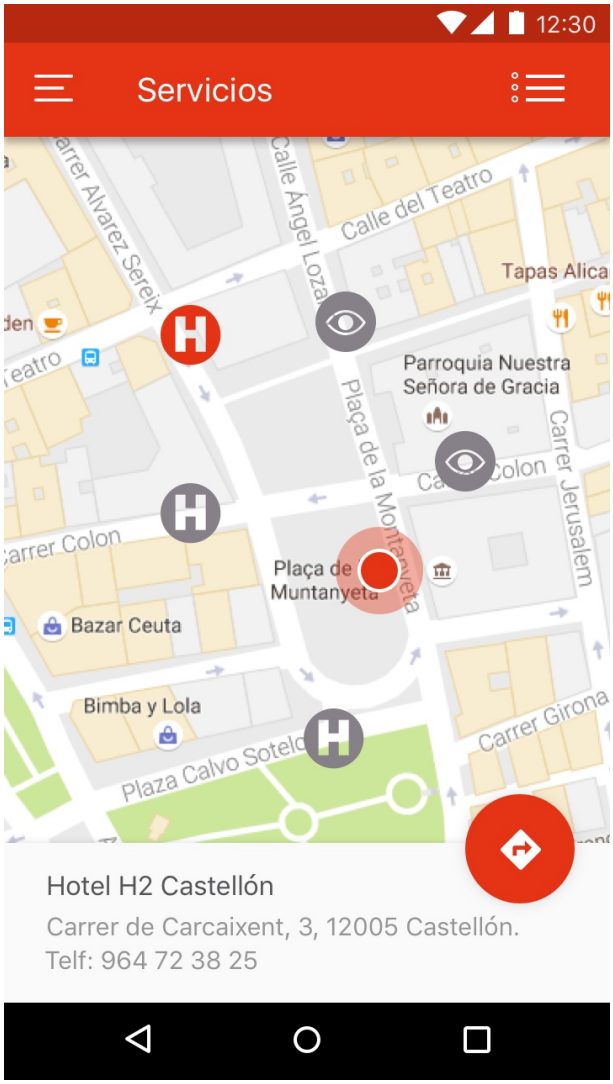

Figura 6.12: Diseño proporcionado para los servicios seleccionados en el mapa.

Capítulo 7

Implementación, pruebas y despliegue

En este capítulo se detalla, paso a paso, el proceso de implementación del proyecto, y todo el proceso de pruebas que se ha ido realizando a lo largo del mismo y los resultados que se han ido obteniendo.

7.1. Implementación

Se introduce en esta sección la implementación del proyecto, por fases, correspondiendo cada fase a uno de los sprints. Se concluye con un apartado para el despliegue, donde se detalla cómo se ha realizado.

7.1.1. Fases de implementación

En el capítulo 5 se detallaba la estimación, en puntos de historia, de todas las historias de usuario de la aplicación, dando un total de 197 PH. Esto hace una media de 39,4 PH por sprint. Se ha intentado ajustar la carga de trabajo de cada sprint a esta media.

Además de exponer detalles de la implementación, puesto que los sprints comprenden la creación, fundamentalmente, de la parte cliente, que es una aplicación gráfica, también se mostrarán imágenes de los resultados que se han obtenido.

7.1.1.1. Sprint 1

En el sprint 1 se han realizado las siguientes historias de usuario:

- HU05 - Menú (no autenticado)

- HU06 - Menú (autenticado)
- HU02 - Pantalla principal (no autenticado)
- HU03 - Pantalla principal (autenticado)
- HU04 - Pantalla principal (más información)
- HU01 - Pantalla de carga
- HU40 - Formulario de inscripción
- HU42 - Ver mi perfil

7.1.1.1.1. Detalles técnicos del Sprint 1

Ionic proporciona toda una serie de componentes con un funcionamiento muy similar al que se obtiene en un dispositivo móvil, entre ellos, el del menú, *ion-menu* [77].

En este sentido, el reto ha sido crear los estilos, en CSS, puesto que se tenía que conseguir un resultado lo más parecido posible al de los diseños.

Por otra parte, se tiene la pantalla principal. Ionic cuenta con una herramienta *cli* [78] (Command-line Interface) que permite crear proyectos y componentes dentro de un proyecto con un único comando, lo que puede llegar a ahorrar cantidades ingentes de tiempo. Al generar un proyecto con esta herramienta, se genera una plantilla de aplicación que es ya una versión funcional, por lo que para poner a punto la pantalla principal ha bastado únicamente con editar la que venía por defecto. Al igual que con el menú, el reto ha estado en el uso del CSS para conseguir un diseño idéntico al que se pedía.

La pantalla de carga es una funcionalidad que proporciona Cordova, por lo que se ha tenido que modificar el fichero de configuración (*config.xml)* y añadir los ficheros necesarios en la carpeta *assets* del proyecto. La *cli* de Ionic proporciona un comando (*ionic resources)* genera los *assets* a todos los tamaños necesarios, algo muy cómodo, por lo que añadir la pantalla de carga ha sido bastante rápido.

Para crear el formulario de inscripción se ha aprovechado el módulo que proporciona Angular para crear formularios (*@angular/forms*), con el que se puede realizar *data-binding* muy fácilmente, y añadir validadores a cada campo como métodos independientes, lo que hace muy fácil crear test para probarlos.

Por último, para crear el perfil, la complicación ha estado en adaptar la barra de navegación para que ocupase el espacio que tenía que ocupar. Se ha hecho todo con CSS, como en el resto de componentes, incluyendo el diseño para la foto de perfil.

Los resultados de cada parte del *sprint* pueden verse en las figuras 7.1, 7.2, 7.3, 7.4.

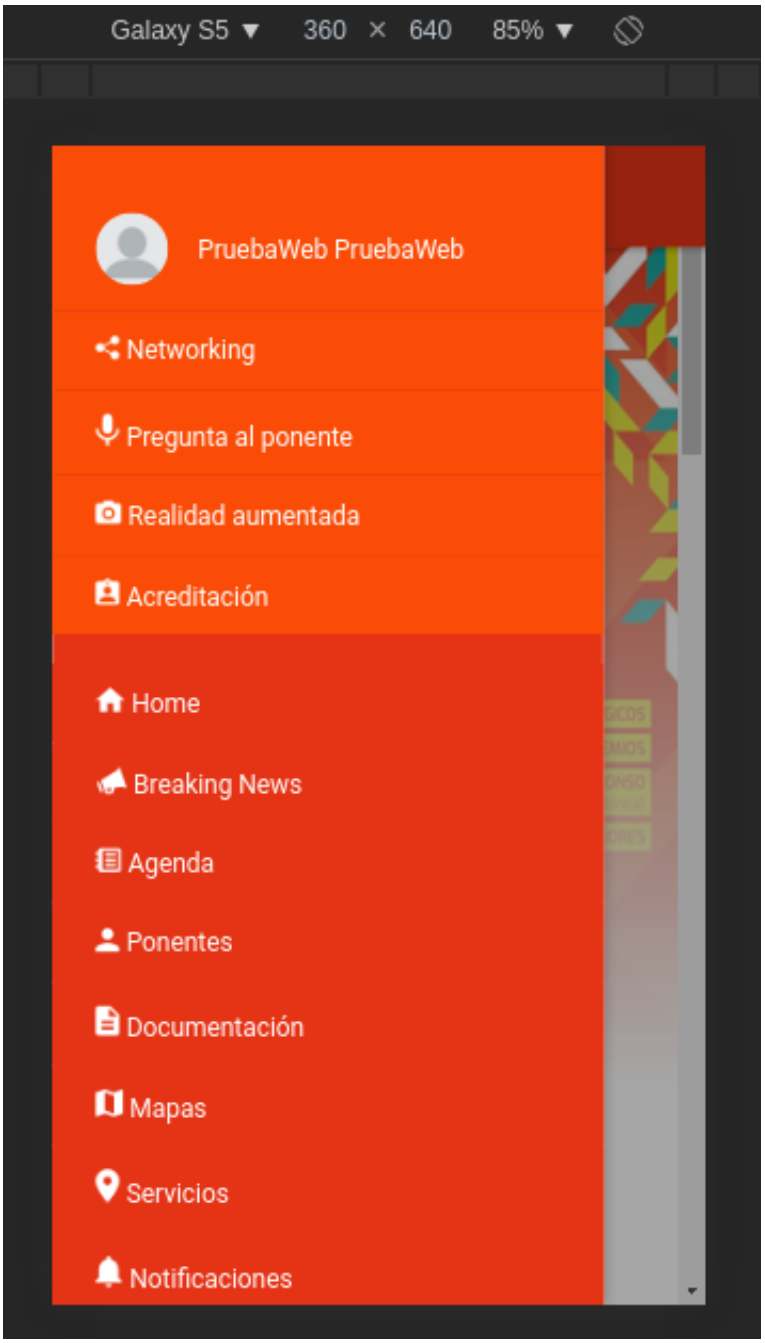

Figura 7.1: Implementación final del menú para un usuario autenticado.

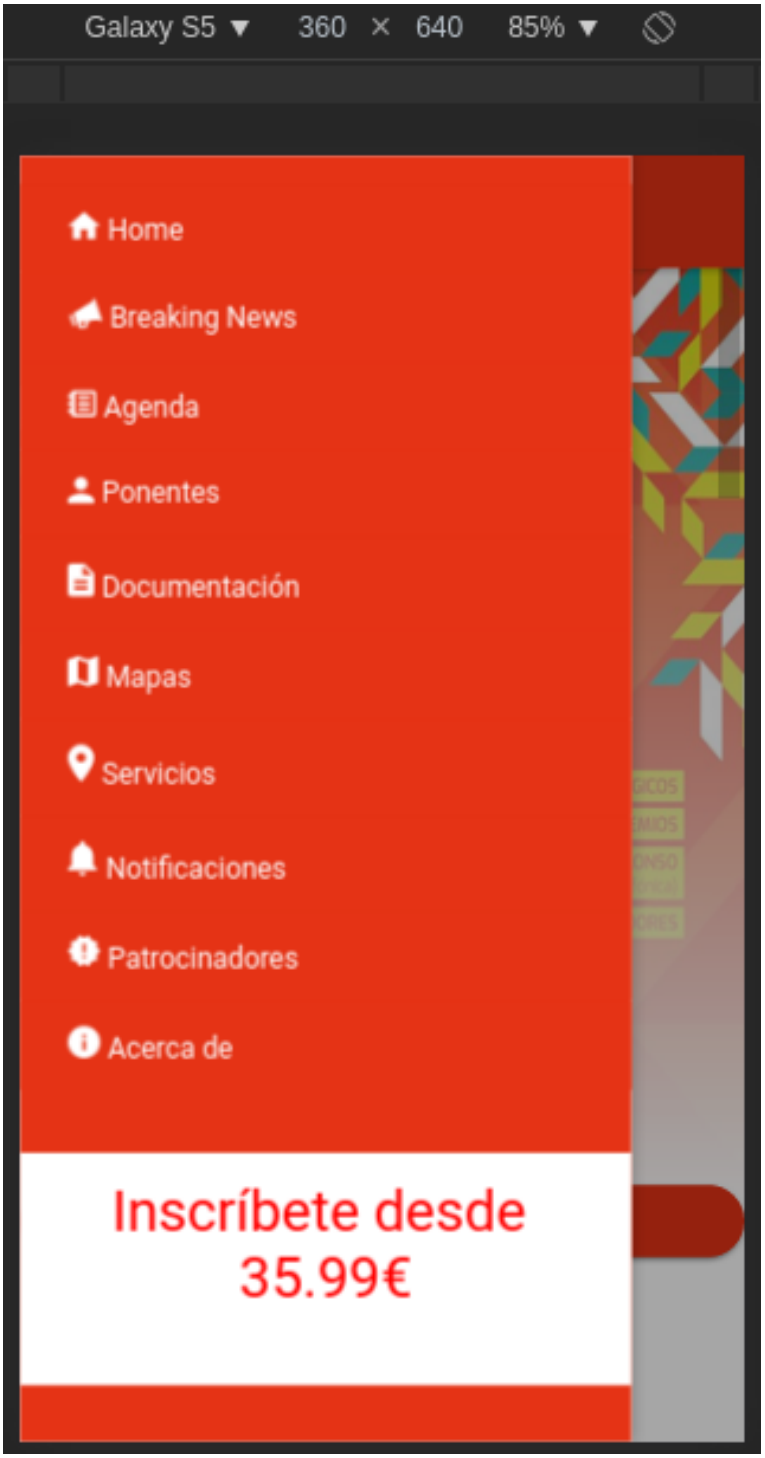

Figura 7.2: Implementación final del menú para un usuario no autenticado.

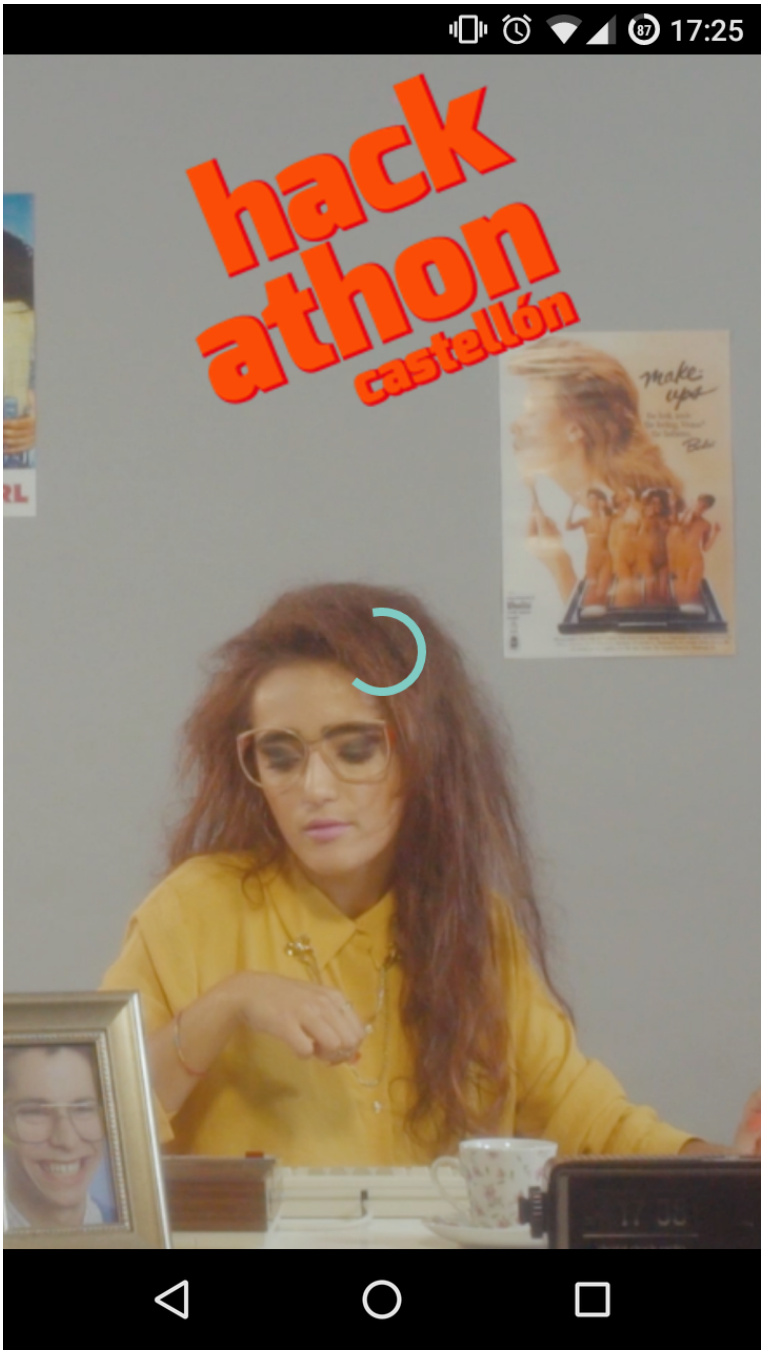

Figura 7.3: Resultado de la implementación de la pantalla de carga.

Figura 7.4: Resultado de la implementación del formulario de inscripción.

7.1.1.2. Sprint 2

En el sprint 2 se han realizado las siguientes historias de usuario:

- HU21 - Sección de patrocinadores
- HU22 - Sección 'Acerca de'
- HU23 - Sección de aspectos legales
- HU24 - Retroalimentación a los organizadores
- HU41 - Ver mi acreditación

7.1.1.2.1. Detalles técnicos del Sprint 2

La sección de la patrocinadores no ha ofrecido ningún reto a nivel de diseño, puesto que es bastante simple, consiste fundamentalmente en logos distribuidos en pantalla. El reto aquí ha estado en la interacción: cuando se pulsa sobre cada uno de los logos ha de abrirse el navegador del dispositivo en la página web a la que el logo enlaza. Este detalle, que es muy simple en el desarrollo nativo, en Ionic se complica un poco más.

La solución, actualmente, se proporciona en *ionic-native*, uno de los módulos que forman Ionic, concretamente con la herramienta *In App Browser* [79], que permite abrir el navegador que el dispositivo esté usando.

La parte de retroalimentación conecta con el servidor para guardar la retroalimentación que se hace. Aquí el reto vino al tener que aprender a usar los objetos *Observable*, necesarios para poder gestionar las llamadas al servidor empleando el servicio *Http* de Angular. Estos objetos son parte de la librería *RxJS*, implementación de la programación reactiva.

En este instante del desarrollo todavía no se tenía la parte servidor a punto, por lo que se decidió dejarla preparada para tener que cambiar únicamente la configuración a posteriori, ya que construir el servidor quedaba fuera del sprint en esos momentos.

La sección de la acreditación emplea una librería para generar códigos QR, se usa para obtener un código QR con el identificador del usuario y mostrarlo.

El resultado del sprint 2 puede verse en las figuras 7.5, 7.6, 7.7 y 7.8.

Figura 7.5: Resultado de la implementación de la sección de patrocinadores.

Figura 7.6: Resultado de la implementación de la sección 'Acerca de'.

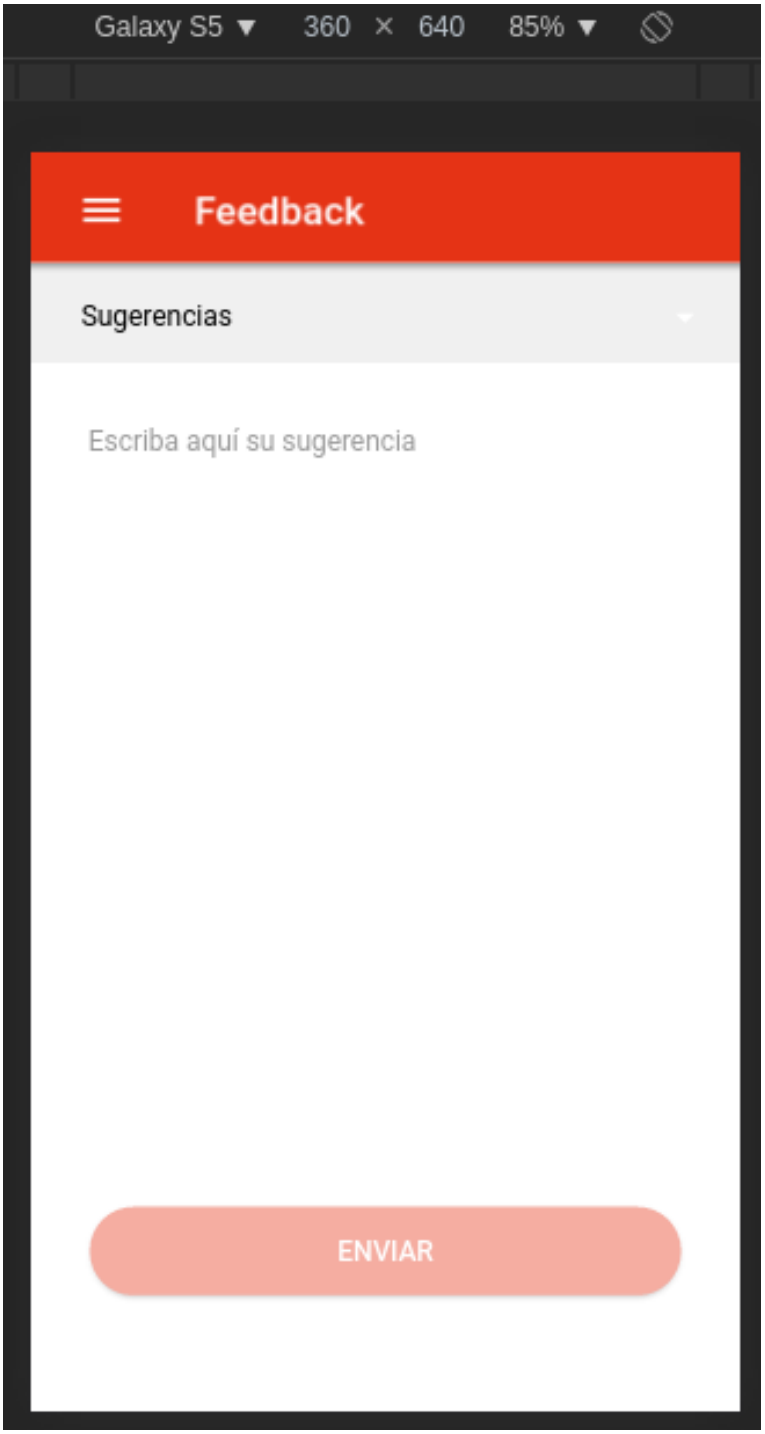

Figura 7.7: Resultado de la implementación del formulario de retroalimentación.

Figura 7.8: Resultado de la implementación de la acreditación.

7.1.1.3. Sprint 3

En el sprint 3 se han realizado las siguientes historias de usuario:

- HU07 - Listado de noticias
- HU08 - Noticia en detalle
- HU11 - Sección de ponentes
- HU12 - Detalles de un ponente
- HU13 - Sección de documentos (no autenticado)
- HU14 - Sección de documentos (autenticado)
- HU15 - Sección de mapas
- HU17 - Sección de servicios cercanos al congreso (listado)
- HU20 - Detalles de un servicio
- HU25 - Sección de networking (sin activar)
- HU29 - Sección de preguntas al ponente
- HU34 - Sección de notificaciones (autenticado)
- HU35 - Sección de realidad aumentada (no autenticado)

7.1.1.3.1. Detalles técnicos del Sprint 3

La creación de la sección de noticias a nivel de diseño ha tenido bastante complejidad. Algo a destacar en este punto, que pueda ser de interés es el uso de ventanas modales para mostrar los datos de una noticia, bastante fácil gracias al controlador que ofrece Ionic para realizar la navegación entre páginas (*NavController* [80]), que tiene un funcionamiento similar al de una pila de datos, con métodos como *push*, que inserta una página en la parte superior, *pop*, que retira la página superior de la pila para acceder a la siguiente, y *setRoot*, que elimina la pila de páginas para mostrar la página que se inserta.

En cuanto a la sección de ponentes, su estructura es prácticamente idéntica a la de las noticias, cambiando, eso sí, la orientación del *scroll*.

El diseño de la sección de documentos no ha supuesto un reto por su complejidad. Cabe destacar que, para abrir el documento, se ha optado por emplear el navegador. Hacerlo en un lector de pdf era bastante complejo desde Ionic, mientras que, haciéndolo desde el navegador implicaba usar exactamente la misma estructura que se había empleado en la sección de patrocinadores sabiendo que puede funcionar en cualquier dispositivo.

En cuanto a la sección de mapas y servicios, se ha implementado únicamente el listado, y, en el caso de los servicios, la ventana modal para mostrar los detalles.

Para la sección de preguntas al ponente se ha utilizado una estructura muy similar a la usada

para enviar la retroalimentación, al igual que para la sección de notificaciones se ha empleado una estructura como la de la sección de documentos. En este sentido no ha habido ninguna novedad a nivel técnico.

Todas estas partes se han probado utilizando *mocks* y se han preparado para ser fácilmente adaptables cuando el servidor esté listo.

Los resultados pueden verse en las figuras 7.9, 7.10, 7.11, 7.12, 7.13 y 7.14.

Figura 7.9: Resultado de la implementación de la sección de noticias.

Figura 7.10: Resultado de la implementación del modal de noticias.

Figura 7.11: Resultado de la implementación de la sección de ponentes.

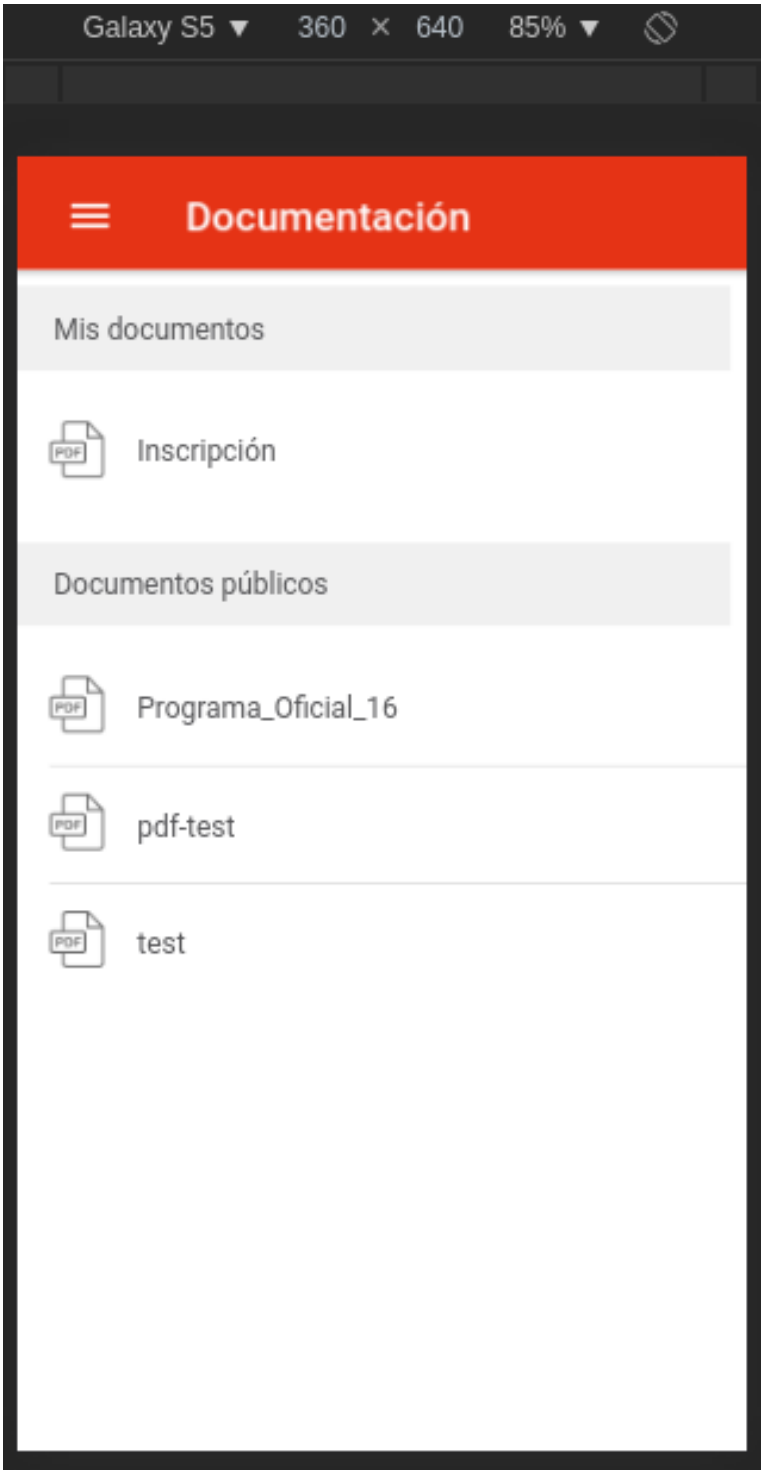

Figura 7.12: Resultado de la implementación de la sección de documentos.

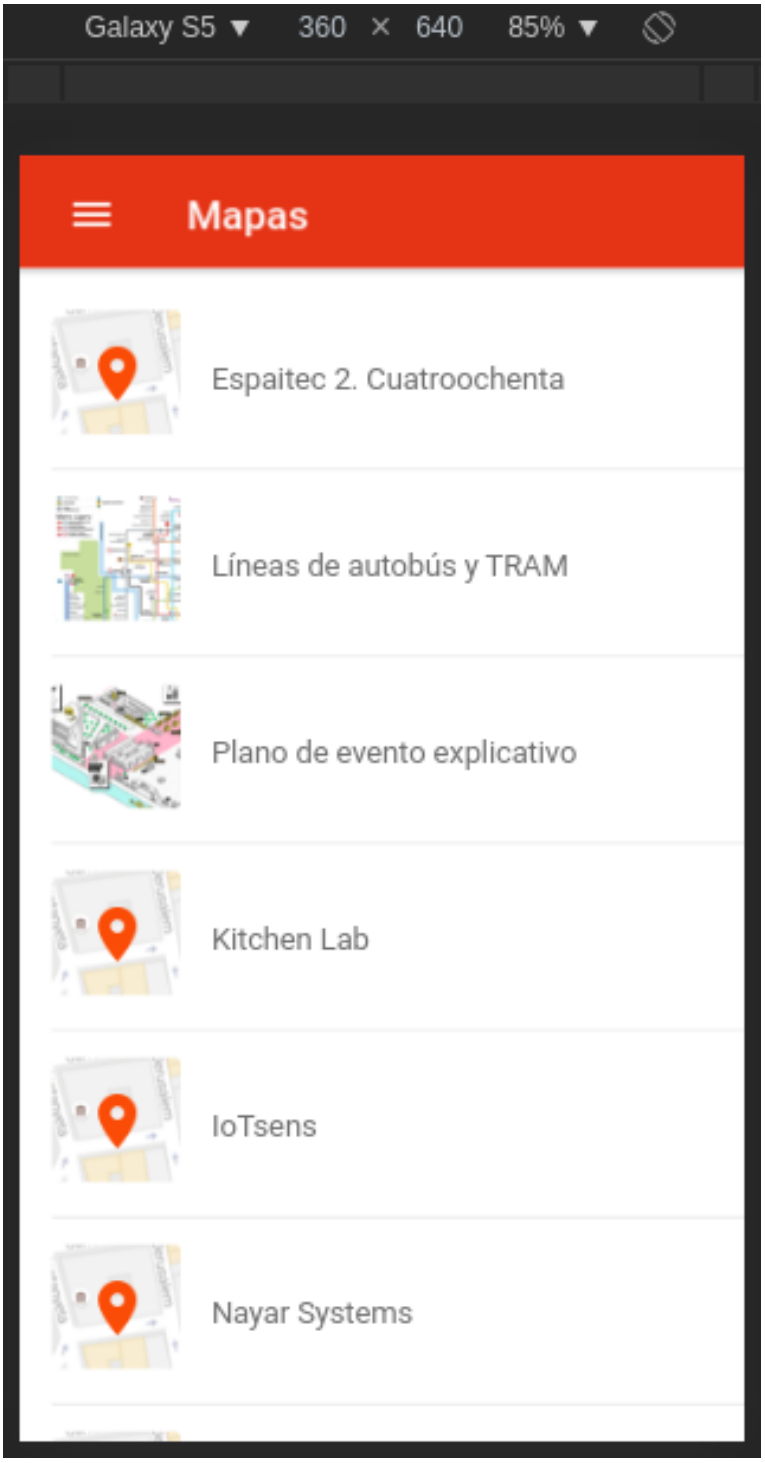

Figura 7.13: Resultado de la implementación del listado de mapas.

Figura 7.14: Resultado de la implementación del modal de servicios.

7.1.1.4. Sprint 4

El sprint 4 se ha dedicado a la construcción de la infraestructura de despliegue de la parte servidor, al despliegue de la base de datos en un servidor remoto, al despliegue y la configuración de la integración continua.

7.1.1.4.1. Detalles técnicos del Sprint 4

Se ha decidido dedicar todo un sprint únicamente a la parte servidor por la complejidad que puede conllevar su creación, a nivel técnico, y por la necesidad de documentarse sobre todas las tecnologías que se emplean.

Como ya se ha expuesto en el capítulo 3, se ha decidido emplear NodeJS para su elaboración. Hay que separar el desarrollo del servidor en tres partes:

1. Creación de la base del servidor
2. Creación del módulo para la conexión con la base de datos
3. Creación de la infraestructura REST

La base del servidor pone a punto y configura todos los módulos que emplea, como por ejemplo el del *logger* o el del *cors*, además del usado para la conexión con la base de datos y las rutas a las que atiende.

Para la conexión con la base de datos se ha creado una clase dedicada a ello, que emplea *mongoose* para realizar la conexión, esta clase se utiliza en la configuración del servidor antes de arrancar, y sólo arranca si la conexión se ha realizado con éxito. La configuración, y todo lo relacionado con la base de datos se trata en esta clase, los elementos que conecten con ella únicamente tendrán que preocuparse de realizar las consultas correspondientes.

La infraestructura REST consiste en una serie de métodos independientes. Cada ruta principal (/speakers/*, /events/*, etc...) tiene el suyo y es añadido al objeto *router* de *express* para ser usado. En este método se definen todas las sub-rutas que sean necesarias, devolviendo al final el objeto *router* con todas las rutas.

Esta infraestructura hace que, añadir nuevas rutas consista en añadir nuevos métodos, lo único que hay que modificar es la clase base donde se usan, añadiendo una línea que añada el objeto *router* devuelto por el método al objeto *router* principal.

7.1.1.5. Sprint 5

En el sprint 5 se han realizado las siguientes historias de usuario:

- HU09 - Sección de la agenda
- HU10 - Detalles de un evento de la agenda
- HU18 - Sección de servicios cercanos al congreso (mapa)
- HU19 - Mostrar un servicio en el mapa

Han quedado por hacer:

- HU26 - Sección de networking (activa)
- HU27 - Listado de contactos
- HU28 - Pantalla de gestión de un emparejamiento
- HU30 - Detalles de la pregunta a un ponente
- HU31 - Crear una pregunta para un ponente
- HU32 - Comentar una pregunta a un ponente
- HU33 - Sección de notificaciones (no autenticado)
- HU37 - Escaneo de un objeto de realidad aumentada
- HU39 - Inscripción por medio de LinkedIn

7.1.1.5.1. Detalles técnicos del Sprint 5

La creación de la agenda ha sido todo un reto a nivel técnico, por las exigencias del diseño, aunque no ha aportado nada nuevo a nivel tecnológico, todo se ha realizado con HTML y CSS, como en el resto de elementos.

No ha ocurrido lo mismo con la introducción del mapa. Habían dos opciones para incluirlo en la aplicación, o bien utilizar la adaptación que Ionic tiene implementada dentro de *@ionic/native*, o bien emplear la *API* de *Google Maps* [81], desarrollada en JavaScript, que es la usada en los navegadores.

Al final se optó por usar la *API*, puesto que daría menos problemas que la versión desarrollada en Ionic y además también podría ser usada en la versión web.

Sobre este mapa se han desarrollado el resto de elementos gráficos obteniendo los resultados que se muestran a continuación, en las figuras 7.15, 7.16, 7.17, 7.18 y 7.19.

Figura 7.15: Resultado de la implementación de la sección de la agenda.

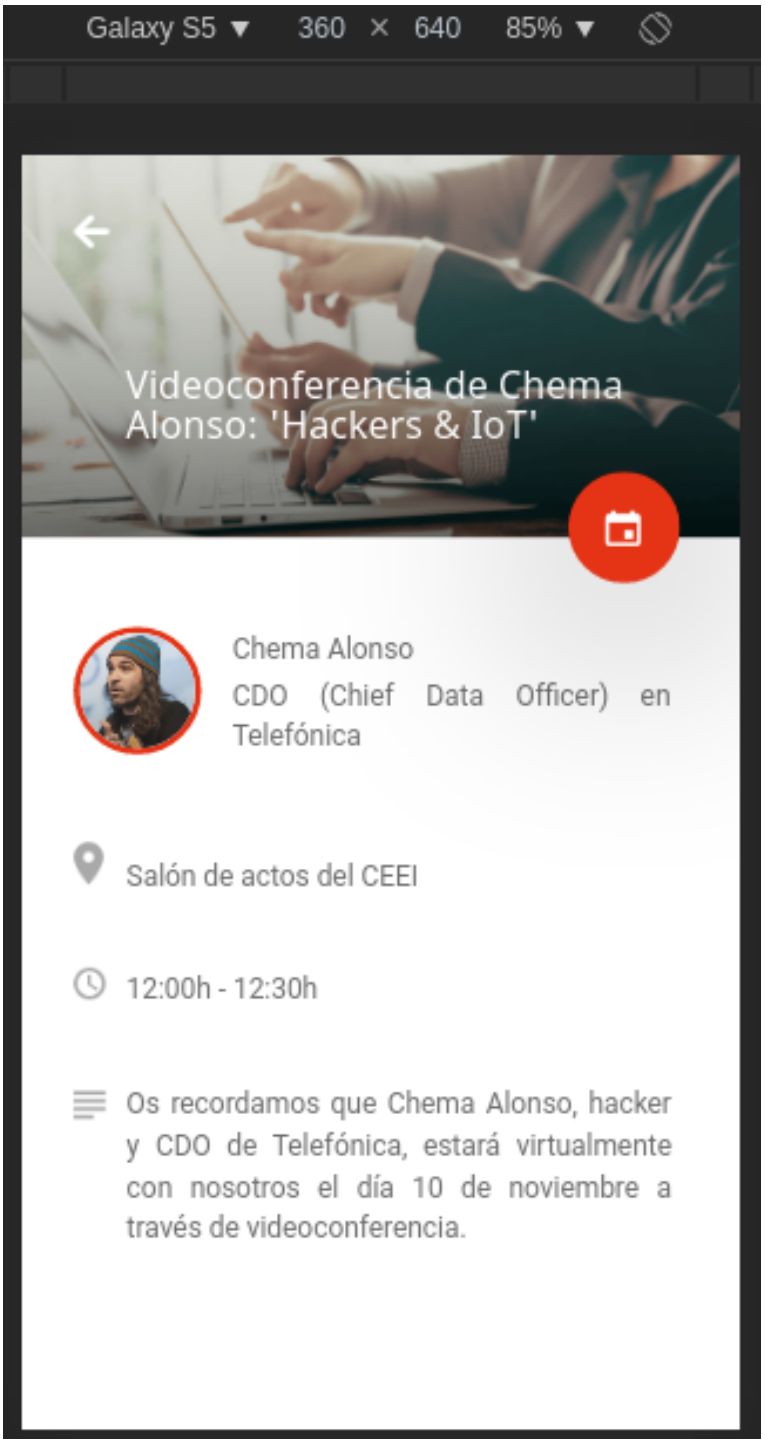

Figura 7.16: Resultado de la implementación del modal de la agenda.

Figura 7.17: Resultado de la implementación del mapa de servicios.

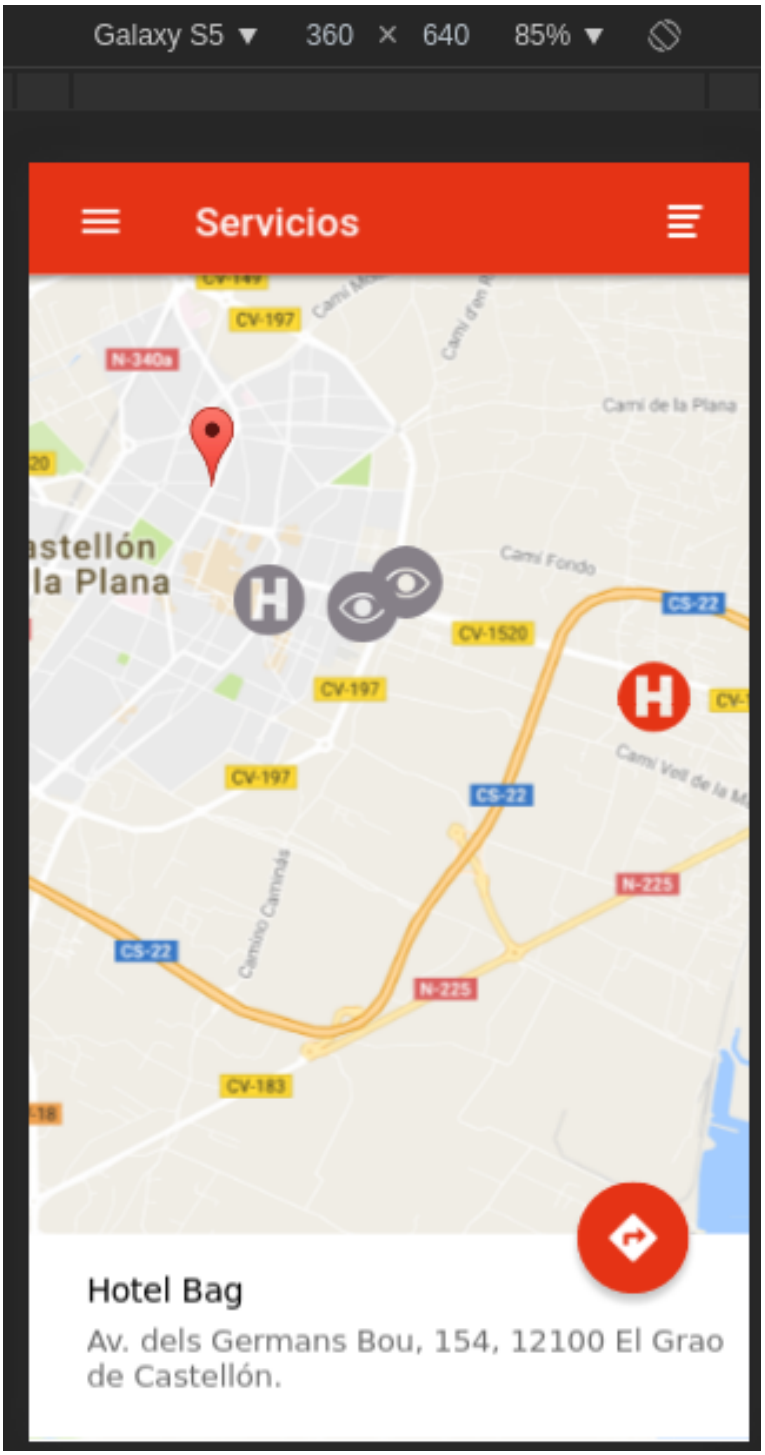

Figura 7.18: Resultado de la implementación de la selección de un servicio en el mapa.

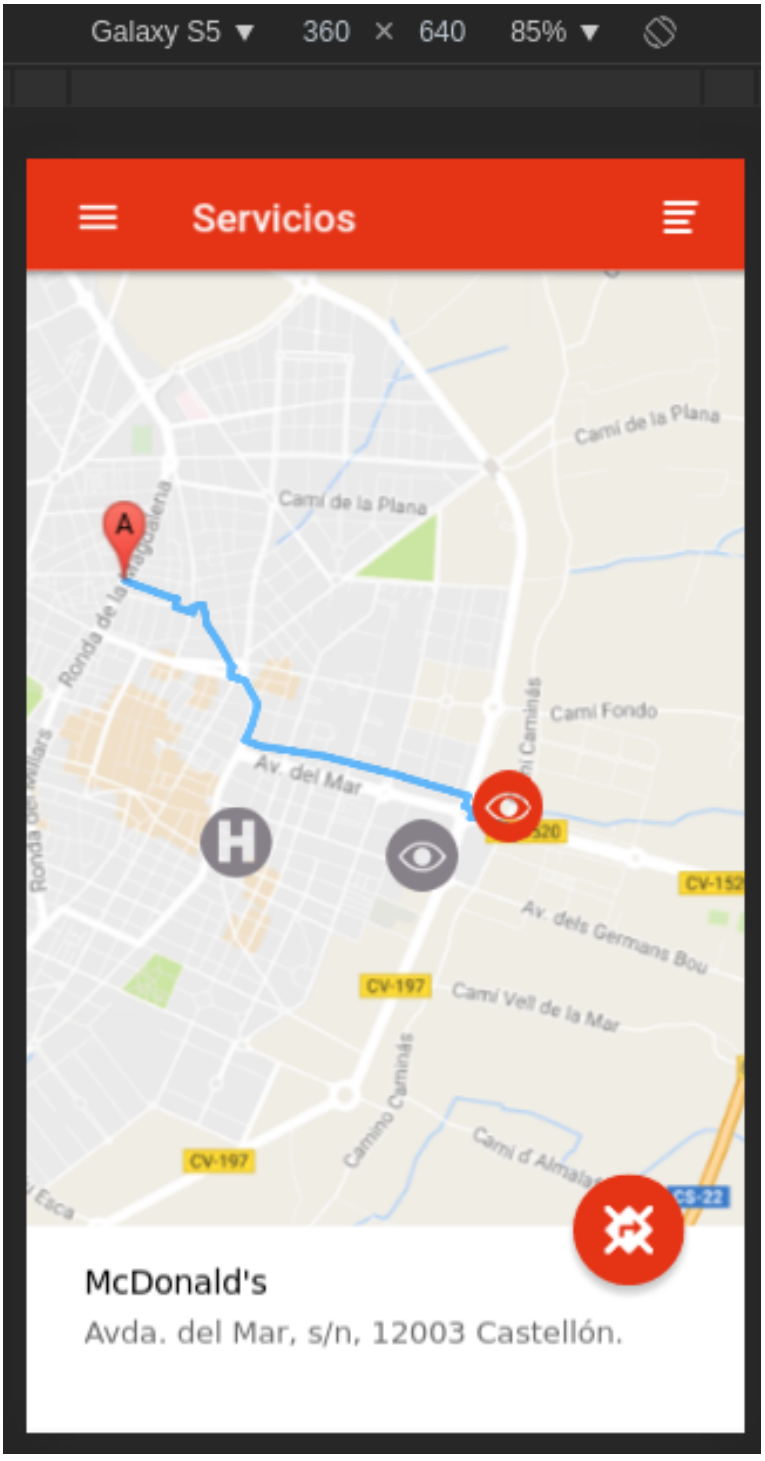

Figura 7.19: Resultado de la implementación del sistema de guiado en el mapa. La aplicación va actualizando la posición del dispositivo y renovando el camino.

7.2. Verificación y validación

En esta sección se detallan los tipos de pruebas presentes en el proyecto, cómo se han diseñado, las tecnologías que emplean, y cómo se han usado.

En primera instancia se ha tratado de seguir un desarrollo que emplease las técnicas de *ATDD* y *TDD*. Al iniciar con la planificación, se optó por preparar un entorno de desarrollo profesional para el uso de estas metodologías en un equipo de desarrollo, con fines académicos, y mantener un nivel mínimo de pruebas, fundamentalmente por la restricción temporal.

7.2.1. Pruebas unitarias

Las pruebas unitarias [82] operan sobre un componente aislado para asegurar que ese componente se comporta como se espera que se comporte.

Por ello, lo normal es que las pruebas sean muy numerosas, por lo que deben ser pequeñas, rápidas y automatizables. En el proyecto se ha usado *Karma* para automatizar la ejecución de los test unitarios, escritos usando *Jasmine*.

Para eliminar las dependencias de los componentes, puesto que deben ser probados de forma aislada, un método es el uso de dobles de prueba, o *mocks*, componentes falsos que imitan el comportamiento del real, no dependientes de otros componentes y cuyo resultado podemos controlar.

En el proyecto se han usado las pruebas unitarias fundamentalmente en los validadores de todos los formularios, pero su ámbito no ha ido más allá, debido a la limitación temporal del proyecto.

7.2.2. Pruebas de integración

Las pruebas de integración [83] prueban que la combinación de diversos componentes funciona como se espera. Lo normal es que sean más numerosas que las pruebas unitarias y que estén a más alto nivel, también son más lentas que las pruebas unitarias y, por tanto, menos automatizables.

Angular tiene una *suite* que se adapta bastante bien a este tipo de pruebas, *TestBed* [84], que ofrece una serie de utilidades que facilitan las pruebas, y que también se pueden usar en pruebas unitarias y e2e. Entre ellas, por ejemplo, ofrece servidores *mock*, que pueden simular un tiempo de espera y puede configurarse el tipo de respuesta que devuelven, son especialmente útiles en este tipo de pruebas.

En el proyecto, dos de los puntos más conflictivos son la conexión con el servidor y la conexión del servidor con la base de datos, puesto que dependen de la conexión.

El número de pruebas de integración que hay en el proyecto es mínimo, fundamentalmente por la restricción temporal, la necesidad de aprender en detalle el módulo TestBed de Angular y debido a la dificultad de diseñar las pruebas en el servidor, donde no se emplea Angular y se debería diseñar un mock de la base de datos que tenga en cuenta todos los estados y todos los tipos de llamadas.

7.2.3. Pruebas de aceptación

Las pruebas de aceptación [85], ya introducidas en el punto 4.1.3, prueban que un componente de software, sección o módulo de la aplicación se comporta exactamente como el usuario final quiere que lo haga.

Suelen derivarse de los escenarios de las historias de usuario, como los que se han concretado en el capítulo 5.

En el proyecto se han probado principalmente de forma manual y en la mayoría de casos por medio de pruebas pruebas e2e.

7.2.4. Pruebas *end-to-end* (e2e)

Las pruebas e2e [20] comprueban que la aplicación se comporta como el usuario final espera que lo haga.

La diferencia principal con las pruebas de aceptación, es que las pruebas de aceptación comprueban partes del código, como puede ser por ejemplo la respuesta del sistema ante los datos de un registro, mientras que las pruebas e2e se ejecutan sobre la propia aplicación y suelen operar sobre un flujo completo de ejecución, como un usuario final la usaría, siguiendo con el ejemplo, la respuesta del sistema ante los datos de un registro introducidos desde la interfaz gráfica, siendo este un caso real y concreto de uso.

La diferencia es mínima, en la mayoría de los casos suelen tratar los mismos escenarios que las pruebas de aceptación, pero no tiene por qué, pueden existir componentes que no sean usados por el usuario final y que estén presentes en las pruebas de aceptación.

La principal herramienta para diseñar este tipo de pruebas en Angular es *protractor* [86], cuyo equivalente en otros entornos sería *Selenium Webdriver* [87].

Es perfectamente compatible con Ionic, no obstante, finalmente se decidió por realizar las pruebas de forma manual sobre el dispositivo móvil, fundamentalmente por la falta de tiempo para aprender a usar la herramienta para diseñar buenos test.

7.3. Despliegue

Una de las partes más importantes en cualquier proyecto de software es la fase de despliegue, cuando el código pasa a producción y el usuario final o el cliente pueden empezar a usar la aplicación.

Hay muchas formas de realizar esto, y depende en buena medida del tipo del proyecto y de las tecnologías que emplee, el despliegue de una aplicación web puede ser muy distinto al de una de escritorio, por ejemplo.

Un concepto importante a tener en cuenta en esta sección, es el de la *entrega continua* [88], estrechamente ligado el de la *integración continua*, ya que lo amplía pasando los cambios a producción.

Con un sistema de entrega continua, todos los cambios que pasen por la integración continua de forma satisfactoria pasarán al entorno de producción, consiguiendo, por una parte, aportar valor lo antes posible y, por otra parte, tener un sistema que sea rápidamente adaptable, especialmente útil si hay alguna parte que modificar o algún error que no se ha detectado.

Se exponen, a continuación, las infraestructuras que se han empleado, tanto en la parte cliente como en la parte servidor, para realizar toda esta fase de despliegue, incluyendo la integración continua.

7.3.1. Parte cliente

En la parte cliente se ha empleado Bitbucket con Bitbucket pipelines para conseguir un sistema de integración continua. A esto se le ha añadido Codecov, que genera informes de cobertura de código, y que puede añadir condicionantes al ser ejecutado en el sistema de integración continua, como, por ejemplo, que se cubra un porcentaje mínimo de cobertura por parte de los test.

Esto permite rechazar cambios que no han sido lo suficientemente probados. Al ser obligado probar y que todas las pruebas pasen, es mucho más complicado que los errores lleguen a producción.

En este sentido se ha montado toda la infraestructura, pero no se está aplicando esto, se genera el informe pero no se exige un mínimo de cobertura, puesto que no se está probando el código de forma exhaustiva.

Tampoco se está generando un .apk y un .ipa donde deberían generarse para lograr la entrega continua, al término del *script* usado en la integración continua. No se está haciendo esto por la limitación temporal que tiene Bitbucket pipelines como servicio gratuito, hacerlo así consumiría mucho tiempo, puesto que requeriría construir el proyecto en la máquina remota y luego enviarlo a algún otro lugar.

No se está desplegando la versión web de la aplicación en producción puesto que es una aplicación preparada para funcionar en móvil, no se ha adaptado para tener un diseño *responsive* en web por el tiempo que ello habría conllevado, en sí, el subir la aplicación a producción podría hacerse fácilmente añadiendo las órdenes necesarias en el *script* de la integración continua, pero el resultado obtenido tendría bastante deficiencias y no sería del todo usable, por lo que se ha decidido no hacerlo.

Una solución idónea, a mi juicio, en un equipo de desarrollo, sería poder emplear una máquina local dedicada exclusivamente a realizar esta tarea, la de construir el proyecto, a la que se pudiesen encolar proyectos, que se ejecutase mediante una señal enviada por la integración continua y que pudiese publicar los cambios en producción.

La arquitectura completa empleada para el desarrollo de la parte cliente se muestra en la figura 7.20.

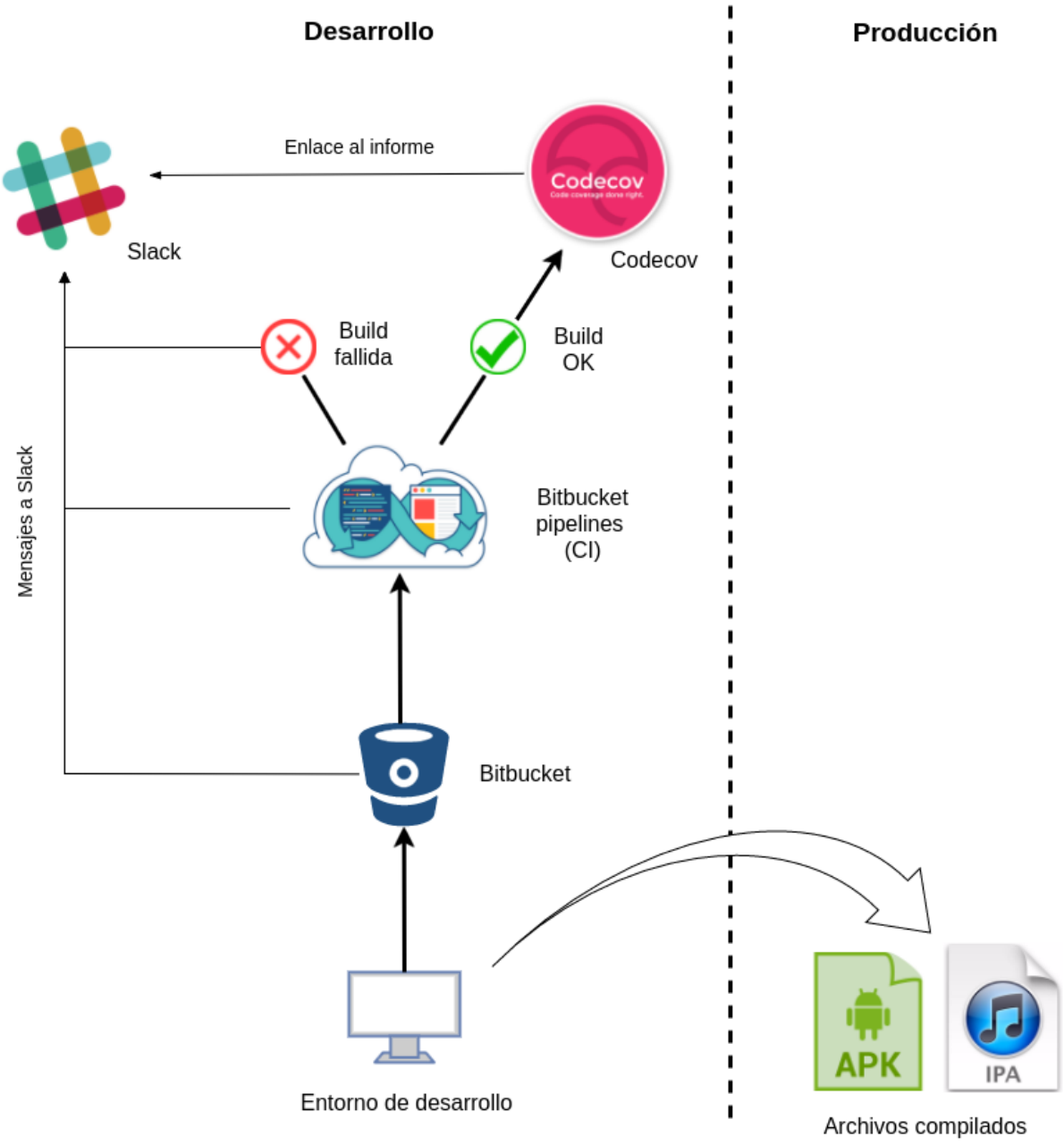

Figura 7.20: Diagrama de la arquitectura empleada para el desarrollo en la parte cliente. La solución con entrega continua propuesta arriba se lograría cambiando la flecha blanca del entorno de desarrollo a Codecov.

7.3.2. Parte servidor

En el caso de la parte servidor ocurre al contrario que en la parte cliente. Se realiza la entrega continua, pero no se está realizando la cobertura de código.

Esto ocurre porque Codecov ofrece un plan gratuito para un único proyecto privado por lo que, al pasar la integración continua, se despliega directamente en Heroku, eso sí, todo de forma automatizada, los cambios que se suben acaban en producción si puede construirse el proyecto.

La arquitectura completa empleada para el desarrollo de la parte servidor se muestra en la figura 7.21.

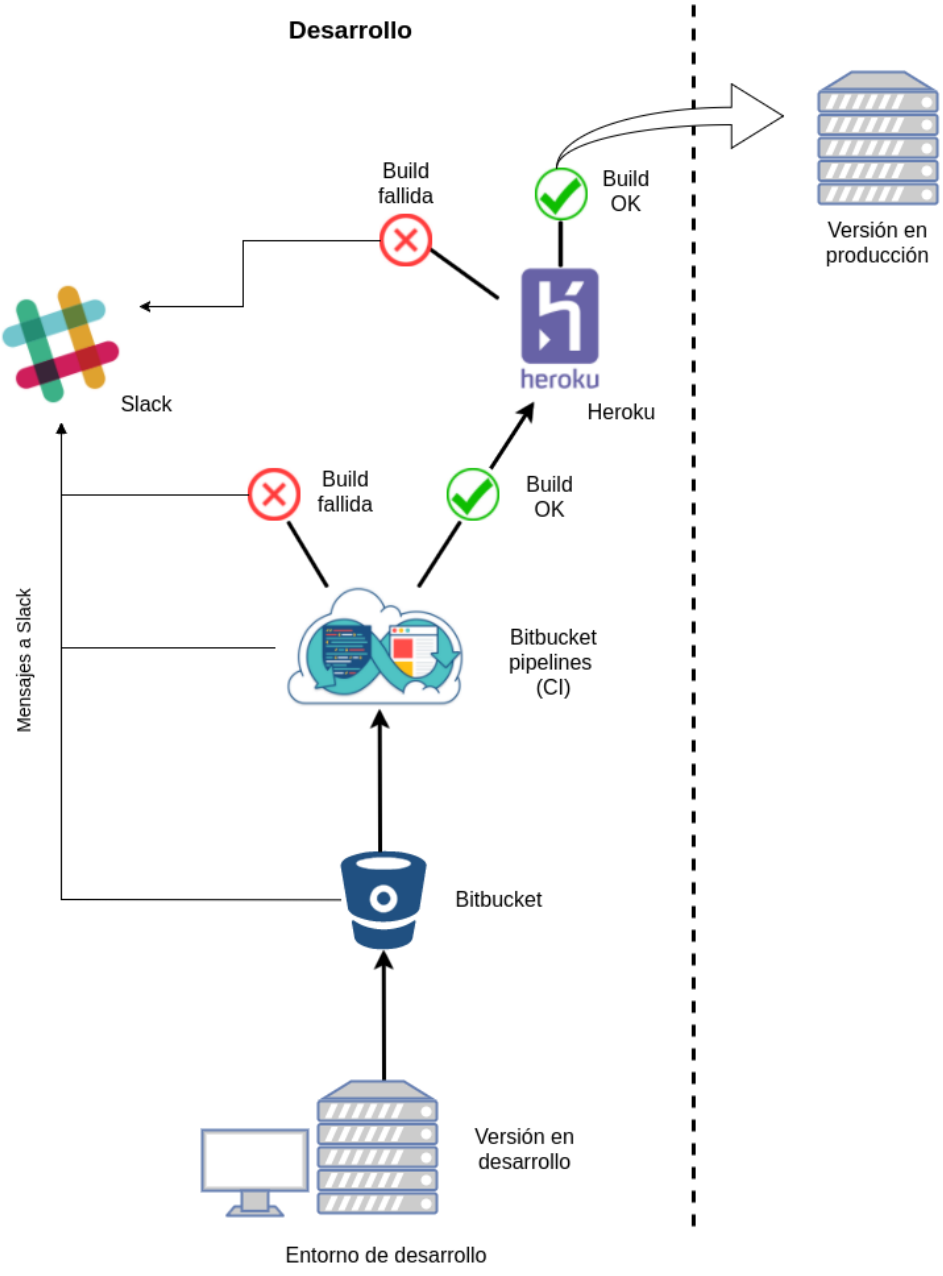

Figura 7.21: Diagrama de la arquitectura empleada para el desarrollo en la parte servidor.

Capítulo 8

Conclusiones

En este capítulo se exponen las conclusiones obtenidas al término del proyecto, sobre el proyecto en sí, a nivel formativo, profesional y personal.

8.1. Sobre el proyecto y las tecnologías empleadas

Gran parte de las tecnologías empleadas en el proyecto son tecnologías muy recientes, incluso el paradigma que se ha empleado en sí es relativamente reciente en comparación al de las aplicaciones web o nativas.

El resultado obtenido en este proyecto es muy similar, si no idéntico en comparación con una aplicación nativa, a nivel visual, la fluidez con la que se ejecuta, etc...

En mi opinión, estas tecnologías están preparadas hoy en día para poder ser usadas en un entorno profesional, más allá de las "pruebas de laboratorio", o los entornos más "técnicos".

No obstante, sí que hay que tener en cuenta dos problemas, en mi opinión muy serios, que pueden condicionar, y mucho, el uso de una aplicación desarrollada con este *stack*.

El primero, es el espacio que pueden llegar a ocupar, según el número de *plugins* que se utilicen, en general, aunque no tiene por qué ser siempre, las aplicaciones resultado de estas tecnologías suelen ocupar mucho espacio en comparación a las nativas. En algunos casos la diferencia es mínima, pero en otros puede llegar a ser un problema serio.

El segundo es el tiempo que tardan en cargar, que puede ser mucho, muchísimo, especialmente si lo comparamos con las aplicaciones nativas, si bien, superado este tiempo de carga, el funcionamiento es similar. Posiblemente con el uso de *NativeScript* o *React Native* podría resolverse, puesto que no dependen de Cordova, aunque comprobarlo ha quedado fuera del alcance de este proyecto.

En conclusión, si bien en mi opinión Ionic es una herramienta más que válida hoy en día para crear aplicaciones híbridas, creo que es usable sólo en determinados tipos de aplicaciones. No lo usaría en aplicaciones que requieran generar o trabajar con gráficos 2D o 3D, como son videojuegos, a menos que sean con tecnologías específicas como *WebGL*, y con ciertas dudas. Tampoco lo usaría en aplicaciones complejas, que tengan muchas pantallas y que requieran de muchas

funcionalidades (cámara, gps, teléfono, notificaciones, etc...) puesto que serán aplicaciones muy pesadas por la cantidad de módulos que será necesario instalar, y muy lentas en el arranque.

8.2. A nivel formativo

Personalmente, a nivel formativo he aprendido muchísimo sobre las tecnologías más recientes usadas en el entorno web y del ecosistema JavaScript. También he adquirido cierta experiencia en su uso, que creo que me hará afrontar proyectos similares en mucho menos tiempo, y también me servirá para proyectos web, no híbridos.

8.3. A nivel profesional

A nivel profesional la experiencia ha sido interesante, puesto que me ha permitido tener una experiencia en primera persona del funcionamiento de equipos de desarrollo y de la forma de organización, en este caso, de Cuatroochenta.

8.4. A nivel personal

A nivel personal considero que la experiencia ha sido bastante enriquecedora, y desde luego pienso que la dedicación ha merecido la pena por todos los conocimientos adquiridos, por todo lo nuevo que he aprendido y por la oportunidad de poner en práctica lo que he aprendido en el grado.

Bibliografía

[1] Bluumi.net, "Comparativa de tipos de aplicación." http://bluumi.net/app-nativa-vs-app-web-vs-app-hibrida/. [Consulta: 28 de Abril de 2017].

[2] Carlo Zapponi, "Página web de githut." https://madnight.github.io/githut/. [Consulta: 30 de Abril de 2017].

[3] Google, "Popularidad de los principales sistemas de control de versiones." https://trends.google.com/trends/explore?date=all&q=%2Fm%2F05vqwg,%2Fm%2F012ct9,%2Fm%2F08441_,%2Fm%2F08w6d6,%2Fm%2F09d6g&hl=en-US. [Consulta: 14 de Mayo de 2017].

[4] Amazon, "Definición de integración continua." https://aws.amazon.com/es/devops/continuous-integration/. [Consulta: 14 de Mayo de 2017].

[5] Helder de Oliveira, "Esquema del método scrum." http://agilizandoando.blogspot.com.es/2012/05/scrum.html. [Consulta: 15 de Mayo de 2017].

[6] Josh Morony, "An in depth explanation of providers in ionic 2." https://www.joshmorony.com/an-in-depth-explanation-of-providers-in-ionic-2/. [Consulta: 19 de Mayo de 2017].

[7] Arelano Gómez, "Diagrama de la arquitectura cliente-servidor." http://arellanoprogcs.blogspot.com.es/. [Consulta: 18 de Mayo de 2017].

[8] Soluciones Cuatroochenta, "Sobre cuatroochenta." http://www.cuatroochenta.com/sobre-cuatroochenta/. [Consulta: 28 de Abril de 2017].

[9] Google, "Página web de angular." https://angular.io/. [Consulta: 28 de Abril de 2017].

[10] Facebook, "Página web de react." https://facebook.github.io/react/. [Consulta: 28 de Abril de 2017].

[11] Tilde Inc, "Página web de emberjs." https://www.emberjs.com/. [Consulta: 28 de Abril de 2017].

[12] Node.js Foundation, "Página web de nodejs." https://nodejs.org/es/. [Consulta: 28 de Abril de 2017].

[13] The PHP group, "Página web de php." https://secure.php.net/manual/es/intro-whatis.php. [Consulta: 12 de Mayo de 2017].

[14] Fabien Potencier, "Página web de symfony." https://symfony.com/. [Consulta: 12 de Mayo de 2017].

[15] Taylor Otwell, "Página web de laravel." https://laravel.com/. [Consulta: 12 de Mayo de 2017].

[16] The Apache Software Foundation, "Página web de apache cordova." https://cordova.apache.org/. [Consulta: 28 de Abril de 2017].

[17] Adobe Systems Inc, "Página web de phonegap." http://phonegap.com/. [Consulta: 28 de Abril de 2017].

[18] Sencha Inc, "Página web de sencha." https://www.sencha.com/. [Consulta: 28 de Abril de 2017].

[19] Ionic, "Página web de ionic." https://ionicframework.com/. [Consulta: 28 de Abril de 2017].

[20] Techopedia, "Definición de test e2e." https://www.techopedia.com/definition/7035/end-to-end-test. [Consulta: 12 de Mayo de 2017].

[21] Bohemian BV, "Página web de sketch." https://www.sketchapp.com/. [Consulta: 28 de Abril de 2017].

[22] W3C, "Especificación de html5." https://www.w3.org/TR/html/. [Consulta: 30 de Abril de 2017].

[23] Mozilla Developer Network, "Una re-introducción a javascript." https://developer.mozilla.org/es/docs/Web/JavaScript/Una_re-introducci%C3%B3n_a_JavaScript. [Consulta: 30 de Abril de 2017].

[24] W3C, "Página web de css del w3c." https://www.w3.org/Style/CSS/. [Consulta: 13 de Mayo de 2017].

[25] GitHub, "Página web de github." https://github.com/. [Consulta: 30 de Abril de 2017].

[26] Ecma International, "Especificación de ecmascript 5." https://www.ecma-international.org/ecma-262/5.1/. [Consulta: 13 de Mayo de 2017].

[27] Microsoft, "Página web de typescript." https://www.typescriptlang.org/. [Consulta: 30 de Abril de 2017].

[28] Sass, "Documentación de sass." http://sass-lang.com/documentation/file.SASS_REFERENCE.html. [Consulta: 30 de Abril de 2017].

[29] Webpack, "Página web de webpack." https://webpack.js.org/. [Consulta: 30 de Abril de 2017].

[30] Browserify, "Página web de browserify." http://browserify.org/. [Consulta: 30 de Abril de 2017].

[31] Fundación Wikimedia Inc, "Definición de spa." https://es.wikipedia.org/wiki/Single-page_application. [Consulta: 14 de Mayo de 2017].

[32] Facebook, "Página web de react native." https://facebook.github.io/react-native/. [Consulta: 30 de Abril de 2017].

[33] Dan Abramov, "Página web de redux." http://redux.js.org/. [Consulta: 30 de Abril de 2017].

[34] RxJS, "Página web de rxjs." http://reactivex.io/rxjs/. [Consulta: 30 de Abril de 2017].

[35] José Antonio Íñigo, "¿qué es la programación reactiva? una introducción." http://profile.es/blog/que-es-la-programacion-reactiva-una-introduccion/. [Consulta: 30 de Abril de 2017].

[36] Jonas Bonér, Dave Farley, Roland Kuhn, Martin Thompson, "Reactive manifesto." http://www.reactivemanifesto.org/. [Consulta: 30 de Abril de 2017].

[37] Brandon Roberts y Mike Ryan, "Repositorio de ngrx." https://github.com/ngrx. [Consulta: 30 de Abril de 2017].

[38] Google, "Página web del motor v8 de chrome." https://developers.google.com/v8/. [Consulta: 14 de Mayo de 2017].

[39] StrongLoop Inc, "Página web de expressjs." http://expressjs.com/es/. [Consulta: 14 de Mayo de 2017].

[40] Douglas Wilson, "Página web de body-parser en github." https://github.com/expressjs/body-parser. [Consulta: 14 de Mayo de 2017].

[41] Troy Goode, "Página web de cors en github." https://github.com/expressjs/cors. [Consulta: 14 de Mayo de 2017].

[42] Fundación Wikimedia Inc, "Definición de la cabecera cors en wikipedia." https://en.wikipedia.org/wiki/Cross-origin_resource_sharing. [Consulta: 14 de Mayo de 2017].

[43] LearnBoost, "Página web de mongoose." http://mongoosejs.com/. [Consulta: 14 de Mayo de 2017].

[44] MongoDB Inc, "Qué es mongodb." https://www.mongodb.com/what-is-mongodb. [Consulta: 14 de Mayo de 2017].

[45] Amazon Web Services, Inc, "Definición de base de datos documental." https://aws.amazon.com/es/nosql/document/. [Consulta: 14 de Mayo de 2017].

[46] The Apache Software Foundation, "Página web de apache couchdb." http://couchdb.apache.org/. [Consulta: 14 de Mayo de 2017].

[47] The Apache Software Foundation, "Página web de apache cassandra." http://cassandra.apache.org/. [Consulta: 14 de Mayo de 2017].

[48] ObjectLabs Corporation, "Página web de mlab." https://mlab.com/. [Consulta: 14 de Mayo de 2017].

[49] Charlie Robbins, "Página web de winston en github." https://github.com/winstonjs/winston. [Consulta: 14 de Mayo de 2017].

[50] ObjectLabs Corporation, "Página web de jasmine." https://jasmine.github.io/. [Consulta: 14 de Mayo de 2017].

[51] Friedel Ziegelmayer, "Página web de karma." https://karma-runner.github.io/1.0/index.html. [Consulta: 14 de Mayo de 2017].

[52] Codecov, "Página web de codecov." https://codecov.io/. [Consulta: 14 de Mayo de 2017].

[53] Software Freedom Conservancy, "Página web de git." https://git-scm.com/. [Consulta: 14 de Mayo de 2017].

[54] Atlassian, "Página web de bitbucket." https://bitbucket.org. [Consulta: 14 de Mayo de 2017].

[55] GitLab, "Página web de gitlab." https://about.gitlab.com/. [Consulta: 14 de Mayo de 2017].

[56] Atlassian, "Página web de bitbucket pipelines." https://bitbucket.org/product/features/pipelines. [Consulta: 15 de Mayo de 2017].

[57] Yaml.org, "Especificación de yaml." http://www.yaml.org/spec/1.2/spec.html. [Consulta: 15 de Mayo de 2017].

[58] GitLab, "Uso de ficheros yml en gitlab." https://docs.gitlab.com/ee/ci/yaml/. [Consulta: 15 de Mayo de 2017].

[59] Slack, "Página web de slack." https://slack.com/. [Consulta: 15 de Mayo de 2017].

[60] GitLab, "Página web de travis ci." https://travis-ci.org/. [Consulta: 15 de Mayo de 2017].

[61] CircleCI, "Página web de circle ci." https://circleci.com/. [Consulta: 15 de Mayo de 2017].

[62] Codeship, "Página web de codeship." https://codeship.com/. [Consulta: 15 de Mayo de 2017].

[63] Semaphore CI, "Página web de semaphore ci." https://semaphoreci.com/. [Consulta: 15 de Mayo de 2017].

[64] Jenkins, "Página web de jenkins." https://jenkins.io. [Consulta: 15 de Mayo de 2017].

[65] Solano Labs, Inc, "Página web de codeship." https://www.solanolabs.com. [Consulta: 15 de Mayo de 2017].

[66] Salesforce, "Página web de heroku." https://www.heroku.com/. [Consulta: 15 de Mayo de 2017].

[67] Atlassian, "Página web de jira." https://es.atlassian.com/software/jira. [Consulta: 15 de Mayo de 2017].

[68] Jetbrains, "Página web de youtrack." http://www.jetbrains.com/youtrack/. [Consulta: 15 de Mayo de 2017].

[69] Wikipedia, "Definición de historia de usuario." https://es.wikipedia.org/wiki/Historias_de_usuario. [Consulta: 16 de Mayo de 2017].

[70] Wikipedia, "Definición de atdd." https://en.wikipedia.org/wiki/Acceptance_test%E2%80%93driven_development. [Consulta: 16 de Mayo de 2017].

[71] Samuel Casanova, "Definición de planning poker." http://samuelcasanova.com/2016/01/estimacion-agil-con-la-tecnica-planning-poker/. [Consulta: 17 de Mayo de 2017].

[72] Javier Garzás y Noemí Navarro, "¿por qué utilizamos puntos de historia y no horas?." http://www.javiergarzas.com/2015/06/puntos-historia-para-estimar-y-no-horas.html. [Consulta: 17 de Mayo de 2017].

[73] Wikipedia, "Definición de inyección de dependencias." https://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias. [Consulta: 19 de Mayo de 2017].

[74] Wikipedia, "Definición de singleton." https://es.wikipedia.org/wiki/Singleton. [Consulta: 19 de Mayo de 2017].

[75] Wikipedia, "Definición de la arquitectura cliente-servidor." https://es.wikipedia.org/wiki/Cliente-servidor. [Consulta: 18 de Mayo de 2017].

[76] Wikipedia, "Definición de sistema distribuido." https://es.wikipedia.org/wiki/Computaci%C3%B3n_distribuida#Sistemas_distribuidos. [Consulta: 18 de Mayo de 2017].

[77] Ionic Framework, "Componente menú de ionic 2." https://ionicframework.com/docs/api/components/menu/Menu/. [Consulta: 19 de Mayo de 2017].

[78] Wikipedia, "Definición de interfaz de línea de comandos (cli)." https://es.wikipedia.org/wiki/Interfaz_de_l%C3%ADnea_de_comandos. [Consulta: 19 de Mayo de 2017].

[79] Ionic Framework, "In app browser." https://ionicframework.com/docs/native/in-app-browser/. [Consulta: 21 de Mayo de 2017].

[80] Ionic Framework, "Navcontroller." https://ionicframework.com/docs/api/navigation/NavController/. [Consulta: 21 de Mayo de 2017].

[81] Google, "Google maps api." https://developers.google.com/maps/documentation/javascript/?hl=es-419. [Consulta: 21 de Mayo de 2017].

[82] Martin Fowler, "Unittest." https://martinfowler.com/bliki/UnitTest.html. [Consulta: 19 de Mayo de 2017].

[83] Microsoft, "Integration testing." https://msdn.microsoft.com/en-us/library/aa292128(v=vs.71).aspx. [Consulta: 19 de Mayo de 2017].

[84] Angular.io, "Testbed class specification." https://angular.io/docs/ts/latest/api/core/testing/index/TestBed-class.html. [Consulta: 19 de Mayo de 2017].

[85] Wikipedia, "Acceptance criteria and tests." https://en.wikipedia.org/wiki/Acceptance_test%E2%80%93driven_development#Acceptance_criteria_and_tests. [Consulta: 19 de Mayo de 2017].

[86] Angular, "Página web de protractor." http://www.protractortest.org/. [Consulta: 19 de Mayo de 2017].

[87] OpenQA, "Página web de selenium webdriver." http://www.seleniumhq.org/projects/webdriver/. [Consulta: 19 de Mayo de 2017].

[88] Amazon, "¿qué es la entrega continua?." https://aws.amazon.com/es/devops/continuous-delivery/. [Consulta: 19 de Mayo de 2017].