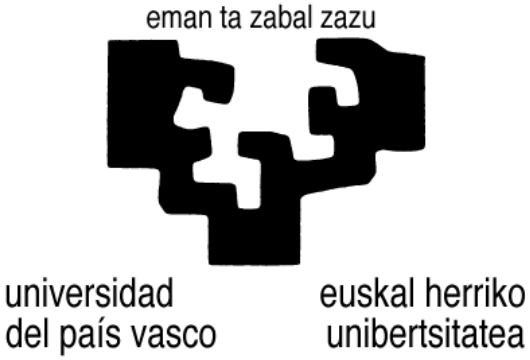
Proyecto Fin de Carrera

Aplicación Android para bicicletas basado en sistemas de geolocalización

Jon González Martín

2 de febrero de 2015

Director:
José Miguel Blanco Arbe



En primer lugar quisiera agradecer a José Miguel Blanco Arbe la oportunidad que me ha dado para realizar este proyecto, por tu ayuda, consejos y paciencia.

A mi familia por la confianza que han tenido en mí y por la paciencia de cada día, sobre todo a mis padres.

A mis amigas y amigos, por ayudarme con todos los problemas que me han surgido durante el desarrollo de este proyecto.

A cada bizitxapeldun por colaborar desinteresadamente en cada una de las fases de desarrollo del proyecto, en especial a Janire López Guillén.

Gracias a todos.

Resumen

Esta documentación corresponde a la memoria del Proyecto Fin de Carrera JonBike, desarrollado para la titulación de Ingeniería Informática en la Facultad de Informática de Donostia – San Sebastián de la UPV-EHU.

En este Proyecto Fin de Carrera se ha realizado el diseño y la implementación de la aplicación JonBike. Tiene como objetivo mejorar y optimizar el seguimiento de los entrenamientos de los ciclistas. El desarrollo se ha realizado en un marco de integración y colaboración directa de los usuarios en el proyecto, partiendo de un Producto Mínimo Viable o "Minimum Viable Product" (MVP) inicial e integrando su feedback en la progresiva ampliación de las características del servicio.

El sistema está compuesto por una aplicación móvil nativa para Android y una fuente de datos alojada en un servidor remoto. En implementación se han utilizado tecnologías emergentes, todas de código abierto: MongoDB junto con MongoLab para el almacenamiento de datos y plataforma base para el backend, y desarrollando como cliente, una aplicación Android nativa.

Índice general

Índice de figuras

Índice de cuadros

Extractos de código

Capítulo 1.

Introducción

La tecnología móvil ha evolucionado de una manera espectacular en esta última década. Haciendo memoria, recordaremos los móviles de los años 90, de grandes dimensiones, peso y funciones tan sencillas como realizar llamadas. Han pasado siete años ya desde que Android presentase su primera versión Android v1.0 Apple Pie hasta el último hasta ahora Android v5.0 Lollipop.

En los últimos años se ha producido una completa socialización de los smartphones. Poca gente a día de hoy no utiliza a diario un teléfono móvil con conexión a Internet y con la posibilidad de instalar en él aplicaciones de todo tipo. No sólo se usa Android en smartphones sino también en ordenadores portátiles, netbooks, tablets, Google TV, relojes de pulsera, auriculares y demás dispositivos.

Esta popularización se ha conseguido gracias a su apertura al mundo de los desarrolladores y la facilidad que tienen estos de ampliar su funcionalidad con programas que los usuarios pueden instalar fácilmente.

Android, al contrario que otros sistemas operativos para dispositivos móviles como iOS o Windows Phone, se desarrolla de forma abierta[1] y se puede acceder tanto al código fuente como a la lista de incidencias donde se pueden ver problemas aún no resueltos y reportar problemas nuevos.

De esta manera, está abierto un inmenso abanico de nuevas ideas y posibilidades que no solo han cubierto nuevas necesidades sino que es posible hacer de una cosa sencilla como un paseo en bicicleta, un entrenamiento con una base de gamificación.

En este contexto de saturación del mercado de las aplicaciones móviles es donde JonBike toma forma y es creada precisamente para potenciar la motivación, la concentración, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos de un entrenamiento.

El progresivo desarrollo del producto ha sido conducido por un grupo de usuarios selectos, denominados bizitxapeldunes, en los cuales se ha delegado la tarea de tomar las decisiones sobre el diseño y las características del producto.

[1] Expresión con la que se conoce al software distribuido y desarrollado libremente. http://es.wikipedia.org/wiki/Código_abierto

Esta integración de personas externas ha requerido tareas de gestión exclusivas, con entrevistas guiadas por cuestionarios, comunicación constante sobre el progreso del proyecto y tecnologías para la automatización del feedback en caso de error.

JonBike ha sido implementado con tecnologías de código libre. Tras descartar otros servicios y opciones, se ha optado por hacer uso de MongoDB-as-a-Service. MongoLab y su REST API [2] junto con bases de datos NoSQL MongoDB para la persistencia de los datos de la aplicación.

Se ha desarrollado un cliente, una app para Android. Se ha optado por el sistema operativo Android por el conocimiento previo adquirido junto con el entorno de desarrollo Eclipse + ADT Android. Por esta razón se ha descartado el desarrollo en forma de aplicación multiplataforma.

JonBike hace uso de los servicios de autenticación de Google+, descartando la tarea de gestionar credenciales. Y los servicios de posicionamiento y mapas de GoogleMaps para la representación de la información del entrenamiento.

Este documento, que es la memoria del trabajo realizado por Jon González Martín bajo la dirección del doctor José Miguel Blanco Arbe durante el curso académico 2014/2015, está estructurado de la siguiente forma:

- Los objetivos del proyecto, explicando el producto a desarrollar junto con el alcance que abarca y las exclusiones de éste.

- Las tecnologías que se han utilizado para desarrollar el producto, describiendo para cada una las razones que llevaron a su elección frente a las alternativas existentes.

- Los procesos de selección, gestión y motivación de los bizitxapeldunes del proyecto. Se abordan los métodos utilizados para la recogida del feedback y su integración dentro del desarrollo del proyecto.

- Se describe la arquitectura global de JonBike, enumerando las partes que lo componen y la interacción entre todas ellas.

- Para cada una de las partes desarrolladas (backend y cliente Android), se describe su funcionamiento interno, la lógica de negocio que tratan y las interfaces gráficas. Todos estos elementos van acompañados de ilustraciones y diagramas que ayudan a entender su contenido.

[2] Técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. http://es.wikipedia.org/wiki/Representational_State_Transfer

- El siguiente capítulo relata la gestión propia del proyecto en las áreas principales no tratadas en el resto de la memoria: el alcance, el tiempo y los costes.

- El proyecto concluye con las valoraciones personales como extracto de la experiencia de este proyecto, a los cuales les acompañan algunas lecciones aprendidas probablemente útiles en proyectos futuros similares a este.

A lo largo de toda la memoria se repiten numerosos términos y acrónimos que podrían ser desconocidos para el lector o tener, en el contexto de este proyecto, un significado distinto. Por ello, se adjunta un glosario que describe la definición de estas palabras, al cual conviene acudir en caso de duda.

Capítulo 2.

Objetivos del proyecto

Este capítulo explica los objetivos del proyecto, describiendo el funcionamiento de JonBike y después se detalla el alcance del proyecto junto con las exclusiones de éste.

El objetivo del proyecto es desarrollar una aplicación para el sistema operativo Android. En concreto, se trata de implementar una aplicación que permite al usuario realizar un seguimiento de sus entrenamientos en bicicleta. Por medio del uso del GPS del Smartphone y GoogleMaps el usuario visualizará su posición en el mapa y datos relativos al entrenamiento en su teléfono Android.

Se trata de una aplicación con la que el usuario obtiene información extra a medida que va entrenando. Se muestra la ubicación en el mapa, la velocidad actual, la velocidad máxima, la distancia recorrida, la velocidad media del entrenamiento, la altitud actual y la altitud máxima. Todo ello junto con un cronómetro en la parte inferior de la pantalla entre los botones de START y STOP.

Los datos de cada entrenamiento están vinculados a una cuenta de usuario que permite a los ciclistas visualizar y guardar las rutas registradas.

La aplicación se desarrolla desde un Minimum Viable Product.[1] Con un desarrollo dividido en tres ciclos, se han fijado objetivos a lo largo del desarrollo del proyecto mediante la obtención de feedback y su posterior desarrollo en el proyecto.

2.1. Alcance del proyecto

Se ha desarrollado un sistema compuesto por una fuente de datos y una aplicación móvil, para el sistema Android, que permite el uso del dispositivo como punto de información para los entrenamientos y rutas del usuario ciclista.

Los detalles del alcance sobre la aplicación desarrollada han sido:

- Los usuarios pueden iniciar sesión a través de Google+.

[1] Versión de un producto nuevo que permite a un equipo recolectar la máxima cantidad de conocimiento validado sobre clientes con el menor esfuerzo posible.

- Cada usuario ve en tiempo real su ubicación en el mapa y los datos relativos al entrenamiento.

- Los usuarios pueden, una vez finalizado el entrenamiento, guardar o descartar dicho entrenamiento en su perfil.

- Se puede personalizar el icono de ubicación así como también el mapa de entrenamiento en todo momento.

- Cada usuario tiene acceso a su historial de ciclista. Se muestran los últimos entrenamientos realizados.

2.2. Exclusiones del proyecto

Se han excluido del alcance del proyecto los siguientes puntos:

- Se excluye la implementación de un módulo propio para la autenticación de los usuarios, que se ha delegado en un servicio externo.

- El análisis de carácter legal que resultaría necesario en una aplicación final. La aplicación realiza una importante recogida de datos personales (geolocalización) que se almacenan en el servidor. Esta recogida requiere de una consideración en las distintas consecuencias legales, sobre todo los relativos a la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) y la Ley de Servicios en la Sociedad de la Información (LSSI), lo cual sería materia suficiente para un Proyecto Fin de Carrera separado.

- La labor de documentación y sistematización de la gestión del servidor de bases de datos la aplicación. En el proyecto se pondrán en línea los sistemas esenciales para el funcionamiento de JonBike, pero no se profundizará.

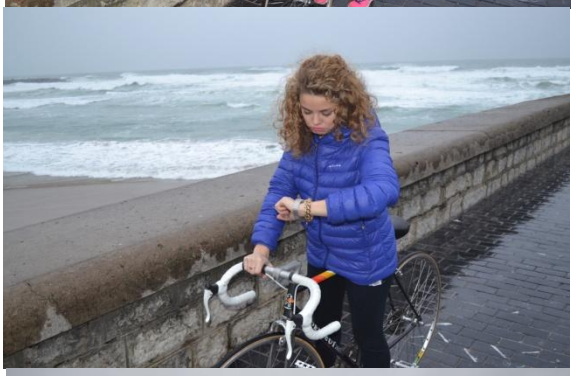
- De la misma forma, se excluye la realización de un análisis exhaustivo de los costes y de las características de un servidor de pago, ya que se opta por una alternativa gratuita.

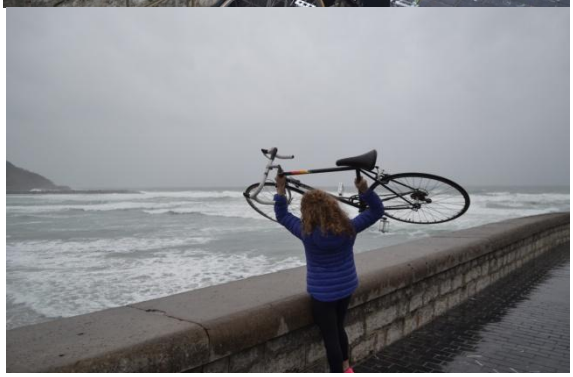
- Se excluye el desarrollo de la aplicación multiplataforma.

2.3. El objetivo en imágenes

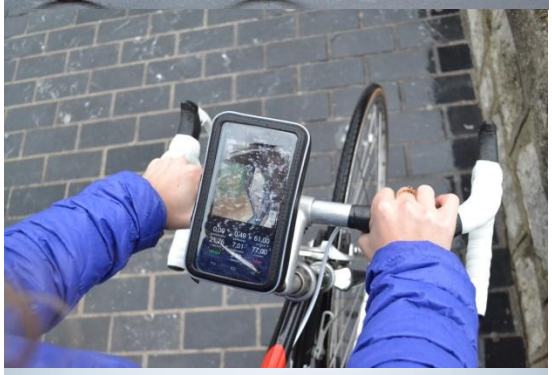
La **primera** transición de imágenes, se muestra como se realiza un entrenamiento sin la aplicación JonBike. Se aprecia la pérdida de información para un buen seguimiento. La **segunda** transición se visualiza al bizitxapeldun haciendo uso de la app y concluyendo su entrenamiento satisfecho al saber exactamente la ruta que ha realizado, los km que ha recorrido y el tiempo que ha empleado en recorrerlos.

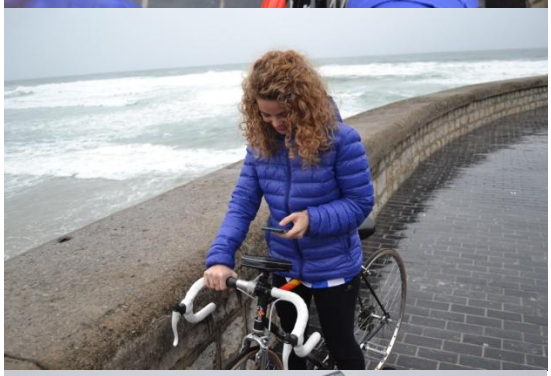
SIN JonBike





CON JonBike





Capítulo 3.

Tecnologías utilizadas

A lo largo del proyecto se han ido variando las tecnologías utilizadas. En cada ciclo del proyecto se han ido adquiriendo conocimientos y nuevas tecnologías las cuales se han valorado y comparando con el fin de mejorar la aplicación. Para cada una de ellas se explica la necesidad que motivó su adquisición, las alternativas existentes y las razones para la elección, y las características principales que la componen.

Cada adquisición ha ido marcando el camino de desarrollo del proyecto en cada ciclo. La mayoría de tecnologías habían sido seleccionadas desde el principio, pero dependiendo de las necesidades de cada ciclo se han ido reemplazando. En casi todas las tecnologías, un factor decisivo para la elección de una frente a sus competidoras ha sido la experiencia previa con ellas y la facilidad de implantación en el sistema.

Las secciones de este capítulo desarrollan las siguientes tecnologías: MongoDB como base de datos, la plataforma de desarrollo móvil Android y finalmente los servicios de MongoLab como MongoDB-as-a-Service bajo el soporte de Amazon web services.

3.1. Base de datos: MongoDB

MongoDB es una base de datos NoSQL orientada a documentos, desarrollada con código abierto. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON[1], que es una representación binaria de JSON[2].

Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección pueden tener esquemas diferentes.

3.1.1. Elección

El proyecto al principio no tuvo ninguna restricción fuerte en cuanto al sistema de almacenamiento. La complejidad del modelo de datos es pequeña y las operaciones pueden permitirse perder consistencia en favor de un mayor rendimiento.

[1] www.bsonspec.org
[2] www.json.org

Tras la elección de MongoLab como backend, se escoge MongoDB como base de datos NoSQL para el almacenamiento de datos. MongoDB, software libre y popular, es la opción de referencia en otros proyectos basados en geolocalización.

3.2. Elección de la plataforma de desarrollo móvil

A la hora de estudiar la plataforma sobre la que el proyecto iba a funcionar, se estudiaron diferentes posibilidades. La primera de ellas, era decidir si la aplicación sería desarrollada de forma nativa para Android, o si, por el contrario, sería una aplicación tipo web, en concreto, HTML5.

La aplicación del sistema se ha desarrollado de forma nativa, si bien la barrera que separa ambos mundos cada día es más frágil, sí que existen ciertas diferencias que pueden decantar el desarrollo por una o por otra tecnología. La decisión se ha basado prácticamente en dos factores que han sido los más determinantes: rendimiento y distribución.

Se conoce que las aplicaciones desarrolladas en HTML5 son más lentas que las aplicaciones nativas, y se ha considerado que, en esta aplicación móvil es de gran importancia, ya que se usará en espacios cortos de tiempo, y se quiere un acceso inmediato y en tiempo real a la información. Además, en este caso particular, el consumo de batería sería mayor al utilizar una aplicación web, ya que conectarse continuamente al GPS para localizar al usuario es más costoso que hacerlo mediante una aplicación nativa, debido a la optimización de las aplicaciones para el uso del GPS.

El otro dato que ha decantado la balanza es el que podemos observar en la figura 3.1. Esta gráfica muestra por un lado el número de usuarios a nivel global. Y el tiempo que el usuario medio pasa tanto en las aplicaciones nativas contra el tiempo que pasa en las aplicaciones de tipo web.

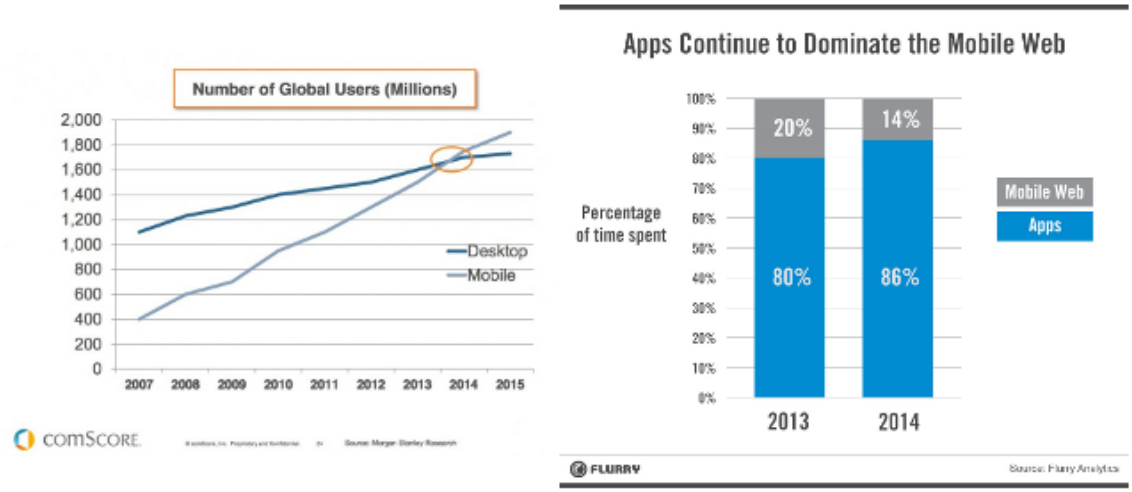


Figura 3.1.: Porcentaje de uso app nativas vs web.

Siendo el uso y la distribución uno de los grandes objetivos de este proyecto a la hora de implementar la aplicación; y tras ver estos datos; se puede entender por qué se ha decidido por hacerlo de forma nativa.

3.2.1. La elección para JonBike

Desde el inicio, la apuesta que se realizó para JonBike fue la de aplicación Android nativa. Las razones son las anteriormente mencionadas, las funcionalidades del entorno multiplataforma no son suficientes. La ubicación y los parámetros de JonBike deben ser inmediatos, mostrando los datos y la ruta en un corto plazo de tiempo a los usuarios.

Las aplicaciones web solo se encuentran activas cuando están en primer plano, por lo que no es posible tener presente siempre la localización, realizar operaciones de red o actualización de parámetros en todo momento.

Por estas razones, se tomó la decisión de realizar la aplicación de forma nativa y no multiplataforma, con el coste de desarrollo y aprendizaje que ello supone.

3.2.2. Elección de Android

Dentro de las plataformas nativas de desarrollo, la elección ha sido la de Android. La primera de las razones que han llevado a esta decisión es que Android es el sistema operativo de móviles más vendido actualmente, con un 78,4% de cuota de mercado. En el caso de España, esa cifra sube al 90%.

El dispositivo con el que se cuenta para el desarrollo y prueba es Android, tanto los propios como la gran mayoría de los de los bizitxapeldunes.

En cuanto a coste económico, la primera opción alternativa a Android sería iOS, el cual tiene un coste anual de 99$ (87,86€) anuales para la distribución de la aplicación. El coste de Google Play Store es un pago único de 25$ (22,19€).

Por último, como software libre, el código fuente de Android se encuentra disponible en Internet y pueden recogerse ideas del mismo.

3.3. Proveedor MDBaaS: MongoLab

El backend escogido para JonBike debe estar disponible a la red constantemente para que en cualquier momento los usuarios hagan uso del servicio. Existen multitud de proveedores que ofrecen servicios de distinto tipo según las necesidades de cada proyecto.

A continuación se explican la forma en la que los distintos proveedores pueden ser categorizados según su filosofía, las distintas características de cada categoría, algunos ejemplos de proveedores y las características básicas del proveedor escogido.

3.3.1. Categorías de proveedores

Los proveedores de servicios en la nube pueden ser clasificados en tres grandes categorías, según si el servicio ofrecido es de más «bajo» o «alto» nivel. En ese orden, las categorías son Infraestructure–as–a–Service (IaaS), Platform–as–a–Service (PaaS) y Software–as–a– Service (SaaS).

Infraestructure–as–a–Service

Estos servicios ofrecen una infraestructura sobre la que construir la arquitectura de nuestra aplicación. El proveedor pone a disposición del cliente una instancia de un sistema operativo (normalmente virtualizado). Así, el desarrollador tiene un control total sobre esta instancia, y es su responsabilidad poner en marcha en él los distintos servicios que le hagan falta.

Un mismo proveedor suele categorizar las distintas opciones según el rendimiento de las instancias virtualizadas. Es decir, el servidor tendrá un coste mayor en cuanta más capacidad tenga este en términos de CPU, memoria RAM y tamaño del disco duro.

Uno de los proveedores más conocidos es Amazon AWS[3], que tiene disponibles instancias en centros de proceso de datos a lo ancho de todo el mundo. Otras alternativas conocidas son Joyent o Rackspace. El cuadro muestra una comparación de estos servicios y su precio, con características similares y de una gama media-baja.

Proveedor	Características	Precio mensual
Amazon AWS	1 vCPU, 160 GB de disco y 1,7 GiB de RAM	31,68 $ (28,12 €)
Joyent	1 vCPU, 56 GiB de disco y 1,75 GiB de RAM	40,32 $ (35,79 €)
Rackspace	1 vCPU, 20 GiB de disco y 1 GiB de RAM	29,20 $ (25,92 €)

Cuadro 3.1.: Comparativa de servicios Infraestructure–as–a–Service (IaaS)

Platform–as–a–Service

Esta categoría de servicios, en vez de poner a disposición un sistema operativo completo sobre el que se tiene total control, el desarrollador tiene un servidor preinstalado capaz de ejecutar aplicaciones web de una determinada pila tecnológica.

3 https://aws.amazon.com/es/

Los clásicos hostings de Apache+PHP+MySQL sobre los que se ejecutan la mayoría de webs personales y profesionales encajan en esta categoría. No obstante, el número de plataformas que se utilizan se ha diversificado y lo mismo ha ocurrido con los proveedores.

Existen proveedores de esta categoría que tienen una tarifa gratuita, para proyectos pequeños o que están comenzando a ponerse en marcha. A partir de ahí, las tarifas escalan de forma similar a los servicios IaaS, aumentando en prestaciones, pero también añadiendo características al servidor[7].

Google App Engine, Windows Azure, Heroku, CloudBees o Cloud Foundry son algunas de las alternativas existentes en esta categoría. En el cuadro 3.2 se muestran las tecnologías con las que funcionan y si disponen de versión gratuita.

Proveedor	Tecnologías soportadas	Tarifa gratuita
Google App Engine	Python, Java, Go y PHP	Sí, con cuotas
Heroku	Java, Scala, Python, Node.js, Ruby y Clojure	Sí, una instancia
Windows Azure	.NET, PHP, Node.js y Python	Sí, con limitaciones
Cloudbees	Java	Sí
Cloud Foundry	Java, Python y Node.js	Según proveedor

Cuadro 3.2.: Comparativa de servicios Platform-as-a-Service (PaaS)

Software–as–a–Service

Aunque generalmente no se les suele etiquetar con este nombre, estas son las conocidas aplicaciones en la nube, contrapuestas a las aplicaciones de escritorio. Gmail, Hotmail, Dropbox o Evernote son algunos de los ejemplos más conocidos de los muchos existentes.

No obstante, estas aplicaciones en la nube quedan en el nivel del usuario y no son herramientas para desarrolladores de aplicaciones, por lo que su análisis queda fuera de este proyecto.

3.3.2. Proveedor escogido

De entre todas las alternativas anteriores, en el primer ciclo de desarrollo de la aplicación se seleccionó Heroku. Su servicio se basa en la infraestructura de Amazon AWS, la cual soportaba las plataformas de desarrollo y no supone coste económico alguno.

Pese a las recomendaciones y conocimiento con el proveedor, la elección cambio en el segundo ciclo de desarrollo. En el comienzo de la implementación del backend de la aplicación, se hizo un estudio en profundidad de las tecnologías. Se descubre

MongoLab, tecnología que implementa una base de datos MongoDB y una conexión directa a través de la REST API de MongoLab. De esta manera se conecta directamente la base de datos con la aplicación. De esta manera la aplicación se basa en una conexión directa a través de la REST API como muestra la imagen:

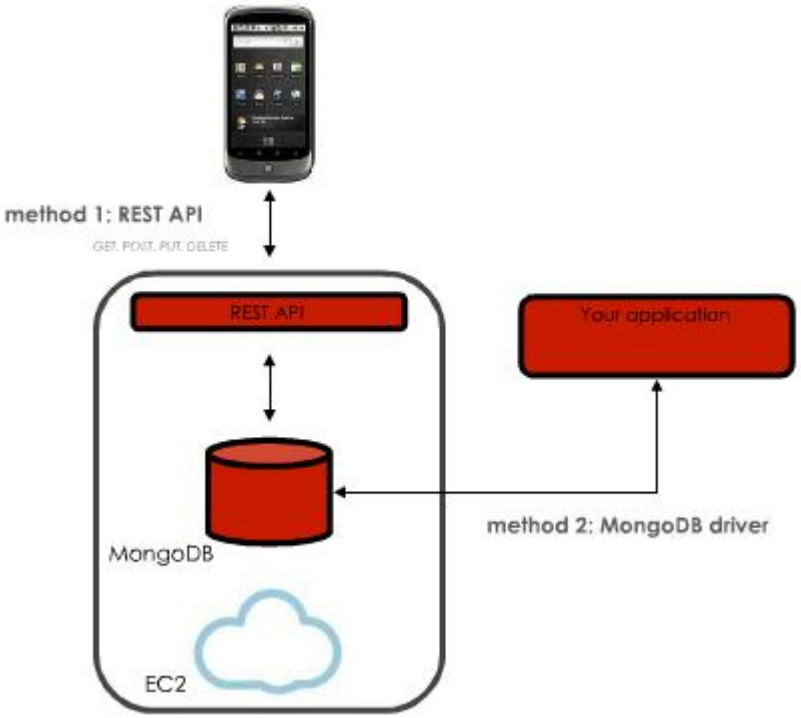


Figura 3.2.: conexión base de datos con app.

MongoLab es un servicio completo de código abierto de base de datos en la nube que aloja bases de datos MongoDB. MongoLab se ejecuta en proveedores en la nube como Amazon, Google, Joyent, Rackspace y Windows Azure, y se ha asociado con proveedores de Platform-as-a-Service.

Los proveedores de Platform-as-a-Service (PaaS) son Cloud Foundry, Engine Yard, Heroku, Nodejitsu, OpenShift y Windows Azure. Y como único y exclusivo proveedor de bases de datos, MongoDB.

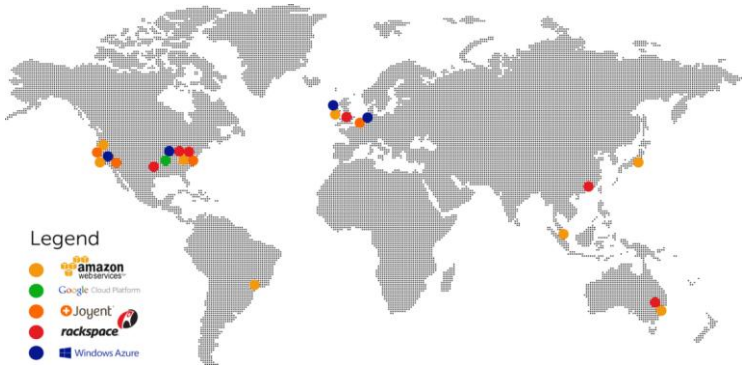


Figura 3.3.: Proveedores de PaaS.

3.4. Feedback de errores automático con Splunk de MINT Express

El desarrollo de aplicaciones para Android es casi idéntico al de desarrollar un programa que se vaya a ejecutar en el propio ordenador. La diferencia está en que el entorno de desarrollo compila y empaqueta la aplicación, la instala en un dispositivo Android simulado o uno real conectado por USB y lo ejecuta.

Mediante esta conexión USB es posible realizar todas las tareas de depuración que se podrían realizar en un programa de ordenador: revisar la ejecución con interrupciones, vigilar la memoria consumida, pero lo más importante es trazar los errores a las líneas de código exactas donde se producen, para proceder al arreglo.

No obstante, no siempre es posible detectar todos los potenciales focos de errores durante el desarrollo, y es probable que una aplicación se publique con errores. Los usuarios detectarían estos errores porque la aplicación se cerraría con una alerta, pero el mensaje de error no llegaría al desarrollador.

Es por ello que es necesario crear un sistema por el cual, sin depender de la voluntad o conocimiento de los usuarios, se recojan de manera automática y ordenada estos informes de errores. A continuación se repasan las posibles soluciones a esta necesidad, la escogida para llevar adelante este proyecto, la forma de integración en la aplicación y el método de funcionamiento para el trabajo con el servicio.

3.4.1. Alternativas disponibles

Google proporciona su propio sistema de reporte de errores integrado en Google Play. Si una aplicación ha sido instalada a través de su plataforma, en el momento del error se muestra un mensaje de alerta que ofrece dos opciones: aceptar para cerrar sin realizar ninguna acción, o reportar el informe de error.

No obstante, pocos usuarios reportan el fallo, por la sensación de pérdida de privacidad que supone (se debe enviar una importante cantidad de información sobre el dispositivo) o por pura incomodidad de seguir los pasos.

Por los inconvenientes anteriores, se decidió analizar y pasar a utilizar un servicio alternativo no dependiente de Play Store. Existen numerosas opciones por Internet, fácilmente localizables por Google. Sin embargo, por la falta de conocimiento sobre el tema y las alternativas disponibles, una rápida comparación llevó a la elección: Splunk de MINT. Las características eran similares entre unos y otros servicios, pero este dispone de un plan básico totalmente gratuito, apropiado para este proyecto.

3.4.2. Integración

Splunk se integra con la aplicación mediante dos sencillos pasos. Por un lado, se debe instalar una librería de Java que la propia empresa facilita para ser descargada. Después, se debe añadir una línea de código en el arranque del programa, que tiene como único parámetro un token que identifica al desarrollador dentro de Splunk.

De esta forma, cuando la aplicación se cierra por un error, se guarda el informe antes del cierre, y este es enviado de forma automática cuando la aplicación vuelva a abrirse y exista una conexión a Internet activa. No es necesario ningún tipo de intervención por parte del usuario.

3.4.3. Método de funcionamiento

Los informes de errores recibidos se almacenan y se muestran al desarrollador, con la siguiente información, entre otros:

- La traza del error, con los archivos y número de línea dentro del código donde se han producido los errores.
- Toda la información del dispositivo: el modelo de móvil, la versión del sistema operativo Android, la cantidad de memoria RAM y otros datos relativos al estado, como las conexiones a Internet disponibles y la disponibilidad de la localización.
- Datos sobre la configuración del dispositivo, como el idioma establecido en el sistema operativo, o la cuenta de usuario activa en la aplicación, que el desarrollador debe encargarse de establecer en el propio programa.

Como la cantidad de errores que pueden recibirse para un mismo fallo puede ser grande, dependiendo del número de usuarios, estos se agrupan para la comodidad, y se muestran estadísticas relativas a los datos anteriores.

El método de funcionamiento con estos informes de errores es muy similar al trabajo con un sistema de tickets de soporte. El desarrollador debe corregir el fallo y después indicarlo en Splunk, junto con la versión que lo arregla, para llevar un histórico. De esta manera, puede ignorar los informes de versiones en los cuales es conocido el error, o mostrar una alerta si se vuelve a repetir la incidencia en una versión posterior.

Capítulo 4.

Participación de los usuarios y ciclo de vida del proyecto

Este capítulo de la memoria describe la metodología seguida durante el trabajo en cada uno de los tres ciclos de vida con los usuarios, cuáles han sido los criterios para su selección y categorización, las vías utilizadas para la obtención de feedback y su posterior procesamiento en cada ciclo del proyecto.

Desde las primeras ideas de JonBike, hasta la obtención de una definición y materialización de este proyecto, existe un trabajo planificado y desarrollado en tres ciclos de vida para la buena recopilación de ideas y toma de decisiones.

La decisión de integrar a los usuarios y de realizar tres ciclos de desarrollo fue tomada desde el principio. Es de vital importancia para el proyecto dicha decisión para lograr un producto de éxito y no derivar en un producto con errores de diseño y carencias de funcionalidad y utilidad.

Lo que se trata de evitar con esta metodología es que el ciclo de vida del proyecto esté constituido únicamente por el propio alumno. Que las fases de especificación, análisis, diseño e implementación no sea modificado y mejorado mediante la interacción con posibles clientes o usuarios.

4.1. Ciclo de vida del proyecto

Para lograr un producto exitoso, el desarrollo de la aplicación se ha realizado en tres ciclos de vida.

Ciclo 1.

- Se desarrolla entre las fechas 12.10.2014 y 13.11.2014.
- Con una duración de 31 días.
- La primera participación de los bizitxapeldunes está constituida por Jose Miguel Blanco.

Ciclo 2.

- Se desarrolla entre las fechas 14.11.2014 y 11.12.2014.
- Con una duración de 28 días.
- La segunda participación de los bizitxapeldunes está constituida por Janire López, Guillermo López, Jon Gabilondo, Víctor Sancha y Dersu García.

Ciclo 3.

- Se desarrolla entre las fechas 12.12.2014 y 02.02.2015.
- Con una duración de 53 días.
- La tercera participación de los bizitxapeldunes está constituida por un amplio grupo de testers seleccionado y entrevistado para realizar las pruebas.

4.2. La muestra de bizitxapeldunes

Los usuarios seleccionados para la fase de pruebas deben cumplir con algunos criterios para que la experiencia sea útil y satisfactoria tanto para el desarrollador como para los usuarios en cada uno de los ciclos del producto.

El grupo de bizitxapeldunes de divide en tres, un grupo constituido por usuarios con un nivel de conocimiento bajo en informática, un segundo grupo de usuarios con un nivel avanzado en informática y por último, un tercero con usuarios de nivel avanzado en Android. Para escoger a las personas de cada grupo se han analizado los siguientes parámetros:

- Una amplia variación de puntos de vista de la aplicación. Desde usuarios con conocimientos básicos a usuarios con conocimientos expertos en Android.

- Una amplia gama de dispositivos en los cuales probar la aplicación aumenta la probabilidad de encontrar fallos y optimizar la aplicación para múltiples dispositivos. Entre otros, los factores a tener en cuenta serán: si es un teléfono o una tableta, versión del sistema operativo, tamaño de la pantalla, etc.

- Un amplio rango de perfiles dependiendo de la personalidad de los usuarios. Es un factor de extrema importancia para obtener el tipo de feedback que se busca. Se descartan los perfiles demasiado críticos (que pueden sobrecargar con comentarios no constructivos) o aquellos que pueden tender a llamar la atención

mediante críticas absurdas en la aplicación. Por último, es conveniente contar con testers de ambos géneros.

Estos bizitxapeldunes, además de cumplir con los puntos anteriormente enumerados, serán personas cercanas y de confianza, con la disponibilidad suficiente como para mantener reuniones regulares y su actitud deberá ser proactiva desde el inicio. La función de estas personas será proponer mejoras y tomar las decisiones sobre la aplicación para cumplir los objetivos del proyecto.

4.3. Documentación a preparar

Los testers necesitan están informados de su condición, de sus funciones y de lo que ocurre alrededor de la aplicación, por lo que se preparará la siguiente documentación para estos, que ayuden a cumplir los objetivos deseados:

Guía del bizitxapeldun: Documento breve y preciso que describe, por una parte, la función de JonBike y las características que lo componen. Por otra, las pautas de actuación definidas como testers: invitación a utilizar la aplicación, a fijarse en las características que puedan ser mejoradas y a cómo transmitir el feedback.

Cuestionarios personales: Los cuestionarios introducen inicialmente los cambios que ha habido desde la última entrevista para poner en contexto al usuario. Posteriormente, se enumeran las acciones que debería realizar junto con las preguntas sobre estas características centrando la atención en puntos deseados. Esta dirección se ha llevado a cabo utilizando unos cuestionarios preparados antes de las entrevistas, para ser entregados en ellas.

Actas de feedback: Los comentarios, aportaciones y cualquier otro dato interesante aportado se anotan en actas fechadas. Al ser un documento directo del feedback, es importante no realizar filtro de los comentarios y recibir todos los aportes tal cual sean transmitidos por el tester.

Propuestas de mejora: Tras analizar su viabilidad, unificar las similares y adaptarlas a las posibilidades, son exportadas en un documento que recoge las propuestas de mejora que se llevarán a cabo en un futuro. Este documento es dinámico, ya que se irán añadiendo a la lista nuevas propuestas, se descartarán algunas anteriores o nuevas ideas pueden superar y reemplazar a las anteriores.

Para cada uno de los documentos mencionados en esta sección, se incluyen los formatos de documento con un ejemplo real de cada uno en el apéndice A.

4.4. Metodología de interacción

La participación de los usuarios en el proceso de desarrollo requiere de un procedimiento dedicado y de planificación propia, teniendo claro en cada interacción la información que se quiere obtener de ellos.

Esta es la metodología seguida para integrar los comentarios de los bizitxapeldunes durante el desarrollo de este proyecto. El proceso pasa por las fases de introducción, feedback y de notificación posterior sobre el progreso.

Los bizitxapeldunes deben tener vías de comunicación directas con el desarrollador desde el propio teléfono móvil como puede ser una llamada telefónica, mensajería directa o mail.

4.4.1. Introducción a la aplicación

La interacción comienza en el momento que el usuario se topa con la aplicación. Los bizitxapeldunes son informados en el inicio sobre qué es JonBike y sobre su participación en el desarrollo como testers mediante el documento "Guía de los bizitxapeldunes".

Como muchos de los comentarios de los usuarios surgen en el primer contacto y pueden pasar desapercibidos, se debe estar atento y preparado desde el inicio.

4.4.2. Feedback sobre la usabilidad

Dadas las funcionalidades de la aplicación, se les invita a que pongan en marcha la aplicación, pero sin las instrucciones para ello. Se toma nota de cómo los usuarios interactúan con la aplicación.

Los cuestionarios personales serán los que encaminen este procedimiento, asegurando que incluyan los apartados que se quiere mejorar dado el perfil del usuario.

4.4.3. Notificación de novedades

Todos los usuarios notarán que la aplicación es automáticamente actualizada en el momento que se suba una nueva versión, pero no obstante no conocerán las novedades que esta versión pueda traer consigo.

Para cada actualización significativa en cada ciclo de vida, los bizitxapeldunes serán informados. De este modo, se mantiene su nivel de implicación haciéndoles parte de toda la información y prueban las novedades de la aplicación

Capítulo 5.

Arquitectura del servicio

Para entender el funcionamiento de la aplicación JonBike es necesario explicar la arquitectura de la solución final, identificando cada una de sus partes y la forma en la que estos interaccionan para funcionar. En la figura 5.1 se muestra de forma gráfica el conjunto del sistema.

Con MongoLab, tenemos un servidor MongoDB-as-a-Service.

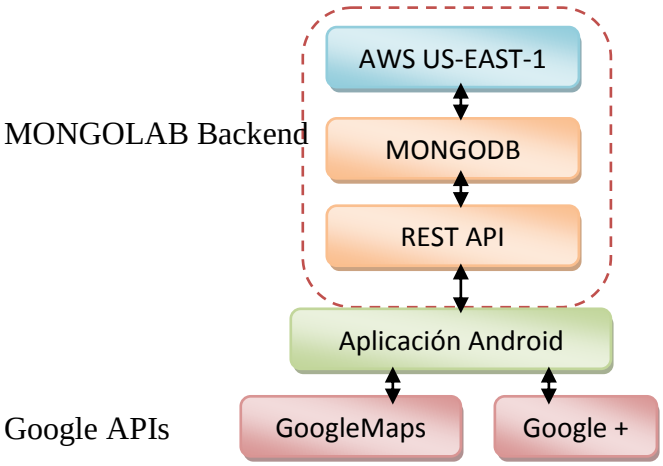


Figura 5.1.: Arquitectura de JonBike

El servidor de la aplicación MongoLab se encuentra alojado en la base Amazon AWS, que ejecuta el backend y proporciona los servidores de MongoDB.

La aplicación en el smartphone Android juega de punto central de la aplicación, que realiza las tareas de geolocalización y visualización de parámetros del entrenamiento sobre los datos GPS obtenidos. Ambos se conectan a través de la REST API como se muestra en la figura 5.1.

El cliente Android hace uso de la REST API de MongoLab, del cual guarda o descarga las rutas almacenadas de los ciclistas en el historial ciclista de la aplicación. También hace uso de dos servicios de Google, la API de GoogleMaps para la visualización del mapa y la ruta en el smartphone, y Google+ para la autenticación de usuarios en la aplicación.

En conclusión, la arquitectura del servicio es del tipo cliente – servidor. Comparando el uno con el otro, habría que destacar que el cliente es el que tiene un mayor peso en el servicio al tener este el estado general de los datos y la información que el usuario visualiza en todo momento. La comunicación es del tipo pull y push al enviar o recoger datos del servidor para completar el historial del ciclista.

5.1. Comunicación entre la aplicación Android y REST API

La comunicación entre los distintos componentes se realiza mediante los protocolos HTTP. La codificación de los datos se puede dar en los formatos HTML o JSON.

Los datos codificados en formato JSON tienen la estructura que se muestra en el extracto de código 5.1.

```
{
    "_id": {
        "$oid": "5492c72ce4b04fa5e44b7811"
    },
    "autor": "100095405794992338453",
    "fecha": "2014-12-18 13:23:04",
    "duracion": "0h 0min 10seg",
    "distancia": "0,02",
    "velomedia": "6,20",
    "velomax": "0,00",
    "altmax": "52,60",
    "calorias": "0,00"
}
```

Extracto de código 5.1.: Ruta codificada en formato JSON.

El proceso consiste en enviar la ruta en formato JSON desde el cliente Android al servidor MongoLab con los datos.

Primero se genera el JSON en el propio cliente tal y como muestra el extracto de código 5.2., para su posterior envío. Haciendo uso del Objeto RUTA, la función createRuta(Ruta ruta) devuelve:

```
return String.format("{\"autor\": \"%s\", \"fecha\": \"%s\", \"duracion\": \"%s\", \"distancia\": \"%s\",
\"velomedia\": \"%s\", \"velomax\": \"%s\", \"altmax\": \"%s\", \"calorias\": \"%s\"}", ruta.autorId,
ruta.fechaHora, ruta.datoDuracion, ruta.datoDistancia, ruta.datoVeloMedia, ruta.datoVeloMax,
ruta.datoAltMax, ruta.datoCal);
```

Extracto de código 5.2.: Transformación a JSON.

Una vez el documento JSON correctamente creado, la clase QueryBuilder construye la URL para guardar dicho documento. La URL se crea de la siguiente manera como muestra el extracto de código 5.3.:

```
public class QueryBuilder {

	/**
	 * Especificacion de la base de datos
	 */
	public String getDatabaseName() {
		return "dbjonbike";
	}
	/**
	 * Especificacion de la clave API de MongoLab
	 */
	public String getApiKey() {
		return "OeD368S2Hork8HqQpq2AnS3kM_iBagqt";
	}
	/**
	 * Construccion de la URL que permite administrar la base de
datos
	 */
	public String getBaseUrl() {
		return "https://api.mongolab.com/api/1/databases/" +
getDatabaseName()
				+ "/collections/";
	}
	/**
	 * Completa el formato de la URL y añade la clave al final
	 */
	public String docApiKeyUrl() {
		return "?apiKey=" + getApiKey();
	}
	/**
	 * Devuelve la coleccion rutasjonbike
	 */
	public String documentRequest() {
		return "rutasjonbike";
	}
	/**
	 * Contruye la URL completa usando los metodos de la clase
	 */
	public String buildContactsSaveURL() {
		return getBaseUrl() + documentRequest() + docApiKeyUrl();
	}
	/**
	 * Contruye la URL completa usando los metodos de la clase
	 */
	public String buildContactsGetURL() {
		return getBaseUrl() + documentRequest() + docApiKeyUrl();

	}
```

```
	/**
	 * Formeta la ruta a JSON
	 */
	public String createRuta(Ruta ruta) {
		return String.format("{\"autor\": \"%s\", \"fecha\":
\"%s\", \"duracion\": \"%s\", \"distancia\": \"%s\", \"velomedia\":
\"%s\", \"velomax\": \"%s\", \"altmax\": \"%s\",
		\"calorias\": \"%s\"}", ruta.autorId, ruta.fechaHora,
ruta.datoDuracion, ruta.datoDistancia, ruta.datoVeloMedia,
ruta.datoVeloMax, ruta.datoAltMax, ruta.datoCal);
	}
}
```

Extracto de código 5.3.: Clase QueryBuilder

Una vez construido el documento, se prepara la conexión HTTP de la siguiente manera como muestra el extracto de código 5.4.

```
public class SaveAsyncTask extends AsyncTask<Ruta, Void, Boolean> {
	@Override
	protected Boolean doInBackground(Ruta... arg0) {

		//@Override
		try {
			Ruta ruta = arg0[0];
			QueryBuilder qb = new QueryBuilder();
			HttpClient httpClient = new DefaultHttpClient();
			HttpPost request = new
HttpPost(qb.buildContactsSaveURL());
			StringEntity params = new
StringEntity(qb.createRuta(ruta));
			request.addHeader("content-type", application/json");
			request.setEntity(params);
			HttpResponse response = httpClient.execute(request);

			if (response.getStatusLine().getStatusCode() < 205) {
				return true;
			} else {
				return false;
			}
		} catch (Exception e) {
			@SuppressWarnings("unused")
			String val = e.getMessage();
			return false;
		}
	}
}
```

Extracto de código 5.4.: Clase SaveAsyncTask

5.2. Comunicación entre Google+ y aplicación Android. Autenticación de usuarios

Para el sistema de autenticación de usuarios, JonBike no implementa un sistema propio. No se guardan credenciales de acceso de ningún usuario para su posterior identificación, sino que se delega esta tarea en el servicio externo Google+. Existen alternativas como Twitter, Facebook o incluso Github, pero se ha optado por la opción de Google+ por conocimiento previo de estos de los servicios externos.

Este servicio se basa en el protocolo OAuth2.0, donde JonBike toma el rol de cliente. Una vez iniciada la aplicación JonBike, se muestra al usuario una pantalla para insertar su id de usuario y su contraseña de acceso. El cliente Android hace uso de la librería Google Play Services. En la figura 5.1 se muestra este proceso de forma gráfica.

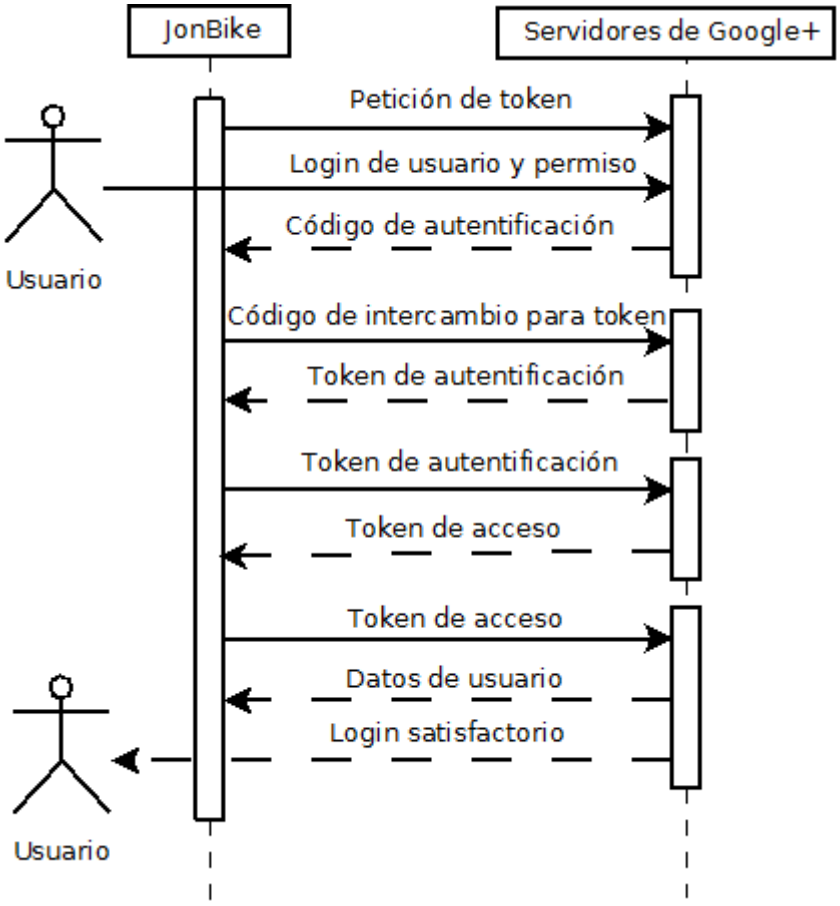


Figura 5.2.: Flujo de autenticación mediante Google+.

Capítulo 6.

Servidor backend

El servidor de JonBike almacena de forma permanente los datos de cada ruta y los clientes los obtienen de este. Este servidor se aloja en Amazon AWS y se basa en la plataforma MongoLab[1].

MongoLab es un servicio completo en la nube de código abierto que aloja bases de datos MongoDB como se ha explicado en el apartado 3.3. La decisión de implantar MongoLab en la aplicación sucedió en la transición del primer ciclo al segundo ciclo de implementación.

Desde el principio se tenía clara la idea de trabajar con software de código abierto y servicios gratuitos pero no se tenía una opción clara. Se decidió trabajar con Heroku por la cantidad de información disponible, por tener una infraestructura basada en Amazon AWS y por soportar Jode.js entre una de sus plataformas de desarrollo, añadiendo que no tiene coste económico alguno utilizando una sola instancia.

También, uno de los añadidos disponibles de Heroku era el tener MongoDB, tecnología popular y de software libre, y que además es la opción de referencia en otros proyectos basados en la geolocalización, perfecto para aplicaciones de tiempo real o que manejan gran cantidad de datos no estructurados en tablas.

En el comienzo de la implementación del backend de la aplicación, se hizo un estudio en profundidad de las tecnologías y se descubre MongoLab, tecnología que implementa una base de datos MongoDB y una conexión directa a través de la REST API de MongoLab.

La decisión de sustituir MongoLab por Heroku junto Node.js fue clara. Era una decisión de elevada importancia, pero no solo se ahorra tiempo de implementación, fallos y pruebas, si no que se obtiene una plataforma con los mismos servicios, de rápida configuración y gratuita.

También es verdad que MongoLab no está provista de la misma cantidad de documentación en comparación con Heroku, pero es el punto en contra de MongoLab. Aun así, el tiempo ganado en comparación con el tiempo calculado que se había propuesto para desarrollar la implementación y su posterior implantación en el sistema como backend de la aplicación Android era de gran diferencia.

1 https://mongolab.com/

6.1. Configuración del servidor

El trabajo de configuración se ha realizado íntegramente en los servidores de Amazon AWS a través de la web del proveedor MongoLab. Una vez iniciada la sesión en la cuenta de MongoLab, accederemos a la lista de bases de datos disponibles en nuestra cuenta. Se muestra a continuación un ejemplo en la figura 6.1.

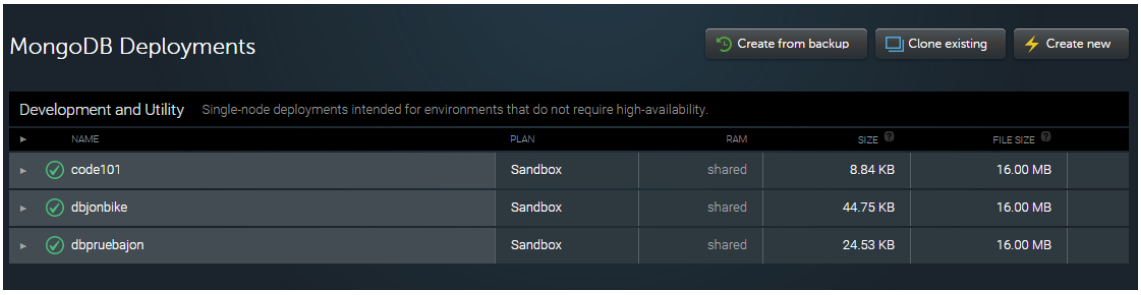


Figura 6.1.: Bases de datos almacenadas en MongoLab.

Para crear una base de datos nueva en el sistema, basta con seleccionar la opción de Create new, como muestra la figura 6.1 en la esquina superior derecha. Una vez seleccionada la opción podremos hacer uso de 0.5GB gratuitos de los siguientes proveedores que muestra la figura 6.2.

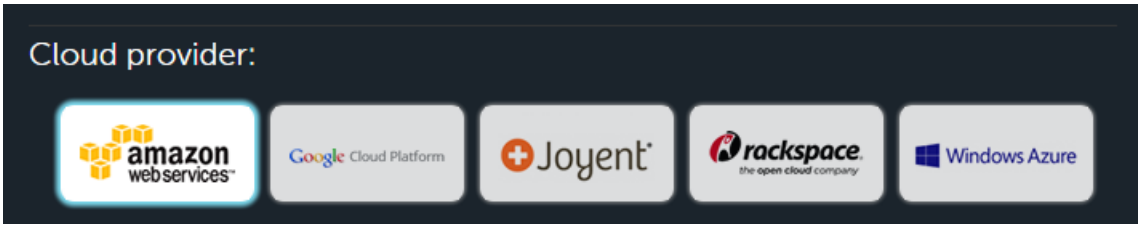


Figura 6.2.: Proveedores de PaaS disponibles.

En este caso, se hace uso de un single-node compartido, basado en Amazon AWS gratuito como muestra la figura 6.3. Una vez elegido el nombre de la base de datos, ya está disponible y preparada para su uso la recién creada base de datos.

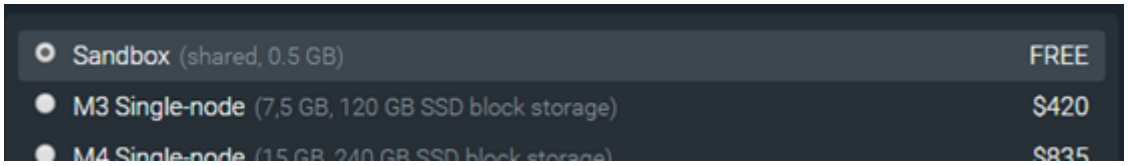


Figura 6.3.: Opciones de planes que oferta MongoLab.

6.2. Configuración de la conexión REST API con la aplicación Android

MongoLab nos da tres modos de conectar a la base de datos de JonBike. La primera de todas es a través de la Shell del sistema. La segunda hace uso del driver de conexión Java a través de la URI estándar como muestra la figura 6.4.

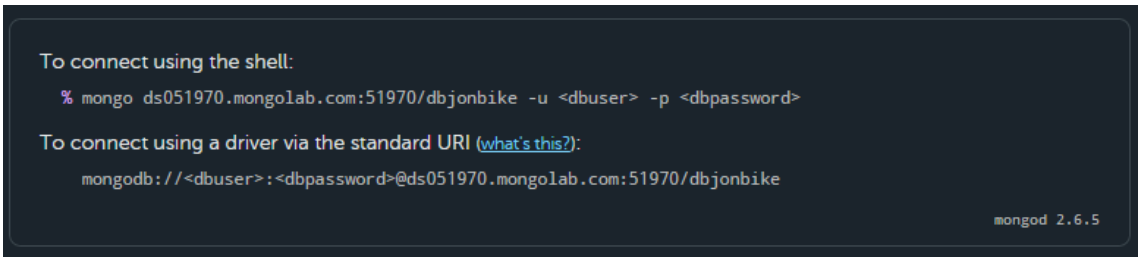


Figura 6.4.: Modos de conexión a la base de datos.

La conexión que se ha implantado en JonBike es la tercera opción, una conexión a través de la REST API del sistema, para que de esta manera, se cree una conexión directa entre la aplicación y el servidor de bases de datos de una manera rápida y sencilla.

Esta conexión viene configurada a través del parámetro API KEY de MongoLab. Con esta clave, a través de la aplicación Android, el sistema es capaz de acceder a cualquier dato de cualquier colección de cualquier base de datos de la cuenta de MongoLab.

Para poder utilizar la API, se debe codificar la clave en el cliente Android para que éste presente dicha clave al servidor en cada petición API. La clave se presenta mediante un parámetro de consulta HTTP llamado 'ApiKey' como muestra la figura 6.5 a continuación.

Figura 6.5.: Ejemplo de código HTTP para la conexión cliente – servidor.

Cada cuenta de usuario tiene una ApiKey para acceso a sus bases de datos. Para mayor seguridad, es posible regenerar la clave para cada usuario en cualquier momento.

En el apartado 5.1 del capítulo acerca de la arquitectura del sistema, se explica de una manera más precisa y detallada la conexión entre el cliente Android y el servidor de bases de datos MongoLab.

6.3. Gestión de la base de datos

Toda la gestión del servidor de la base de datos se realiza a través de la web de MongoLab. Dentro de cada base de datos, en este caso dbjonbike, tenemos una serie de opciones para realizar sobre la colección de datos almacenada.

En la figura 6.6 se muestra una captura de pantalla donde se pueden apreciar las opciones del entorno de MongoLab.

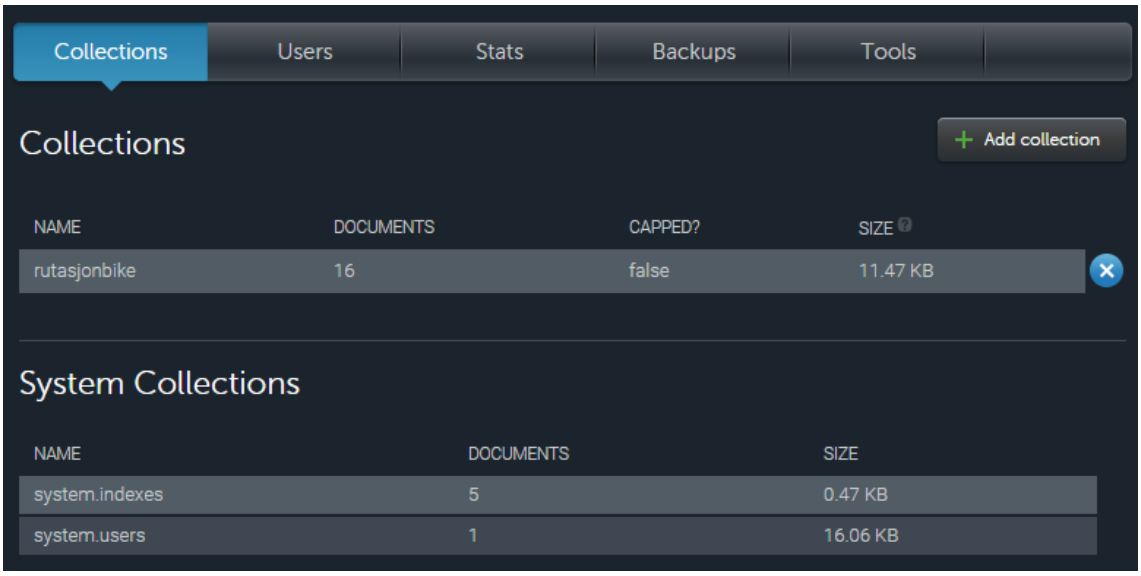


Figura 6.6.: Opciones de entorno para dbjonbike.

Con esta sencilla interfaz gráfica podemos gestionar las colecciones, gestionar los usuarios de la base de datos, visualizar datos estadísticos sobre la base de datos de la aplicación. También se da la opción de gestionar de forma gratuita las copias de seguridad y se nos proporciona una serie de herramientas para la importación o exportación de bases de datos en formato binario, y gestionar las colecciones de la base de datos en tres formas de datos diferentes, en formato binario, formato JSON y formato CSV.

Para cada colección de datos de dbjonbike, se visualizan todos los documentos que la componen, en este caso en formato JSON, con los datos relativos a cada ruta guardada por los usuarios.

Al igual que con las colecciones de la base de datos, para la gestión de documentos de proporcionan las mismas opciones y herramientas. Es posible visualizar datos estadísticos sobre los documento de dicha colección y ejecutar comandos, importaciones y exportaciones desde la interfaz web del servidor MongoLab.

6.4. Pruebas

Las pruebas son un componente fundamental en todo proceso de desarrollo de software. Su finalidad, más allá de ser la de comprobar que todo funciona correctamente, es la de descubrir errores.

Desde el principio, las pruebas realizadas en MongoLab fueron sencillas. Como menciono en la sección 6.2 de este mismo capítulo, las pruebas realizadas se basan en probar la función de la conexión con el cliente Android.

La implementación comenzó con un sencillo tutorial para desarrollar una pequeña base en el sistema a modo de ejemplo de prueba. Al no disponer de toda la documentación disponible en comparación con el resto de alternativas, las pruebas se fueron desarrollando en base a dicho tutorial.

Como se puede apreciar en las secciones anteriores de este capítulo, con MongoLab se ahorra tiempo de implementación con un servidor de bases de datos NoSQL MongoDB sencillo.

Se han realizado pruebas exhaustivas con el cliente Android y MongoLab para el éxito en la transformación de JSON para que no exista problema alguno. Debido a que ante el mínimo error en cualquier envío con formato erróneo de información de ruta en JSON, provoca la pérdida de datos con toda la información de la ruta que ha realizado el usuario.

Al igual que se han realizado pruebas exhaustivas para el envió de información de rutas, se ha trabajado también la conexión y los parámetros relativos a la recepción de información.

Gracias a la sencillez de la interfaz web que proporciona MongoLab, las tareas de prueba han sido relativamente sencillas comparado con la ejecución de pruebas planificada para el backend Heroku descartado.

Capítulo 7.

Cliente Android

Como se ha mencionado en el tercer capítulo, la aplicación se ha implementado mediante Java, haciendo uso de la API nativa de Android. JonBike consta de Activities donde se implementan los casos de uso de la aplicación en distintas partes de la interfaz. Y por otro lado, los servicios en segundo plano, llamados Services, que se detallan en este capítulo más adelante.

7.1. Casos de uso

Los casos de uso de la aplicación de Android se dividen en dos categorías. Por un lado, están todas aquellas operaciones que el usuario ejecuta de forma explícita durante el uso mediante la interfaz gráfica. Por otro, son aquellas operaciones que se ejecutan de forma automática en segundo plano.

7.1.1. Interfaces de usuario

Las interfaces de usuario de Android se implementan mediante las llamadas Activities que contienen toda su lógica y controlan su ciclo de vida. Estas interfaces a su vez pueden estar compuestas por varias Fragments, que son subinterfaces reutilizables con su propio ciclo de vida.

A continuación se presentan las cinco Activities de JonBike: Historial Activity, Identification Activity, Main Activity, MostrarDatosHistorial Activity y Resultados Activity.

IdentificationActivity

Es la interfaz que se muestra al iniciar el programa por primera vez. Se muestra el logo de la JonBike, junto con el botón de inicio de sesión Google+ para iniciar sesión. Este botón comienza la secuencia de autenticación en Google y una vez realizada la autenticación con éxito, redirigir al usuario a MainActivity.

MainActivity

Es la interfaz principal donde se muestran todos los datos relativos al entrenamiento junto con el mapa. Es aquí donde el usuario comienza la sesión de entrenamiento. Junto con los botones de START y STOP para iniciar o terminar el entrenamiento, el usuario

puede seleccionar en el botón de menú, personalizar la aplicación con las opciones de selección de marcador, selección del tipo de mapa y acceso al historial del ciclista. Ésta última opción permite ver una lista de todos los entrenamientos que el usuario ha guardado y lo redirige a HistorialActivity.

En cuanto el usuario presiona el botón de START, JonBike comienza a rastrear una señal GPS válida para comenzar a obtener datos del entrenamiento.

Se hace uso de MapFragment para incluir GoogleMaps en dicha activity, donde se mostrará la ubicación actual junto con las personalizaciones del usuario.

Una vez que el usuario desea terminar el entrenamiento, pulsará el botón de STOP y una vez realizada la confirmación, se le redirige al usuario a ResultadosActivity mostrando los datos del entrenamiento que acaba de realizar.

ResultadosActivity

Esta interfaz muestra todos los datos relativos al entrenamiento realizado. También hace uso de MapFragment para mostrar el mapa de GoogleMaps con el recorrido realizado. Se le permite al usuario configurar el tipo de mapa del resultado para una visión del recorrido según las necesidades del mismo.

El usuario tiene la opción de descartar o de guardar dicho entrenamiento. En el caso de que el usuario desee descartar el entrenamiento, pulsa el botón y se le redirige a MainActivity para volver a empezar un entrenamiento nuevo. Si selecciona la opción de guardar el entrenamiento, se guarda en el servidor dicha información junto con el identificador de usuario y se le redirige de nuevo a MainActivity.

HistorialActivity

Esta interfaz muestra una lista con los entrenamientos realizados por el usuario, ordenados por fecha. Cuando el usuario selecciona algún elemento de la lista se le redirige a MostrarDatosHistorialActivity.

MostrarDatosHistorialActivity

Es la actividad que muestra los detalles del entrenamiento seleccionado en HistorialActivity. Contiene cada detalle de la ruta, la fecha en la que se realizó la ruta, la duración del trayecto junto con la distancia recorrida. La velocidad media del entrenamiento y la velocidad máxima que alcanzó. También incluye la altitud máxima y por último las calorias estimadas.

El usuario tiene la posibilidad de eliminar la ruta seleccionada o volver al historial para visualizar otra ruta.

En la figura 7.1 se muestra la navegación entre ellas.

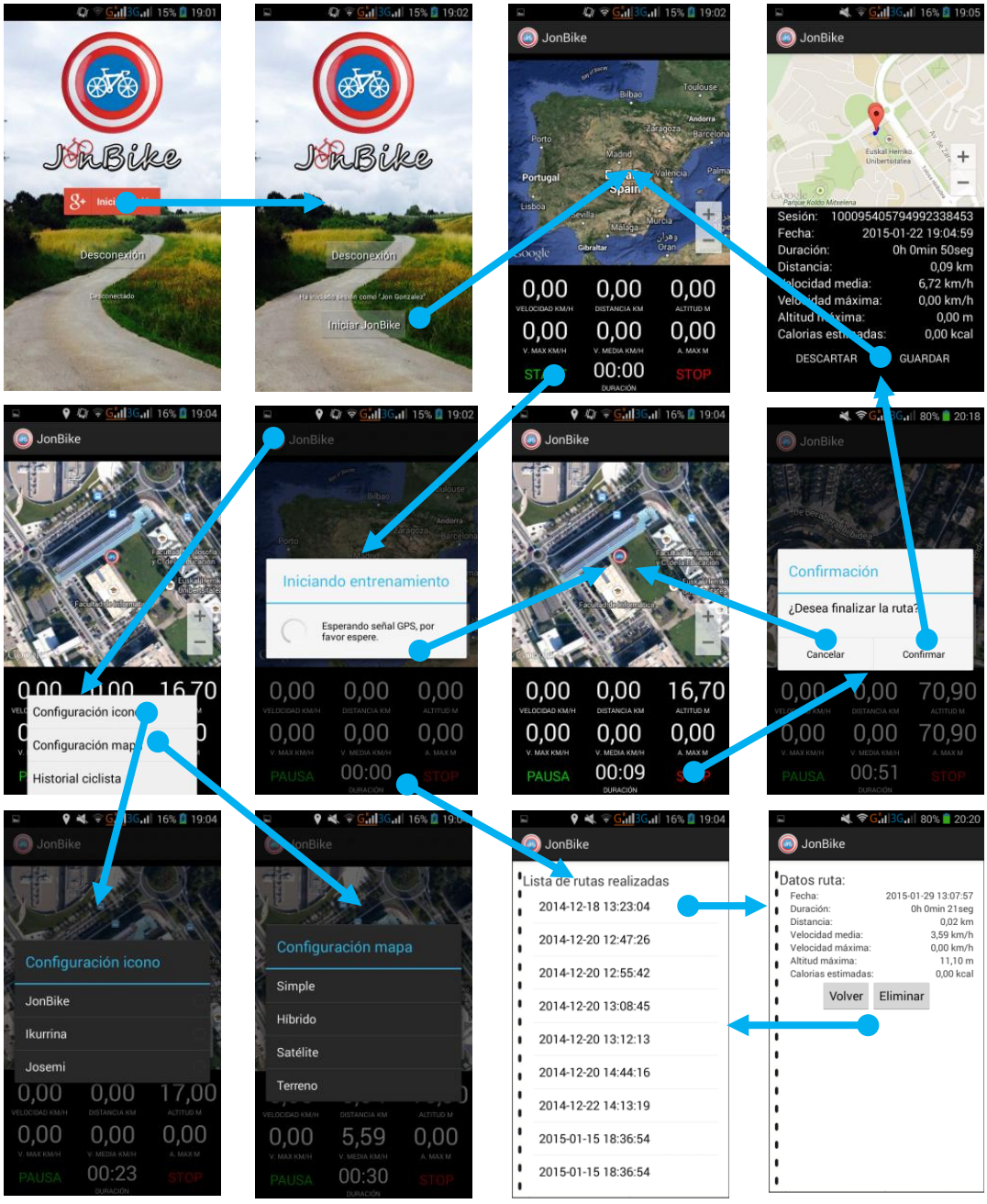


Figura 7.1.: Las pantallas de la interfaz del usuario y el movimiento entre ellas.

7.2.2. Servicios en segundo plano

Existen una única funcionalidad que se encuentra implementada mediante servicios en segundo plano, y que se ejecutan al margen de que la aplicación se encuentre en pantalla o no.

Servicio de localización

Su funcionalidad es la de informar y seguir recopilando datos de posicionamiento a la aplicación de forma periódica.

Se hace uso de la clase LocationManager. Esta clase proporciona acceso a los servicios de localización del sistema. Estos servicios permiten que las aplicaciones obtengan actualizaciones periódicas de la situación geográfica del dispositivo, o para disparar un Intent especifico cuando el dispositivo entra en la proximidad de una zona geográfica determinada.

No se declara una instancia de esta clase directamente, sino que se obtiene a través de Context.getSystemService(Context.LOCATION_SERVICE).

LocationService es un servicio que comienza su ejecución tras el inicio de sesión o automáticamente cuando arranca el sistema operativo Android, si ya se había iniciado sesión anteriormente.

El servicio de localización sobre el que se basa LocationService es el Location APIs de Google Play. Con la finalidad de ahorrar batería, se establecen unos límites sobre la periodicidad en la que se reciben notificaciones del cambio de localización, a la vez que se mantiene un mínimo de precisión filtrando las coordenadas obtenidas. Los criterios son los siguientes:

- Como mínimo el dispositivo se debe haber desplazado 20 metros desde la anterior actualización.

- Se verificará la posición cada segundo, siempre comprobando el criterio de la ubicación recibida. A través del filtro se añade o no a la ruta del usuario.

El servicio está programado para ejecutarse cuando el usuario pulsa START. En el momento de pausa o finalización del entrenamiento cuando el usuario pulsa STOP, se desconecta el servicio de localización Location APIs para dejar de recibir actualizaciones de la posición.

7.3. Pruebas

Las pruebas en el sistema Android son un componente fundamental en el proceso de desarrollo de software. En cada uno de los tres ciclos de desarrollo de la aplicación se han comprobado las funciones de JonBike de dos maneras.

Una, el propio desarrollador a la hora de implementar la aplicación y testeando en su propio smartphone. Y dos, los bizitxapeldunes puestos en situación para testear y recorrer la aplicación de pies a cabeza.

Se han realizado una serie de pruebas unitarias en cada activity de la aplicación, de tal manera que se pudiera validar si el comportamiento de cada acción se correspondía con el esperado.

En el primer ciclo la funcionalidad de la aplicación era sencilla, incorporaba el mapa y la obtención de parámetros a través del GPS del dispositivo. Por falta de experiencia en el ámbito de geolocalización, se realizaron fallos que fueron fácilmente solucionados.

Se realizaron pruebas exhaustivas con la API de Google Maps para no tener errores de visualización de mapa.

Más adelante, en el segundo ciclo, a un bizitxapeldun en concreto se le indicó que hiciese lo posible para detener el funcionamiento de la aplicación, y entre su feedback y el del resto de testers, se obtuvieron mejoras y prevención de fallos para la futura implementación.

Se realizaron pruebas exhaustivas con la conectividad del backend en la aplicación. Como menciono en la sección 6.4 del capítulo seis, la información relativa a MongoLab es escasa. Se han realizado pruebas de conexión con la REST API mediante la ApiKey, la cual dio bastantes problemas ya que por falta de experiencia con JSON, no se enviaba correctamente la información y en consecuencia, se producía la perdida de información.

A través de Google Play Developers Console, se puede hacer un pequeño rastreo de los errores que se generan en la aplicación. Google proporciona su propio sistema de reporte de errores integrado en Google Play.

Si una aplicación ha sido instalada a través de su plataforma, en el momento del error se muestra un mensaje de alerta que ofrece dos opciones: aceptar para cerrar sin realizar ninguna acción, o reportar el informe de error.

Los bizitxapeldunes no reportaban los fallos producidos en la aplicación, por la sensación de pérdida de privacidad que supone o por simple incomodidad de seguir los pasos y reportarlo en el feedback siguiente. Un ejemplo en la figura 7.2.

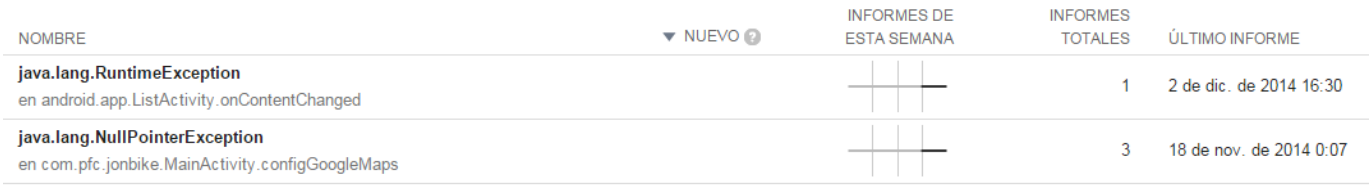


Figura 7.2.: Ejemplo de reporte de error en Google Play Developers Console.

Por los inconvenientes anteriores, como se menciona en el apartado 3.4 del capítulo cuatro, se decidió utilizar un servicio alternativo no dependiente de Google Play Store.

En la figura 7.3 se muestra una captura de Splunk de MINT Express donde se obtiene una visión más gráfica que la de Google Play. Aunque las características eran similares entre los dos, este servicio gratuito proporciona valiosa información apropiada para este proyecto.

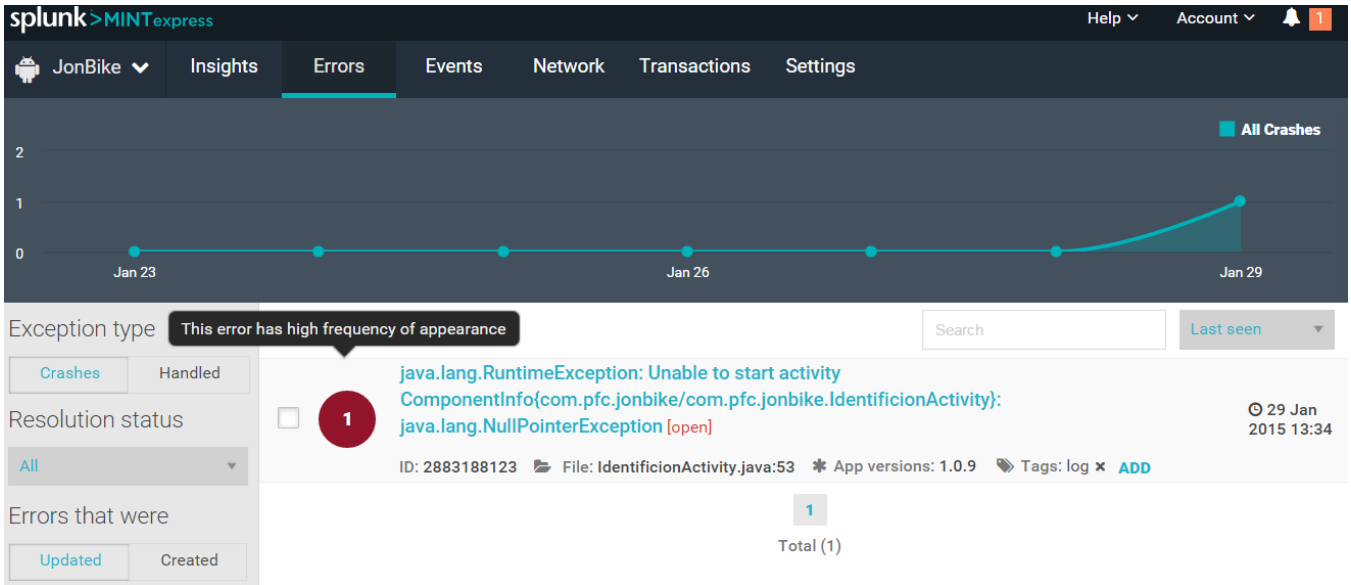


Figura 7.3.: Ejemplo de reporte de error en Splunk de MINT Express.

Capítulo 8.

Gestión del proyecto

JonBike ha seguido la filosofía Lean Startup, que sin una planificación concreta inicial establecida, ha basado su desarrollo en una serie de principios a modo de metaplanificación. Estos principios generales han sido:

- El comienzo mediante un prototipo mínimo y funcional.
- La pronta introducción del feedback de los usuarios en el producto.
- La posterior ampliación progresiva del alcance en base a esta información.

Esta ampliación se ha producido mediante la interacción con los bizitxapeldunes y la integración de sus aportaciones en el proyecto. De la misma forma, la gestión de la calidad se ha realizado con el seguimiento de sus comentarios y críticas, siendo la satisfacción de estos testers el principal indicador.

Siguiendo la misma línea, los bizitxapeldunes han sido considerados la parte interesada más importante del proyecto. Este proceso de comunicación, de recolección de feedback y de síntesis ya se ha tratado más detalladamente en el cuarto capítulo.

En el tercer capítulo también se explican los elementos tecnológicos que se han ido adquiriendo a lo largo de todo el proyecto, los ya han sido están debidamente detallados.

8.1. Gestión del alcance

Tal y como se ha mencionado anteriormente, el alcance del proyecto ha ido ampliándose a medida que este ha ido avanzando como consecuencia del feedback de los bizitxapeldunes.

El alcance del prototipo inicial de la aplicación basada en geoposición era el siguiente:

- Implementación en Android de una aplicación que informa al ciclista sobre su entrenamiento.
- Implementación del backend con node.js.
- Uso de PaaS Heroku.
- Uso de base de datos NoSQL MongoDB.
- Uso de la api de GoogleMaps para la ubicación en mapa.

En la fase de desarrollo del primer ciclo, se modificó el alcance del proyecto estableciendo lo siguiente:

- Implementación en Android de una aplicación que informa al ciclista sobre su entrenamiento.
- Implementación del backend con MongoLab.
- Uso de Base de Datos MongoDB.
- Uso de la api de GoogleMaps para la ubicación en mapa.

Tras el prototipo, y con el feedback recibido de los primeros bizitxapeldunes, se amplió el alcance del segundo ciclo de vida del proyecto con las siguientes características:

- Implementación de propuestas de mejora para la aplicación.
    - Cambio de marcador en la ubicación del mapa.
    - Cambio del tipo de mapa para la visualización de GoogleMaps.
    - Filtro GPS para descartar coordenadas erróneas.
    - Implementar un diseño más atractivo.
    - Sonido para el comienzo y final del entrenamiento.
- Implementación de funcionalidades adicionales.
    - Uso de la api de Google+ para el inicio de sesión y identificación en la aplicación.
    - Añadir al resultado una estimación de las calorías quemadas.
    - Nuevo diseño y datos para mostrar el historial del ciclista.

Después del segundo ciclo, una vez analizado el feedback de los testers, se incluyeron en el alcance del tercer ciclo de vida del proyecto los siguientes puntos:

- Implementación de propuestas de mejora para la aplicación.
    - Que el usuario pueda eliminar las rutas que desee de historial ciclista.
    - Control de los botones STOP y retorno del smartphone para evitar finalizar el entrenamiento por accidente.
    - Implementación de Splunk para la detección de errores de la aplicación.

8.2. Gestión del tiempo

El método de trabajo que se ha llevado a cabo en la elaboración de la aplicación junto al servidor de bases de datos, ha requerido un ritmo de desarrollo y aprendizaje muy exigentes. Las planificaciones a corto plazo eran vitales a la hora de marcar objetivos y obtener resultados a lo largo de los tres ciclos de vida que se han llevado a cabo.

Durante el desarrollo del proyecto han surgido inconvenientes y errores con los que no se contaba al principio por falta de conocimiento y experiencia personal. Por otro lado, también se ha tropezado con soluciones e ideas que han facilitado el desarrollo del producto y en consecuencia, han proporcionado tiempo extra para aquellas tareas en las que se ha invertido más horas de las previstas.

Tras la primera reunión para poner sobre el papel los objetivos del proyecto, los entregables junto con su descripción y las tecnologías implicadas, se estableció como fecha límite el día 24 de septiembre. En dicha reunión de establece la duracion del proyecto, desde el día 15 de septiembre hasta el día 12 de diciembre.

La experiencia previa en el entorno de desarrollo eclipse junto con el ADT Android ha facilitado el desarrollo de la aplicación. El 12 de octubre comenzó el proceso de implementación y un mes después, el 13 de noviembre se publicó el primer prototipo funcional del cliente Android en Google Play. Tras la reunión, adquisición de las aportaciones y pruebas con el bizitxapeldun del primer ciclo se concluye el primer ciclo de fase de desarrollo del proyecto.

Se establecen las nuevas propuestas de mejora para el segundo ciclo de desarrollo de la aplicación y se establece la fecha límite de 27 de noviembre. Un plazo de dos semanas para realizar las mejoras de la versión 1.0. En este momento, es cuando se produce el primer error de planificación. La fecha propuesta, 27 de noviembre, se pospone debido a una intervención quirúrgica con dos semanas de retraso en la planificación. El día 2 de diciembre se publica la segunda versión de JonBike en Google Play Store con las mejoras propuestas y se inicia el periodo de prueba junto con el Plan de bizitxapeldunes. Tras la publicación del cliente Android, comienza de forma inmediata las entrevistas

con los bizitxapeldunes hasta el día 8 de diciembre. Una vez obtenido y analizado el feedback de los bizitxapeldunes, se presenta el día 11 de diciembre el plan de tercera fase con las propuestas de mejora y las funcionalidades adicionales de los bizitxapeldunes.

Una vez realizada la reunión para dar por finalizado el segundo ciclo de vida del proyecto, comienza el tercer ciclo de vida. Se propone añadir un factor de gamificación a la aplicación. Se establece fecha de entrega de la tercera versión de la aplicación para el día 22 de diciembre, junto con la reunión de seguimiento y control el día 23 de diciembre.

Por incompatibilidad de horarios y trabajo, dicha reunión no se pudo llevar a cabo y se decidió realizar resolver las cuestiones pendientes por correo electrónico. Por motivos de trabajo y falta de tiempo, no se pudo dedicar el tiempo necesario al tercer y último ciclo de vida del proyecto. El proyecto se retoma al 50% el día 7 de enero, pero por temas laborales y la mala gestión del tiempo, no se pudo seguir el ritmo necesario para afrontar la tercera fase del proyecto a tiempo.

Una vez realizado el primer borrador de la memoria, el día 27 de enero se establecen nuevos y ajustados objetivos con el fin de conseguir cumplir el nuevo plazo de entrega día 2 de febrero.

El día 28 de enero se entrega el segundo borrador completo y se establecen fechas para cumplir con los objetivos de forma inmediata. Una vez publicada la actualización del cliente Android, comienzan las entrevistas con el nuevo grupo de bizitxapeldunes con el fin de obtener el feedback el día 28 de enero, para ultimar los detalles de implementación y mejora de JonBike.

El mismo día 28, una vez estudiadas y seleccionadas las propuestas de los entrevistados, comienza la implementación y el lanzamiento de la última versión dentro del alcance del proyecto, y fin de las tareas de implementación para el día 29 de enero. La gestión del tiempo del proyecto tendría una desviación como muestra el cuadro 8.1.

	Inicio plan	**Final plan**	**Inicio real**	**Final real**	**Días plan**	**Días Reales**
Ciclo 1	01.10.2014	31.10.2014	12.10.2014	13.11.2014	31 días	31 días
Ciclo 2	01.11.2014	30.11.2014	14.11.2014	11.12.2014	30 días	28 días
Ciclo 3	01.12.2014	12.12.2014	12.12.2014	30.02.2015	12 días	50 días

Cuadro 8.1.: Desviación de fechas planificadas vs fechas reales.

Como se puede apreciar, la gran diferencia y desviación de lo planificado se produce en el tercer ciclo, cuadriplicando la duracion estimada en un principio. También se puede apreciar el retraso en cadena que se produce entre el inicio y final de cada ciclo que al tener fechas y límites ajustados para la fecha de entrega, son desajustes extremadamente críticos para el proyecto.

8.3. Gestión de costes

Los costes económicos que el proyecto ha supuesto se detallan en horas, el coste humano que el proyecto ha supuesto se resume en el siguiente cuadro 8.2. Los costes económicos son despreciables respecto al volumen de trabajo, pero si han supuesto un desembolse a tener en cuenta.

Categoría	Tarea	Dedicación est. (h.)	Dedicación real (h.)
Gestión del proyecto	Adquisición de tecnológicas y servicios	50	60
	Reuniones de control	10	10
	Gestión del alcance	20	30
Backend	Heroku	40	20
	Node.js	40	20
	MongoLab	-	10
	Gestión de conexión	20	30
	Pruebas	40	30
Cliente Android	Diseño gráfico de interfaz	20	20
	Implementación	50	60
	Capa de conexión al backend	20	20
	Corrección de errores	10	20
	Pruebas	50	40
Gestión del feedback	Motivación y promoción	10	10
	Preparación de cuestionarios	10	10
	Entrevistas y síntesis de los comentarios	10	20
Memoria	Redacción	40	50
	TOTAL	**450**	**460**

Cuadro 8.2.: Tabla de dedicación horaria.

Los costes económicos que ha supuesto el proyecto se detallan a continuación. Parte de los costes estaban previstos ya que no se disponía del material ni la formación necesaria para lograr los objetivos. Por otro lado, pueden ocurrir imprevistos como el que se ha sucedido en este caso, donde se han producido daños materiales en el equipo por accidente.

En el siguiente cuadro 8.3 se resumen los costes económicos que han sido necesarios para la realización del proyecto.

Tipo	Elemento	Coste (€)
Formativo	Curso Android Área	575
Formativo	Licencia Google Developers	22
Material	Smartphone Android	110
Material	Placa base Asus	180

Cuadro 8.3.: Costes económicos.

Para visualizar con mayor detalle las diferencias en la gestión de costes de JonBike, se añaden dos histogramas mostrando la diferencia de dedicación de las tareas operativas y de las tareas formativas en el proyecto.

En el primer histograma correspondiente a las tareas operativas, la figura 8.1, con diferencia se aprecia la reducción del coste de horas debido a la elección de MongoLab como backend.

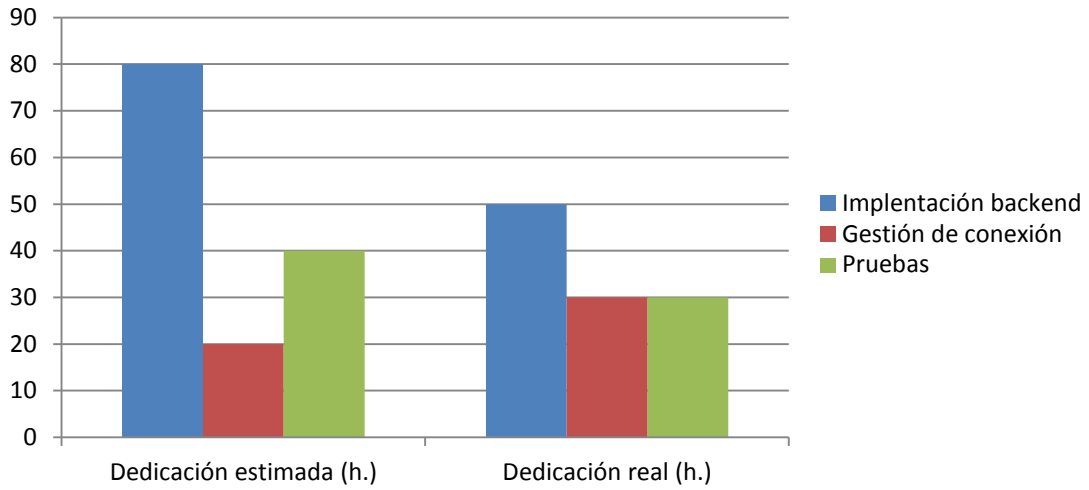


Figura 8.1.: Histograma dedicación tareas operativas.

A diferencia del primer histograma, en la siguiente figura 8.2, se observa la diferencia de coste de horas de las tareas formativas y se aprecia claramente una gran diferencia de exceso de horas en base a las horas que estaban planificadas.

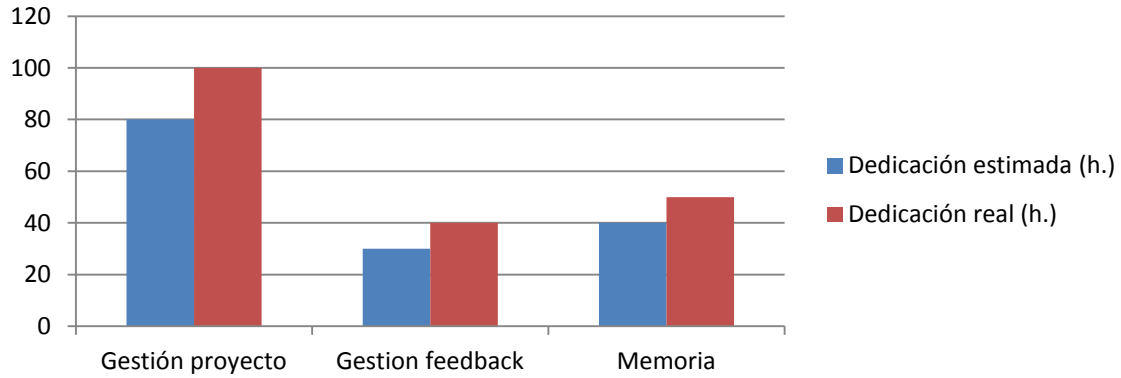


Figura 8.2.: Histograma dedicación tareas formativas

Capítulo 9.

Conclusiones y trabajo futuro

En este último capítulo, se resumen los objetivos conseguidos, se hace una breve valoración personal respecto a la aplicación realizada, y se presentan las líneas principales sobre las que se puede continuar trabajando para extender y mejorar la aplicación.

El objetivo principal del proyecto era la realización de una aplicación para el sistema operativo Android. En concreto, se trataba de desarrollar una aplicación ciclista basada en geolocalización cuyas características han sido definidas por los propios usuarios mediante su participación continua en el proyecto. Por lo tanto, la obtención de resultados se corresponde con lo planteado inicialmente

Se ha podido aprender a desarrollar aplicaciones para Android en un periodo corto de tiempo gracias a la gran cantidad de información que hay disponible, tanto en la red como físicamente. Como es lógico, Google facilita tutoriales y han publicado una API para acelerar la creación de programas por parte de los usuarios. La instalación de los programas es rápida y al utilizar un lenguaje de programación tan conocido como Java han facilitado las cosas.

Relacionado con las APIs de Google, una vez implantada la aplicación en la nube, la labor de configurar y conseguir el acceso con las APIs de Google, sea la API de GoogleMaps o la API de Google+, a base de tutoriales, ha sido difícil de seguir. No por el código a introducir en las diferentes activities de la aplicación sino por la tarea de manejar y gestionar credenciales para los diferentes paquetes de trabajo.

El primero, las claves y credenciales para validar el acceso del entorno de programación y el segundo, la aplicación subida en Google Play Store. Todos los problemas se han producido por falta de experiencia con el manejo de dichos credenciales. No se estimó correctamente la gestión del tiempo y los errores y fallos en la fase de implementación fueron más difíciles de solucionar de lo esperado.

Una vez entendido el sistema y el proceso de autenticación de Google Developers con las APIs de Google, la realización de pruebas fue mucho más rápida y efectiva.

Por el contrario, trabajando con MongoLab no se dispone de la misma cantidad de información y tutoriales de la web oficial como Google. La información y los tutoriales básicos están en internet desarrollados por usuarios con mayor experiencia en la

herramienta. Estos proporcionan información básica, sencilla y aporta lo justo y necesario para la resolución de problemas.

Se ha adquirido una infraestructura del modelo MongoDB–as–a–Service para alojar el backend de JonBike. Este tipo de servidor no requiere apenas configuración, así como la actualización a futuras versiones no es tarea compleja. MongoDB ha sido la base de datos escogida desde el comienzo del proyecto.

El alcance del proyecto ha partido de un prototipo inicial, un Minimum Viable Product, implementado en forma de aplicación Android como cliente y MongoLab como backend.

Con este inicio, como se ha comentado a lo largo de la memoria, la dirección de las posteriores ampliaciones siguiendo las propuestas de mejora ha sido marcada por los bizitxapeldunes con la visión y conocimiento suficientes para aportar propuestas no solo interesantes y creativas, sino viables en gran medida para su inclusión dentro del alcance del proyecto

La participación de estos bizitxapeldunes no ha resultado ser trivial, ya que para optimizar la utilidad de su tiempo se han planificado las distintas entrevistas, acompañadas de cuestionarios que ayuden a centrar la atención de los entrevistados en aquello para lo que han sido entrevistados para su feedback.

Además, la recogida de feedback no ha sido una tarea meramente organizativa. Se han utilizado herramientas tecnológicas que, sin conllevar trabajo al usuario, han recopilado los errores ocurridos durante la ejecución. Para esta tarea se ha utilizado el servicio Splunk, desconocido previamente.

Tras finalizar la implementación de la aplicación, se considera que el resultado final de la misma es el esperado pero aun así hay una gran serie de mejoras que perfeccionarían y mejorarían JonBike añadiéndole un factor de gran importancia, la gamificación.

Gamificación es el empleo de mecánicas de juego en entornos y aplicaciones no lúdicas con el fin de potenciar la motivación, la concentración, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos.

Líneas de trabajo futuro.

Una correcta implementación de estrategias de gamificación permite pasar de la mera conectividad al compromiso, logrando que los miembros de una comunidad, los trabajadores de una empresa, los estudiantes de un instituto, los habitantes de una ciudad participen de manera dinámica y proactiva en acciones que generalmente requieren un esfuerzo de la voluntad.

La gran idea que se podría realizar con JonBike, y la más arriesgada, es crear una red social para dar la opción al usuario de compartir sus rutas con otros usuarios de la aplicación, o explorar las que otros comparten para realizarlas.
La idea no termina de esta manera, la idea consiste también en ubicar en el mapa marcadores en sitios puntuales para que el usuario vaya en bicicleta hasta ellos. Cuantos más sitios marcados en el mapa recorra el usuario, más puntos tendrá.

Toda esta idea de globaliza en el concepto "Los retos de Jon". Que simplemente consiste en una tabla de posiciones donde los usuarios de la aplicación se verán ordenados por número de puntos, en qué posición del ranking están colocados.

De este modo, no solo se pretende motivar a los usuarios ciclistas a mejorar a partir de sus datos de entrenamiento, sino a recorrer y visitar rincones ciclistas, que quizás eran desconocidos, por todo el mapa.

La ventaja de "Los retos de Jon" no solo es el factor de socializar y añadir a la aplicación el factor de gamificación que le falta, sino conseguir una aplicación con la que poder entrenar de una manera diferente. Visualizando en tiempo real a cada usuario en el mapa junto con su icono personalizado y contribuir con el bien común en las calles.

Al conectar a los ciclistas entre sí, se podrían crear comunidades de ciclistas locales más fácilmente para mejorar la calidad de sus entrenamientos. Esto puede ayudar a la gente y motivar al que lo necesita para seguir sus rutas o entrenamientos de continuo.

También podría tomar un rol más activo al compartir alertas sobre peligros o cualquier otro obstáculo que encuentren en el camino, y así dar a los demás usuarios en su área una idea de qué hay más adelante creando de esta manera un mapa en vivo del mundo ciclista.

Bibliografía y referencias

Joan Ribas Lequerica. Manual Imprescindible de Desarrollo de aplicaciones para Android (2013). Ediciones Anaya Multimedia.

Ibai Valencia Itoitz. Trabajo fin de Máster: Análisis de soluciones innovadoras para desarrollar aplicaciones cliente-servidor con tratamiento avanzado de información en la nube (2013). Tesis de Máster. Máster y Doctorado en Sistemas Informáticos Avanzados. Facultad de Informática de Donostia – San Sebastián. UPV-EHU.

Gorka Maiztegi Etxeberria. Documento de memoria: Faborez (2014). Proyecto Fin de Carrera de Ingeniería en Informática. Facultad de Informática de Donostia – San Sebastián. UPV-EHU.

Wei-Meng Lee. Beginning Android Application Development. Wiley Publishing 2011.

Reto Meier. Professional Android 4 Application Development. John Wiley & Sons 2012.

Francisco Jordán Teruel. Documento de memoria: Estudio de la plataforma Android para dispositivos móviles y desarrollo de aplicación para la administración de redes de sensores inalámbricos (2010). Escuela Politécnica Superior. Universidad Carlos III de Madrid.

Guía de Instalación de la API de Android en Eclipse.
URL: http://developer.android.com/sdk/index.html

Sitio web oficial de Stackoverflow, foro sobre todo tipo de dudas de programación.
URL: http://stackoverflow.com/

Sitio web oficial de GitHub, sitio web que aloja proyectos utilizando el sistema de control de versiones Git.
URL: https://github.com/

Página oficial de MongoLab.
URL: https://mongolab.com/

Tutorial para desarrollo y conexión con MongoLab.
URL:https://michaelkyazze.wordpress.com/2014/05/18/android-mongodb-mongolab-hosted-sample-app-part-one/
URL:http://michaelkyazze.wordpress.com/2014/09/22/android-mongodb-mongolab-hosted-sample-app-part-two/

Sitio web oficial de Android Developers.
URL: https://developer.android.com/

Sitio web oficial de Google Developers.
URL: https://developers.google.com/maps/?hl=es

Tutorial sobre Google+ y su implementación.
URL: http://www.androidhive.info/2014/02/android-login-with-google-plus-account-1

Sitio web oficial Android Developers, información widget Chronometer.
URL: http://developer.android.com/reference/android/widget/Chronometer.html

Tutorial, How to use MongoDB to develop Android Apps - App Development Channel.
URL: https://www.youtube.com/watch?v=ivJGhjGKlhI

Tutorial, Learn How to Build Your Mobile Back-End with MongoDB.
URL: https://www.youtube.com/watch?v=psiyHH17DBg

Sitio web oficial Google Developers, información Google+ sign-in.
URL: https://developers.google.com/+/mobile/android/sign-in

Mikel Niño. «Cinco claves sobre entrevistas de "Customer Development" para que el cliente sea PARTE de la creación de tu startup».
URL:http://www.mikelnino.com/2013/12/cinco-claves-entrevistas-Customer-Development-cliente-PARTE-de-la-creacion-de-tu-startup.html

Sitio web, información sobre la gamificación.
URL: http://www.gamificacion.com

Sitio web oficial, Splunk: Monitorización del rendimiento de la aplicación.
URL: https://mint.splunk.com

Glosario y acrónimos

Activity: Clase de Android que implementa las interfaces de usuario y controla su ciclo de vida.

Amazon AWS: Proveedor de servicios IaaS. Web: http://aws.amazon.com

API: API: Acrónimo de Application Programming Interface. Consiste en un conjunto de llamadas que ofrecen acceso a funciones y procedimientos representando una capa de abstracción para el desarrollador.

App: Aplicación, generalmente para dispositivos móviles.

Backend: Parte interna del software, que procesa los datos según la lógica de la aplicación.

Biblioteca: Agrupación de código que proporcionan servicios a programas independientes pasando a formar parte de éstos. Permiten la distribución de funcionalidades y la construcción modular. También conocido como librería.

Dispositivo móvil: Aparato electrónico de reducido tamaño que ofrece capacidades restringidas de computación y memoria así como elementos hardware que aportan funcionalidad al conjunto.

Eclipse: Es un entorno de desarrollo integrado de código abierto y multiplataforma. Fue desarrollado por IBM aunque actualmente su desarrollo recae sobre la Fundación Eclipse.

Fragment: Interfaz secundaria de Android, que puede ser reutilizada en varios Activities.

Framework: Estructura conceptual y tecnológica compuesta por herramientas, librerías, módulos, convenciones y una metodología de trabajo, que tiene como fin organizar y facilitar el desarrollo.

Git: Software de control de versiones distribuido, desarrollado por Linus Torvalds.

GPS: Siglas de Global Positioning System o Sistema de Posicionamiento Global. Es un sistema de navegación que mediante satélites permite ubicar un elemento en una latitud y longitud con un pequeño error de precisión.

Heroku: Proveedor PaaS basado en Amazon AWS que da soporte a distintas plataformas de desarrollo.

Intent: Descripción abstracta de una operación que se va a llevar a cabo, una clase que permite especificar una Activity a ejecutar, llamando a uno de los métodos de la clase Activity con ese Intent de parámetro.

Interfaz: Se refiere a la abstracción que un determinado elemento o conjunto de elementos realiza sobre sí mismo, facilitando el uso y acceso por otros elementos externos.

MongoDB: Base de datos NoSQL de almacenamiento de documentos.

MVP: "Minimum Viable Product" (MVP) o "Producto Mínimo Viable", una de las herramientas básicas dentro de la práctica de Lean Startup.
URL: http://www.mikelnino.com/2014/09/minimum-viable-product-mvp-como-tactica-no-como-producto.html

MySQL: Sistema de gestión de bases de datos SQL, de código libre.

Node.js: Plataforma de desarrollo de servidores basado en JavaScript.

NoSQL: Base de datos no relacional, que no utiliza el lenguaje SQL.

PaaS: Platform–as–a–Service.
URL: http://en.wikipedia.org/wiki/Platform_as_a_service

REST: Relational State Transfer. Técnica de arquitectura software para el desarrollo de sistemas web distribuidos.

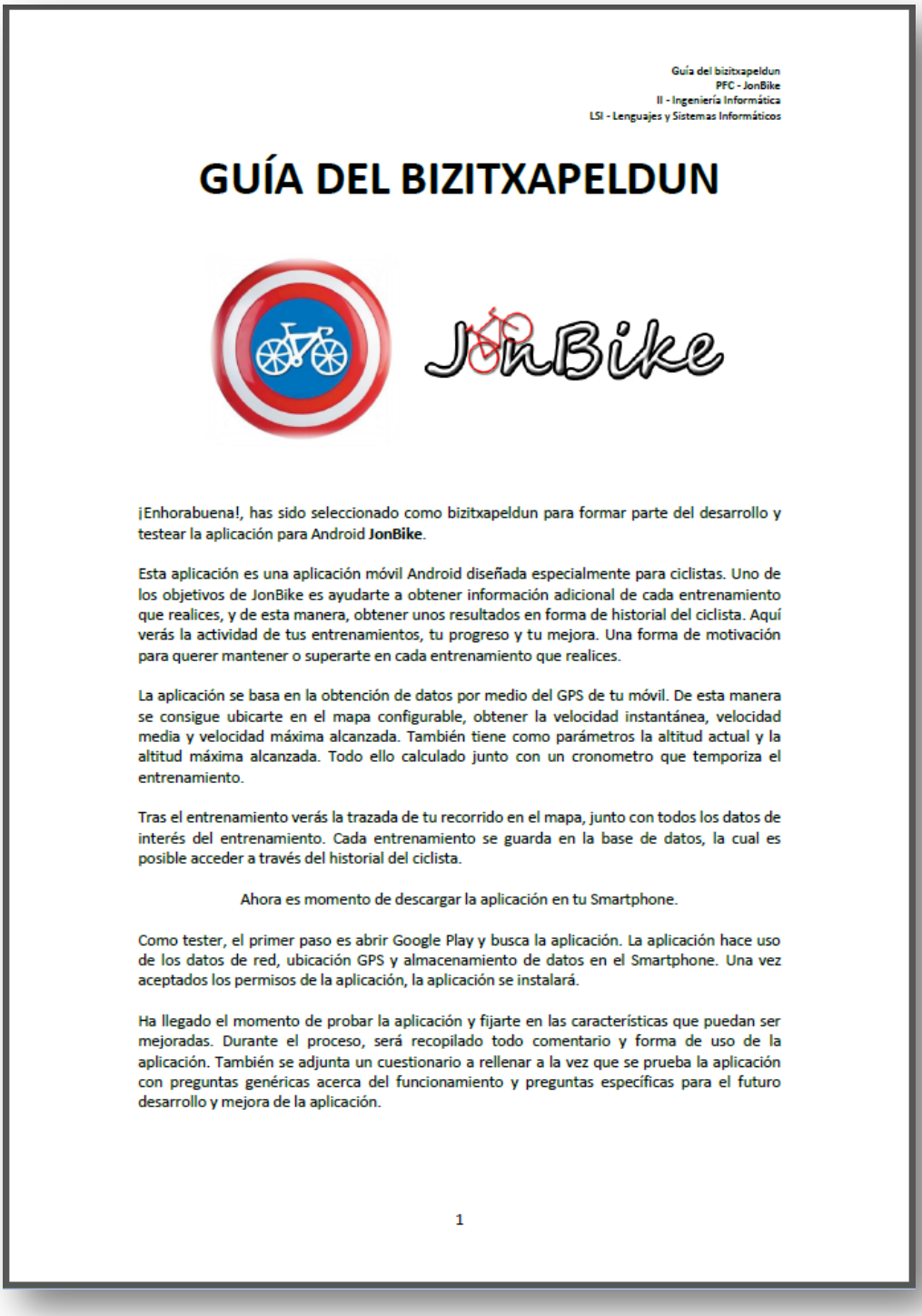
SaaS: Software–as–a–Service.
URL: http://en.wikipedia.org/wiki/Software_as_a_service

SQL: Structured Query Language.

Apéndice A

En el siguiente apartado de muestran los formatos y ejemplos de los documentos mencionados en el apartado 4.3 del capítulo cuatro de la memoria.

La **guía del bizitxapeldun** consiste en un documento breve y preciso que describe, por una parte, la función de JonBike y las características que lo componen. Por otra, las pautas de actuación definidas como testers: invitación a utilizar la aplicación, a fijarse en las características que puedan ser mejoradas y a cómo transmitir el feedback.



Guía del bizitxapeldun
PFC - JonBike
II - Ingeniería Informática
LSI - Lenguajes y Sistemas Informáticos

GUÍA DEL BIZITXAPELDUN

¡Enhorabuena!, has sido seleccionado como bizitxapeldun para formar parte del desarrollo y testear la aplicación para Android **JonBike**.

Esta aplicación es una aplicación móvil Android diseñada especialmente para ciclistas. Uno de los objetivos de JonBike es ayudarte a obtener información adicional de cada entrenamiento que realices, y de esta manera, obtener unos resultados en forma de historial del ciclista. Aquí verás la actividad de tus entrenamientos, tu progreso y tu mejora. Una forma de motivación para querer mantener o superarte en cada entrenamiento que realices.

La aplicación se basa en la obtención de datos por medio del GPS de tu móvil. De esta manera se consigue ubicarte en el mapa configurable, obtener la velocidad instantánea, velocidad media y velocidad máxima alcanzada. También tiene como parámetros la altitud actual y la altitud máxima alcanzada. Todo ello calculado junto con un cronometro que temporiza el entrenamiento.

Tras el entrenamiento verás la trazada de tu recorrido en el mapa, junto con todos los datos de interés del entrenamiento. Cada entrenamiento se guarda en la base de datos, la cual es posible acceder a través del historial del ciclista.

Ahora es momento de descargar la aplicación en tu Smartphone.

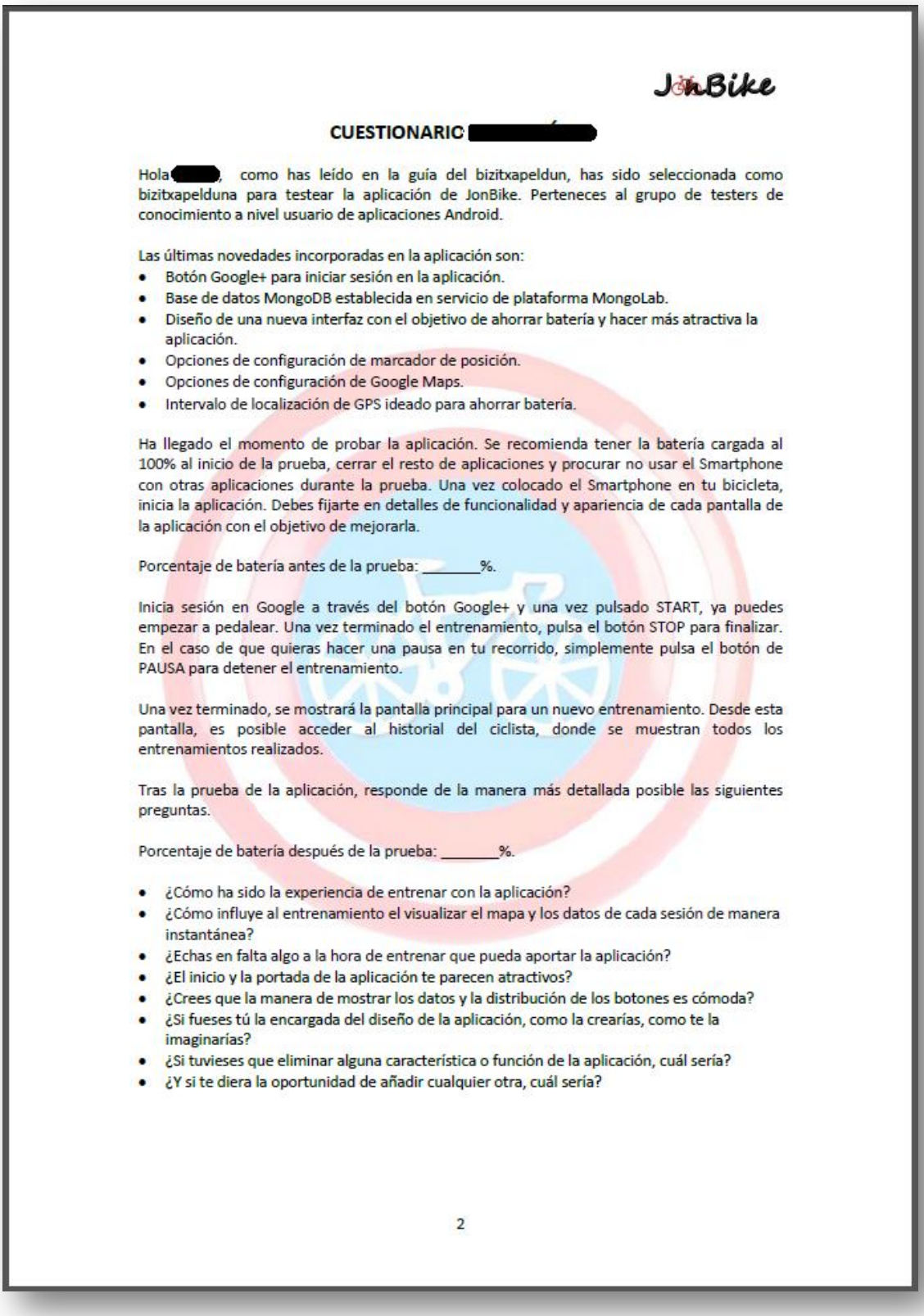
Como tester, el primer paso es abrir Google Play y busca la aplicación. La aplicación hace uso de los datos de red, ubicación GPS y almacenamiento de datos en el Smartphone. Una vez aceptados los permisos de la aplicación, la aplicación se instalará.

Ha llegado el momento de probar la aplicación y fijarte en las características que puedan ser mejoradas. Durante el proceso, será recopilado todo comentario y forma de uso de la aplicación. También se adjunta un cuestionario a rellenar a la vez que se prueba la aplicación con preguntas genéricas acerca del funcionamiento y preguntas específicas para el futuro desarrollo y mejora de la aplicación.

1

Los **cuestionarios personales** son los cuestionarios donde se introducen inicialmente los cambios que ha habido desde la última entrevista para poner en contexto al usuario.

Posteriormente, se enumeran las acciones que debería realizar junto con las preguntas sobre estas características centrando la atención en puntos deseados. Esta dirección se ha llevado a cabo utilizando unos cuestionarios preparados antes de las entrevistas, para ser entregados en ellas. A modo de ejemplo:



JonBike

CUESTIONARIO [redacted]

Hola [redacted], como has leído en la guía del bizitxapeldun, has sido seleccionada como bizitxapelduna para testear la aplicación de JonBike. Perteneces al grupo de testers de conocimiento a nivel usuario de aplicaciones Android.

Las últimas novedades incorporadas en la aplicación son:

- Botón Google+ para iniciar sesión en la aplicación.
- Base de datos MongoDB establecida en servicio de plataforma MongoLab.
- Diseño de una nueva interfaz con el objetivo de ahorrar batería y hacer más atractiva la aplicación.
- Opciones de configuración de marcador de posición.
- Opciones de configuración de Google Maps.
- Intervalo de localización de GPS ideado para ahorrar batería.

Ha llegado el momento de probar la aplicación. Se recomienda tener la batería cargada al 100% al inicio de la prueba, cerrar el resto de aplicaciones y procurar no usar el Smartphone con otras aplicaciones durante la prueba. Una vez colocado el Smartphone en tu bicicleta, inicia la aplicación. Debes fijarte en detalles de funcionalidad y apariencia de cada pantalla de la aplicación con el objetivo de mejorarla.

Porcentaje de batería antes de la prueba: _____%.

Inicia sesión en Google a través del botón Google+ y una vez pulsado START, ya puedes empezar a pedalear. Una vez terminado el entrenamiento, pulsa el botón STOP para finalizar. En el caso de que quieras hacer una pausa en tu recorrido, simplemente pulsa el botón de PAUSA para detener el entrenamiento.

Una vez terminado, se mostrará la pantalla principal para un nuevo entrenamiento. Desde esta pantalla, es posible acceder al historial del ciclista, donde se muestran todos los entrenamientos realizados.

Tras la prueba de la aplicación, responde de la manera más detallada posible las siguientes preguntas.

Porcentaje de batería después de la prueba: _____%.

- ¿Cómo ha sido la experiencia de entrenar con la aplicación?
- ¿Cómo influye al entrenamiento el visualizar el mapa y los datos de cada sesión de manera instantánea?
- ¿Echas en falta algo a la hora de entrenar que pueda aportar la aplicación?
- ¿El inicio y la portada de la aplicación te parecen atractivos?
- ¿Crees que la manera de mostrar los datos y la distribución de los botones es cómoda?
- ¿Si fueses tú la encargada del diseño de la aplicación, como la crearías, como te la imaginarías?
- ¿Si tuvieses que eliminar alguna característica o función de la aplicación, cuál sería?
- ¿Y si te diera la oportunidad de añadir cualquier otra, cuál sería?

2

Las **actas de feedback** proporcionan información sobre la opinión y experiencia del usuario. Los comentarios, aportaciones y cualquier otro dato interesante aportado se anotan en el acta. Al ser un documento directo del feedback, es de gran importancia no realizar filtro de los comentarios y recibir todo aporte y valoración tal cual ha sido transmitido por el bizitxapeldun. A modo de ejemplo:

Acta de feedback [redacted] Ciclo 2 – Grupo usuarios

¿Cómo ha sido la experiencia de entrenar con la aplicación?

- Muy buena, así llevo un recuento de los km que realizo diariamente y el tiempo que tardo en realizarlo y gracias al "historial del ciclista", puedo tener un control preciso de cada salida y de mis mejoras a nivel tiempo en las mismas.

¿Cómo influye al entrenamiento el visualizar el mapa y los datos de cada sesión de manera instantánea?

- Me motiva mucho. Sobre todo a la hora de visualizar mis mejoras en los tiempos en los mismos recorridos.

¿Echas en falta algo a la hora de entrenar que pueda aportar la aplicación?

- Creo que los datos que se reflejan en la aplicación son los básicos para un entrenamiento. Se puede añadir una estimación de las calorías quemadas aunque sea un dato secundario.

¿El inicio y la portada de la aplicación te parecen atractivos?

- No entiendo qué es ni para qué sirve la parte inferior de la portada. Me parece muy feo. El logo queda desdibujado en relación a la foto. El "Ha iniciado sesión como..." debería estar más centrado. Sin embargo, la foto me gusta mucho. Me parece muy chulo que se puedan configurar los iconos.

¿Crees que la manera de mostrar los datos y la distribución de los botones es cómoda?

- Sí, me parece muy sencillo e intuitivo. Además, es muy cómodo el poder marcar los botones inferiores mientras voy encima de la bici sin necesidad de pararme.

¿Si fueses tú la encargada del diseño de la aplicación, como la crearías, como te la imaginarías?

- No me gusta la forma de entrar en la aplicación, es decir, los botones de la página de inicio. Además la bici que aparece en JonBike tapa la letra N, sería más correcto que no se tapase el nombre de la aplicación en la página de inicio. También, probablemente, pondría más iconos para poder personalizar aún más la aplicación. Quizás, con dibujos para los niños, o fotos de tu bici, etc.

¿Si tuvieses que eliminar alguna característica o función de la aplicación, cuál sería?

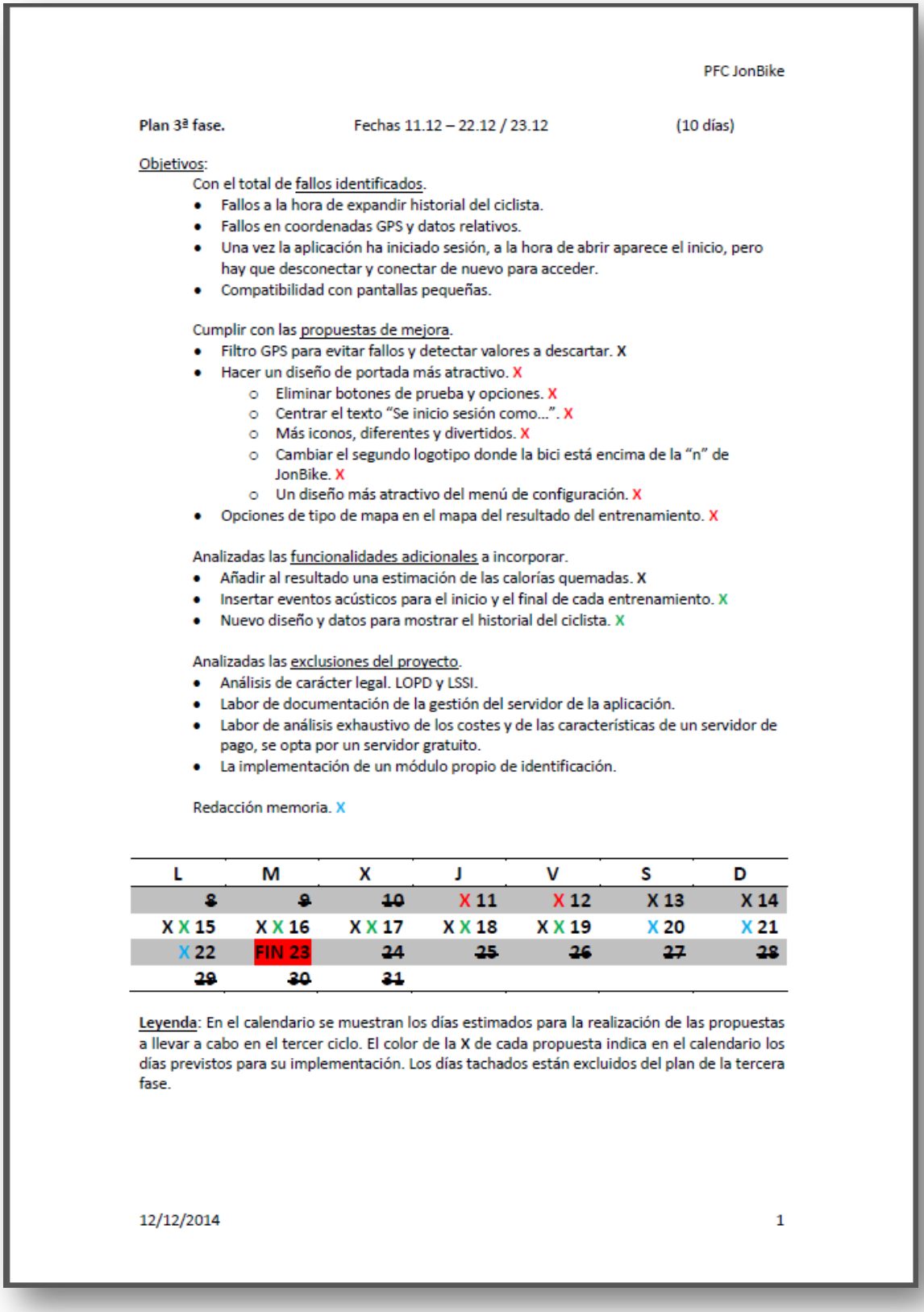
- Nada, me parece que tiene lo justo y necesario.

¿Y si te diera la oportunidad de añadir cualquier otra, cuál sería?

- Nada, me parece una aplicación estupenda. Tiene lo justo y no se "rellena" con elementos absurdos que no se utilizan (como sucede en otras aplicaciones).

1

Las **propuestas de mejora** se realizan al final de cada ciclo. Una vez analizadas su viabilidad, unificadas las similares y adaptadas a las posibilidades, son exportadas a un documento que recoge las propuestas de mejora que se llevarán a cabo en un futuro. Este documento es dinámico, ya que se irán añadiendo a la lista nuevas propuestas, se descartarán algunas anteriores o nuevas ideas pueden superar y reemplazar a las anteriores. A modo de ejemplo, la propuesta de mejora para la tercera fase del proyecto:



PFC JonBike

Plan 3ª fase. Fechas 11.12 – 22.12 / 23.12 (10 días)

Objetivos:

Con el total de fallos identificados.

- Fallos a la hora de expandir historial del ciclista.
- Fallos en coordenadas GPS y datos relativos.
- Una vez la aplicación ha iniciado sesión, a la hora de abrir aparece el inicio, pero hay que desconectar y conectar de nuevo para acceder.
- Compatibilidad con pantallas pequeñas.

Cumplir con las propuestas de mejora.

- Filtro GPS para evitar fallos y detectar valores a descartar. X
- Hacer un diseño de portada más atractivo. X
  - Eliminar botones de prueba y opciones. X
  - Centrar el texto "Se inicio sesión como...". X
  - Más iconos, diferentes y divertidos. X
  - Cambiar el segundo logotipo donde la bici está encima de la "n" de JonBike. X
  - Un diseño más atractivo del menú de configuración. X
- Opciones de tipo de mapa en el mapa del resultado del entrenamiento. X

Analizadas las funcionalidades adicionales a incorporar.

- Añadir al resultado una estimación de las calorías quemadas. X
- Insertar eventos acústicos para el inicio y el final de cada entrenamiento. X
- Nuevo diseño y datos para mostrar el historial del ciclista. X

Analizadas las exclusiones del proyecto.

- Análisis de carácter legal. LOPD y LSSI.
- Labor de documentación de la gestión del servidor de la aplicación.
- Labor de análisis exhaustivo de los costes y de las características de un servidor de pago, se opta por un servidor gratuito.
- La implementación de un módulo propio de identificación.

Redacción memoria. X

L	M	X	J	V	S	D
~~8~~	~~9~~	~~10~~	X 11	X 12	X 13	X 14
X X 15	X X 16	X X 17	X X 18	X X 19	X 20	X 21
X 22	FIN 23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~
~~29~~	~~30~~	~~31~~				

Leyenda: En el calendario se muestran los días estimados para la realización de las propuestas a llevar a cabo en el tercer ciclo. El color de la X de cada propuesta indica en el calendario los días previstos para su implementación. Los días tachados están excluidos del plan de la tercera fase.

12/12/2014 1