

BTT ROUTE

Desarrollo de una app para crear y compartir rutas btt.

Nombre Estudiante: Christian Sanz Novellón
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Nombre Consultor/a: Albert Mata Guerra
Profesor/a responsable de la asignatura: Carles Garrigues Olivella

01/2018



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Desarrollo de app tracker para BTT</i>
Nombre del autor:	<i>Christian Sanz Novellón</i>
Nombre del consultor/a:	<i>Albert Mata Guerra</i>
Nombre del PRA:	<i>Carles Garrigues Olivella</i>
Fecha de entrega (mm/aaaa):	01/2018
Titulación::	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>Ejercicio, rutas, btt</i>
<p>Resumen del Trabajo (máximo 250 palabras): <i>Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.</i></p>	
<p>El objetivo principal de este proyecto, es realizar una aplicación en la que el usuario pueda registrar una ruta con su bicicleta de montaña. Además podrá compartir con otros usuarios la ruta creada con sólo subirla al servidor.</p> <p>Aparte de crear nuestras propias rutas, podremos descargar otras realizadas por diferentes usuarios, cargar archivos GPX procedentes de otras fuentes y exportar en el mismo formato para que se puedan cargar en diferentes dispositivos.</p> <p>Al finalizar cada actividad, la aplicación nos mostrará la ruta realizada y un conjunto de datos relacionados con el ejercicio realizado: distancia, velocidad media, velocidad máxima, desniveles y duración total del recorrido.</p> <p>Para desarrollar la aplicación se utilizará Android Studio. Para el almacenamiento y transferencia de datos, se ha optado por la plataforma Google Cloud. En dicha plataforma tendremos el <i>backend</i> y la base datos MySQL.</p> <p>Los resultados obtenidos en el desarrollo de la aplicación han cumplido los objetivos presentados. No obstante al principio han surgido algunas complicaciones referente a la gestión del mapa y a la obtención de los datos generados en la ruta.</p> <p>Además, ha sido positivo aprender cómo funcionan algunas herramientas necesarias como Google Cloud y como gestionan los datos aplicaciones similares a la desarrollada.</p>	

Abstract (in English, 250 words or less):

The main objective of this project it's to make an application with which the user can register a route with his mountain bike. Also, you can share with other users the route created with only uploading it to server.

Apart from creating our own routes, we can download others made by different users, load GPX files from other sources and export in the same format so they can be loaded on different devices.

At the end of each activity, the application will show us the rout taken and a set of data related to the exercise performed, as for example: distance, average speed and maximum speed, accumulated difference in levels and total duration of the route.

Android Studio will be used to develop the application. For data storage and data transfer, we have opted to Google Cloud platform. In this platform we will have the backend and database MySQL.

The results obtained in the development of the application have fulfilled the presented objectives. However at the beginning some complications have arisen regarding the management of the map and the data obtained during the route.

In addition, it has been positive to learn about how some necessary tools like Google Cloud work and how they manage similar data to the developed applications.

Índice

1. Introducción	1
1.1 Contexto y justificación del Trabajo	1
1.2 Objetivos del Trabajo	2
1.3 Enfoque y método seguido	2
1.4 Planificación del Trabajo	3
1.5 Breve resumen de productos obtenidos	7
1.6 Breve descripción de los otros capítulos de la memoria	7
2. Diseño Centrado en el Usuario (DCU)	8
2.1 Usuarios y contexto de uso	8
2.1.1 Usuarios	8
2.1.2 Contexto de uso	12
3. Diseño conceptual	13
4. Prototipado	15
5. Evaluación	20
5.1 Definición de los casos de uso	20
5.2. Diseño de la arquitectura	24
6. Implementación	28
6.1 Introducción	28
6.2 Herramientas utilizadas	28
6.3 Librerías y API	34
6.4 Desarrollo	37
6.4.1 Estructura del módulo “app”	37
6.4.2 Estructura del módulo “backendBttRoute”	40
6.5 Pruebas realizadas	42
6.6 Cambios y mejoras	45
7. Conclusiones	47
8. Glosario	49
9. Bibliografía	50
10. Anexos	52
10.1 Compilación y ejecución	52
10.2 Instalación	53
10.3 Manual de usuario	53

10.3.1 Registro de un nuevo usuario	53
10.3.2 Login.....	54
10.3.3 Recuperar contraseña.....	54
10.3.4 Cambio de contraseña	55
10.3.5 Crear una ruta nueva	56
10.3.6 Exportar rutas.....	57
10.3.7 Compartir ruta	57
10.3.8 Ampliar mapa	57
10.3.9 Filtrar rutas	58

Lista de figuras

- Figura 1. Aplicaciones disponibles para el registro y seguimiento de la actividad física.
- Figura 2. Unidades vendidas de dispositivos.
- Figura 3. Diagrama de Gantt.
- Figura 4. Pantalla de “Login”.
- Figura 5. Pantalla de “Registro de usuario”.
- Figura 6. Pantalla para solicitar una contraseña nueva.
- Figura 7. Pantalla “Mis rutas” sin datos.
- Figura 8. Pantalla “Mis rutas” con datos.
- Figura 9. Pantalla para grabar ruta.
- Figura 10. Pantalla para grabar ruta en funcionamiento.
- Figura 11. Pantalla de detalle de ruta finalizada.
- Figura 12. Pantalla para guardar ruta.
- Figura 13. Pantalla de detalle ruta creada por el usuario.
- Figura 14. Pantalla filtros de búsqueda de rutas subidas.
- Figura 15. Pantalla de lista de rutas subidas.
- Figura 16. Pantalla de detalle de ruta subida por otro usuario.
- Figura 17. Pantalla de visor de ruta ampliada.
- Figura 18. Pantalla de explorador de archivos.
- Figura 19. Pantalla de cambio de contraseña.
- Figura 20. Pantalla menú lateral de la aplicación.
- Figura 21. Diagrama de casos de uso.
- Figura 22. UC01 – Caso de usuario “Registrar”.
- Figura 23. UC02 – Caso de usuario “Login”.
- Figura 24. UC03 – Caso de usuario “Solicitar contraseña nueva”.
- Figura 25. UC04 – Caso de usuario “Grabar ruta”.
- Figura 26. UC05 – Caso de usuario “Subir ruta”.
- Figura 27. UC06 – Caso de usuario “Descargar ruta”.
- Figura 28. UC07 – Caso de usuario “Exportar ruta”.
- Figura 29. UC08 – Caso de usuario “Cambiar contraseña”.

- Figura 30. Patrón MVP.
- Figura 31. Diseño de la base de datos.
- Figura 32. Diagrama de clases y entidades.
- Figura 33. Esquema de la arquitectura de la aplicación.
- Figura 34. Versiones de Android.
- Figura 35. Consola de la plataforma Google Cloud.
- Figura 36. Pantalla inicial de Google Cloud.
- Figura 37. Pantalla para introducir el nombre del proyecto.
- Figura 38. Crear instancia Cloud SQL.
- Figura 39. Formulario de creación de una instancia nueva.
- Figura 40. Pantalla de autorización de Google Cloud SQL.
- Figura 41. Pantalla de añadir nueva red.
- Figura 42. Pantalla principal de la instancia Cloud SQL.
- Figura 43. Ventana de configurar nueva conexión MySQL.
- Figura 44. Mapa de OpenStreetMaps.
- Figura 45. Mapa de Google Maps.
- Figura 46. Módulos del proyecto Android Studio.
- Figura 47. Módulo “app”.
- Figura 48. Estructura del módulo “backendBttRoute”.
- Figura 49. Entidad correspondiente a la tabla “DificultadFisica”.
- Figura 50. Ejemplo de clase *Endpoint*.
- Figura 51. Ventana Deploy to App Engine de Android Studio.
- Figura 52. Diseño definitivo de la base de datos.
- Figura 53. Terminal de Android Studio.
- Figura 54. Manual de usuario - Acceso al formulario de registro.
- Figura 55. Manual de usuario - Pantalla de registro de usuario.
- Figura 56. Manual de usuario - Tipo de accesos a la aplicación.
- Figura 57. Manual de usuario - Acceso para solicitar una contraseña nueva.
- Figura 58. Manual de usuario - Formulario para solicitar una contraseña nueva.

- Figura 59. Manual de usuario - Acceso a la pantalla de cambio de contraseña.
- Figura 60. Manual de usuario - Pantalla de cambio de contraseña.
- Figura 61. Manual de usuario - Acceso a grabar ruta I.
- Figura 62. Manual de usuario - Acceso a grabar ruta II.
- Figura 63. Manual de usuario - Acceso a grabar ruta III.
- Figura 64. Manual de usuario - Pantalla de grabar ruta.
- Figura 65. Manual de usuario - Opción exportar ruta.
- Figura 66. Manual de usuario - Opción compartir ruta.
- Figura 67. Manual de usuario - Acceso a mapa ampliado.
- Figura 68. Manual de usuario - Mapa ampliado de ruta.
- Figura 69. Manual de usuario - Opción de acceso a filtros.
- Figura 70. Manual de usuario - Pantalla de filtros de búsqueda.

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy día, y gracias al avance de las tecnologías móviles, podemos encontrar un gran número de aplicaciones que nos permiten registrar nuestra actividad física, ver el recorrido que hemos hecho, conocer datos sobre la ruta creada, entre otros datos.

Estas aplicaciones las podemos utilizar, por ejemplo para actividades como senderismo, *running* o *btt*.

Algunas de las aplicaciones más utilizadas son las siguiente:

Aplicación	Sistema	Web
Runkeeper	iOS, Android	www.runkeeper.com
Strava	iOS, Android	www.strava.com
Wikiloc	iOS, Android	www.wikiloc.com
Runtastic	iOS, Android	www.runtastic.com

Figura 1. Aplicaciones disponibles para registro y seguimiento de la actividad física.

El problema principal, es que éstas carecen de funcionalidades que sí disponen otras y viceversa, o bien todas carecen de una determinada funcionalidad o característica concreta. Si analizamos alguna aplicación, como por ejemplo Runkeeper, no dispone de la opción de obtener rutas realizadas por otros usuarios, mientras que sí disponemos de esa funcionalidad con la app de Wikiloc.

Por ejemplo, una carencia que se puede encontrar en cualquiera de las aplicaciones en las que podemos filtrar rutas, es que en ninguna ruta viene identificada la dificultad técnica y/o física a menos que el usuario que ha compartido la ruta lo especifique en algún comentario. Ese problema, en mi opinión es un punto importante a tratar ya que por ejemplo, podemos encontrar a personas que tengan poca técnica pero que tengan una gran resistencia física.

Para resolver todos estos problemas, el usuario tendría que tener instaladas diversas aplicaciones en su dispositivo, y utilizar uno u otra según la necesidad del momento.

El principal objetivo es intentar minimizar dichos problemas y que el usuario disponga de una única aplicación a la hora de realizar la actividad física.

1.2 Objetivos del Trabajo

El objetivo principal del proyecto es crear una aplicación específicamente para actividades *btt* que resuelva los problemas comentados en el anterior punto.

Para conseguir el objetivo, se pondrán en práctica todos los conocimientos adquiridos durante el Máster para así lograr desarrollar un producto que cumpla con las necesidades de los usuarios.

Será necesario seguir los siguientes puntos para poder tener un producto que cumpla con las necesidades de los usuarios, a destacar la relevancia de éstos dos:

- Conocer los productos que existen actualmente en el mercado y analizar sus carencias.
- Estudiar a los posibles usuarios potenciales del producto.

A nivel de requerimientos funcionales, la aplicación deberá cumplir con los siguiente puntos:

- Registro en la plataforma mediante un sencillo formulario.
- *Login* del usuario ya sea con una cuenta registrada mediante el formulario o con una cuenta de Facebook.
- Crear *track* de la actividad física.
- Consultar rutas realizadas por otros usuarios.
- Importar archivo GPX procedente de otra fuente.
- Exportar *track* en formato GPX para poderlo cargar en otro dispositivo, ya sea Smartphone o GPS.
- Subir el *track* realizado al servidor para que otros usuarios lo puedan consultar.
- Filtrar rutas según los parámetros que indique el usuario.
- Mostrar rutas cercanas a la seleccionada por el usuario en el buscador.

1.3 Enfoque y método seguido

Según las necesidades y los problemas comentados en el punto 1.1 y 1.2, se ha optado por crear una aplicación nueva que reúna algunas de las mejores características que se pueden encontrar en otras aplicaciones.

Se ha optado por implementarla para Smartphones con sistema operativo Android. La respuesta a ésta decisión se basa principalmente en que un Smartphone es un dispositivo fácil de transportar a la hora de hacer ejercicio; una Tablet sobre el manillar sería muy incómodo. También nos permite la posibilidad de exportar la ruta en un archivo GPX, el cual podremos cargar por ejemplo, en un GPS Garmin o en un Smartwach.

Otra ventaja destacable, es que casi todo el mundo dispone de Smartphone y resulta bastante más económico que un GPS de gama media.

Por otra parte, el motivo de hacer la aplicación para sistemas Android se basa en el gran dominio en el mercado de este sistema frente a otros como iOS o Windows Phone.

Operating System	1Q17 Units	1Q17 Market Share (%)	1Q16 Units	1Q16 Market Share (%)
Android	327,163.6	86.1	292,746.9	84.1
iOS	51,992.5	13.7	51,629.5	14.8
Other OS	821.2	0.2	3,847.8	1.1
Total	379,977.3	100.0	348,224.2	100.0

Source: Gartner (May 2017)

Figura 2. Unidades vendidas de dispositivos. Fuente: www.gartner.com/newsroom/id/3725117

Como se puede observar en los datos obtenidos de Gartner^[1], en el primer cuatrimestre de 2017, Android alcanza el 86,1% frente al 13,7% de iOS.

Para este proyecto se ha elegido el modelo de desarrollo basado en agile scrum. Este tipo de modelo, nos permite dividir el proyecto en diferentes sprints: toma de requisitos, diseño, implementación, pruebas, etc... Además el usuario podrá ver en cada sprint el resultado obtenido y valorar si se están cumpliendo los requisitos y si se debe cambiar o mejorar alguna funcionalidad.

Este tipo de modelo nos permitirá obtener una aplicación en un corto plazo de tiempo, dada la rapidez en que aparecen nuevas aplicaciones móviles.

Por último, comentar que realizando un desarrollo de forma nativa para una plataforma, siempre podremos aprovechar mejor todos los recursos que nos ofrece el dispositivo. Además, la experiencia de usuario de una aplicación nativa siempre será mejor y tendrá un rendimiento mejor que en una aplicación híbrida.

1.4 Planificación del Trabajo

Los recursos de tipo hardware utilizados para este proyecto son:

- iMac Intel Core i3 con sistema operativo macOS Sierra y 12 GB de memoria RAM.
- Portatil Acer Intel Core i3 con sistema operativo Windows 10 y 12GB de memoria RAM.

- Smartphone Samsung S5 neo (SM-G903F) con sistema operativo Android Marshmallow (6.0.1).

En cuanto al software utilizado, necesitamos:

- Android Studio 2.3.3
<https://developer.android.com/studio/index.html>
- MySQL Workbench para realizar el diagrama de la base de datos.
<https://www.mysql.com/products/workbench/>
- Justinmind para realizar el prototipado.
<https://www.justinmind.com/>
- Google Cloud para gestionar el *backend* y la base de datos MySQL.
<https://cloud.google.com/>
- Microsoft Office para generar la documentación: memoria, manual de usuario y presentación.
<https://www.microsoft.com/>
- Paint2 para edición de imágenes.
<https://itunes.apple.com/es/app/paint-2/id736473980?mt=12>
- Draw.io para crear el diagrama de flujo.
<https://www.draw.io>
- ArgoUML para crear el diagrama de flujo.
<http://argouml.tigris.org/>

Las tareas a realizar en el trabajo, se diferencian en diferentes entregas:

Tarea	Descripción	Fecha de inicio	Fecha de entrega
PEC1	<ul style="list-style-type: none"> • Contexto y justificación. • Objetivos. • Enfoque y método elegido. • Planificación 	20/09/2017	11/10/2017
PEC2	<ul style="list-style-type: none"> • Usuarios y contexto de uso. • Diseño conceptual. • Prototipo. • Evaluación. 	12/10/2017	01/11/2017

PEC3	<ul style="list-style-type: none"> • Desarrollo app. • Desarrollo Base de datos y pruebas de conexión. • Desarrollo de <i>backend</i> y pruebas de conexión. • Pruebas y solución de errores. • Documentación. 	02/11/2017	13/12/2017
PEC4	<ul style="list-style-type: none"> • Manual de usuario. • Vídeo. • Presentación. • Finalizar memoria. 	14/12/2017	03/01/2018

A continuación mostramos el diagrama de Gantt donde se detallan el número de horas empleadas en cada tarea para este proyecto.

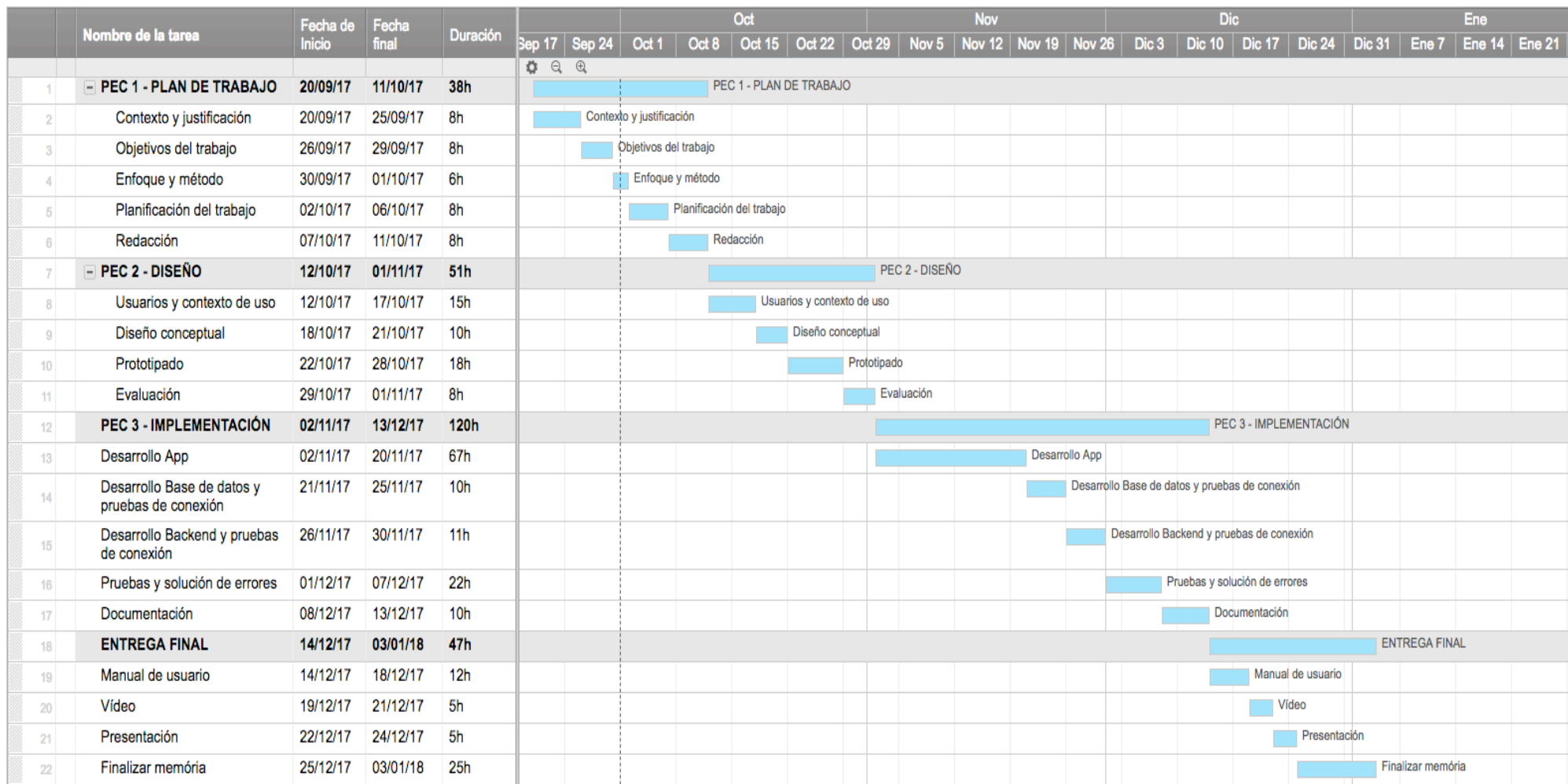


Figura 3. Diagrama de Gantt

Se ha estimado un total de 256 horas (de un total de 300 horas), para completar el proyecto, dejando 44 horas libres para distribuir entre las diferentes tareas en el caso de que sea necesario.

Para los días laborales, se van a dedicar entre 2,5 horas y 3 horas, mientras que para los fines de semana se ve incrementado el número de horas de entre 3 y 6 horas dependiendo de la tarea que se realice. Probablemente, por motivos de trabajo, algún día laboral será muy difícil dedicar las horas estipuladas. Aunque se intentará recuperar en fines de semana.

1.5 Breve resumen de productos obtenidos

Al finalizar el proyecto se entregarán los siguientes productos:

- Aplicación BttRoute junto con el código fuente.
- Manual de usuario de la aplicación.
- Vídeo de presentación donde se explicará el desarrollo y funcionamiento de la aplicación.
- Presentación realizada en Power-Point.
- Memoria de trabajo de fin de Máster.

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos se detallará todo el proceso de creación de una aplicación.

En el capítulo de diseño se presentarán los diversos usuarios potenciales de la aplicación a través de unas fichas donde se explicarán sus necesidades y objetivos. El siguiente paso será detallar diferentes casos de uso a partir de las fichas.

Para finalizar el capítulo de diseño, se presentará un prototipo de la aplicación de alto nivel junto con los diagramas que mostrarán el flujo de trabajo.

En el capítulo de implementación se detallará la tecnología utilizada para el desarrollo como el framework, librerías y APIs que son necesarias, donde se mostrarán algunas de las partes de código más importantes. Después se detallarán las pruebas realizadas y los resultados obtenidos.

Para finalizar la memoria del trabajo de fin de Máster, encontraremos un capítulo dedicado a las conclusiones, otro capítulo dedicado al glosario de términos y acrónimos, otro para la bibliografía y para finalizar, el anexo.

2. Diseño Centrado en el Usuario (DCU)

En este capítulo trataremos los puntos iniciales para cualquier desarrollo de una aplicación. Éstos se dividen en una investigación y toma de requisitos de usuarios potenciales, una elaboración de casos de uso y flujos de interacción, y por último un diseño de prototipo de cómo será la aplicación.

Es importante analizar muy bien a los posibles usuarios, ya que a partir de este análisis podremos detectar las funcionalidades que deberá tener la aplicación y evitar otras que no sean de gran importancia.

2.1 Usuarios y contexto de uso

2.1.1 Usuarios

Como principales usuarios de la aplicación BttRoute podemos encontrar desde personas que suelen salir esporádicamente en bicicleta, como personas que practican este deporte de una forma más habitual y exigente.

A través de las fichas que se presentan a continuación, conoceremos más detalladamente las características, perfiles, y objetivos de los usuarios.

Ficha 1

Nombre

Alberto

Edad

36 años

Profesión

Administrativo

Nivel de conocimientos tecnológicos

- *Redes sociales:* **Alto**
- *Smartphone y Tablet:* **Alto**
- *IT y Internet:* **Alto**

Descripción

Trabaja en unas oficinas situadas cerca de su casa. Su horario es de lunes a viernes con una jornada laboral de 8 horas. Durante la semana suele ir al gimnasio para hacer algo de ejercicio y estar en forma. El fin de semana, suele practicar deporte pero al aire libre y normalmente lo realiza por el parque natural de Collserola. Practica *running* y también *btt*. A la hora de practicar *btt* no realiza rutas muy exigentes tanto a nivel físico ni técnico, sino que escoge rutas bastantes sencillas que le sirven para estar en forma.

Necesidades y objetivos

Para poder practicar *btt* necesita una plataforma que le permita registrar sus rutas. Además, le interesa poder buscar rutas realizadas por otros usuarios. Para buscar rutas, le interesa que se pueda filtrar por dificultad física y técnica, y que ofrezca la posibilidad de descargarla. La opción de descargar es importante ya que le permitiría hacer la ruta en cualquier momento.

Dispositivos y plataformas

- *Smartphone:* **Android**
- *Tablet:* **Android**
- *Portátil:* **Apple**
- *PC:* **Windows**

Ficha 2

Nombre

Marta

Edad

32 años

Profesión

Monitora de gimnasio

Nivel de conocimientos tecnológicos

- *Redes sociales:* **Alto**
- *Smartphone y Tablet:* **Medio**
- *IT y Internet:* **Medio**

Descripción

Marta trabaja en un gimnasio como profesora de *spinning* por las tardes. Las mañanas las dedica a estudiar y a practicar su deporte favorito, *btt*. Sus rutas suelen ser de un nivel físico bastante alto aunque a nivel técnico no son muy difíciles. Para escoger las rutas, se basa casi siempre en rutas hechas por otros usuarios y las carga en su Smartphone.

Necesidades y objetivos

Para realizar sus rutas necesita una aplicación que le permita buscar rutas con diferentes niveles de dificultad física y técnica. Es importante que la plataforma permita descargarlas y poderlas cargar más tarde. Otra característica importante es que le permita obtener rutas pertenecientes a otras fuentes (webs, creadas por otros dispositivos, etc...) y cargarlas en la plataforma.

Dispositivos y plataformas

- *Smartphone:* **Android**
- *Tablet:* -
- *Portátil:* **Windows**
- *PC:* -

FICHA 3

Nombre

Javi

Edad

25 años

Profesión

Camarero

Nivel de conocimientos tecnológicos

- *Redes sociales:* **Alto**
- *Smartphone y Tablet:* **Bajo**
- *IT y Internet:* **Bajo**

Descripción

Javi trabaja de Martes a Sábado como camarero en un restaurante. Al menos uno de los días de los que no trabaja lo dedica a su ejercicio favorito, la bicicleta de montaña. En concreto le gusta la modalidad de descenso. Se trata de una modalidad en la que hay que tener bastante técnica dado los tipos de recorrido. A veces práctica él solo y otras suele quedar con un grupo de amigos.

Actualmente sólo graba las rutas que realiza, no consulta ni descarga ninguna otra.

Necesidades y objetivos

A menudo le gustaría consultar otras rutas realizadas por otros aficionados a esta modalidad, pero no encuentra una plataforma que le permita filtrar y/o compartir las suyas propias. Necesita una plataforma que además de crear sus propias rutas, le permita buscar otras que sean de la misma especialidad y descargarlas.

Dispositivos y plataformas

- *Smartphone:* **Android**
- *Tablet:* -
- *Portátil:* **Windows**
- *PC:* -

2.1.2 Contexto de uso

La aplicación se podrá utilizar en dos principales contextos de uso, y se detallan a continuación:

- El usuario, principalmente utilizará la aplicación en la montaña para crear una nueva ruta o una descargada desde la propia plataforma. Podrá controlar en todo momento el proceso de grabación, ya sea pausando o parando la actividad y manipular el mapa de la forma que crea conveniente. Cuando se termine la actividad, la aplicación permitirá guardar la ruta o eliminarla. En este contexto de uso, siempre será necesario que el usuario tenga activada la geolocalización en el dispositivo y tener una conexión de datos para conectarse a internet.
- Otra forma de utilizar la aplicación, puede ser desde casa o cualquier otro entorno, ya sea un centro comercial, un bar u otro lugar. Podremos buscar rutas en la plataforma y consultar otras creadas por nosotros mismos. A través de unos filtros, obtendremos rutas que se adecuen a nuestras preferencias. Igual que en el punto anterior, necesitamos tener una conexión a internet ya sea con una conexión wifi o con una conexión de datos.

3. Diseño conceptual

En este capítulo se detallan los diferentes escenarios de uso en los que el usuario podrá utilizar las funcionalidades de la aplicación y como lo hará.

Escenario de uso – El usuario quiere crear una nueva ruta

Descripción:

El usuario se encuentra en el punto de partida de la ruta y ya está preparado para comenzar. Primero introduce el *login* con el que se ha registrado anteriormente. Una vez dentro de la aplicación accede a la pantalla que le permitirá grabar una ruta. Para acceder lo podrá hacer desde el menú lateral en la parte izquierda o desde la pantalla de “Mis rutas”.

Para iniciar la grabación únicamente tendrá que pulsar sobre el botón de “Grabar” y comenzar a pedalear. En el momento que lo crea conveniente podrá pausar y reanudar la actividad cuando así desee.

Una vez finalizada la ruta deberá parar la actividad en el dispositivo y accederá a la pantalla donde se muestran los resultados junto con el trazado realizado. En este punto el usuario decidirá si quiere grabar la ruta o prefiere eliminarla.

Escenario de uso – Descargar una ruta realizada por otro usuario

Descripción:

En este escenario de uso, el usuario se puede encontrar en cualquier sitio en el que disponga de una conexión, puede ser en su casa, en la oficina, en un centro comercial, etc...

El objetivo final es encontrar una ruta que le agrade al usuario y descargarla. Para ello primero de todo se tendrá que validar en la aplicación. Una vez dentro, se dirigirá a la pantalla de “Buscar rutas” desde el menú desplegable que tiene a la izquierda de la pantalla.

Inicialmente se le mostrarán ya rutas compartidas por otros usuarios. En este punto el usuario podrá acceder a la pantalla de filtros a través del icono que se encuentra en la parte derecha de la barra superior. En ella podrá filtrar por el tipo de ruta que desea.

Cuando ya ha obtenido todas las rutas que se ajusten a su filtrado, seleccionará la que más le guste y accederá al detalle de la misma. Dentro del detalle y a través de un icono podrá descargar la ruta en su dispositivo en un archivo con formato GPX.

Escenario de uso - Carga de archivos GPX

Descripción:

Cuando el usuario quiera cargar una ruta descargada o procedente de otra fuente (copiando el archivo GPX en la ruta de la app), deberá acceder a la

aplicación a través de su *login*. Una vez dentro, irá a la pantalla de “Cargar archivo GPX” accesible desde el menú desplegable. Esta pantalla le mostrará los diferentes archivos que tiene en el directorio y el usuario sólo tendrá que seleccionar el que quiera cargar. Una vez seleccionado, la aplicación dirigirá al usuario directamente a la pantalla de grabar ruta mostrando en el mapa el recorrido a realizar.

4. Prototipado

El prototipo de una aplicación nos permite mostrar con el máximo de detalle qué apariencia tendrá la aplicación.

Esto permite al cliente/usuario evaluar el prototipo y proponer cambios o aceptar el diseño para comenzar a desarrollar la aplicación.

A continuación se muestra el prototipo de la aplicación BttRoute:



Figura 4. Pantalla de "Login".

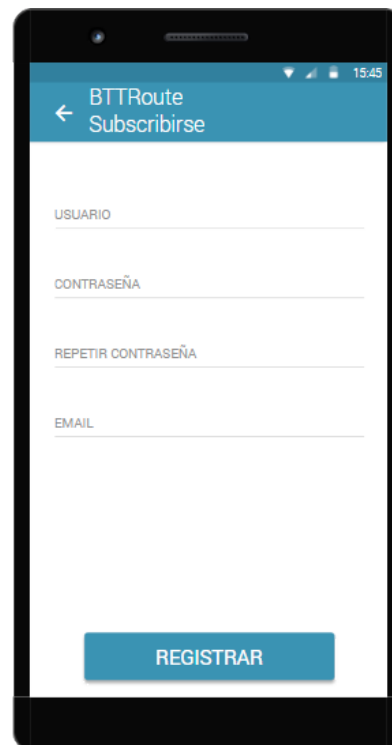


Figura 5. Pantalla de registro de usuario.

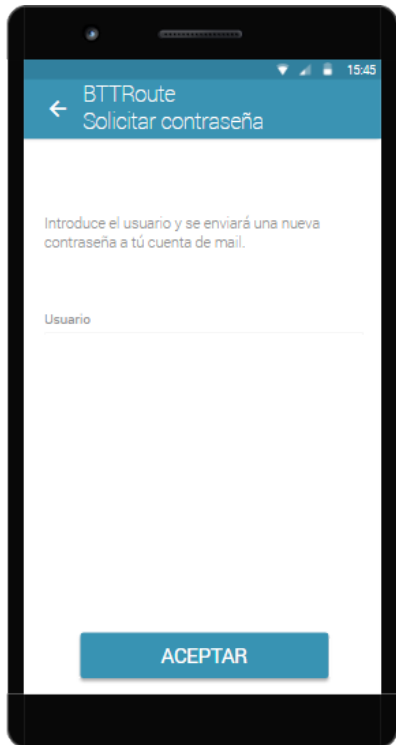


Figura 6. Pantalla para solicitar una contraseña nueva.



Figura 7. Pantalla "Mis rutas" sin datos.

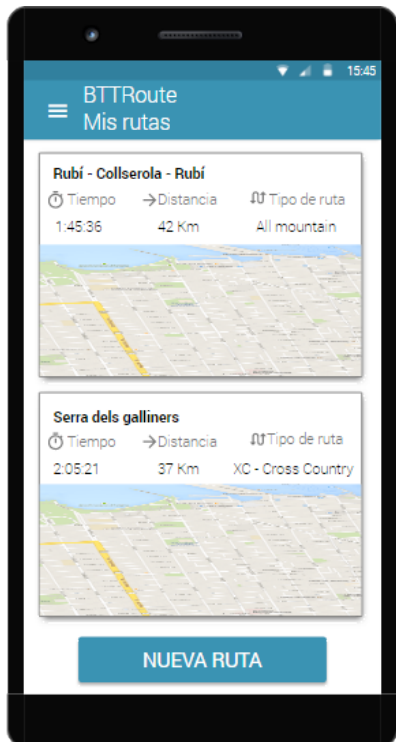


Figura 8. Pantalla "Mis rutas" con datos.

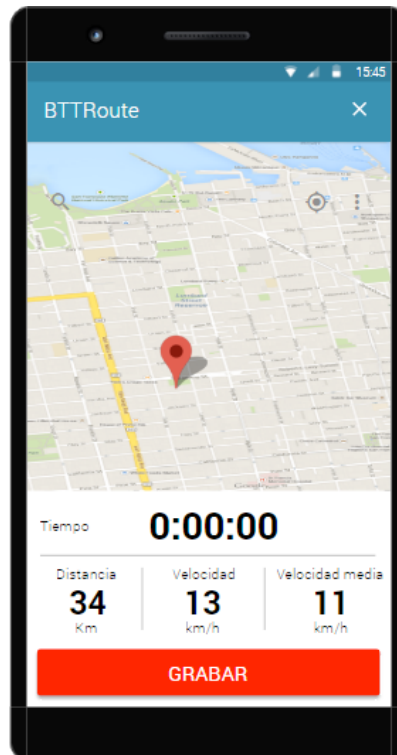


Figura 9. Pantalla para grabar ruta.

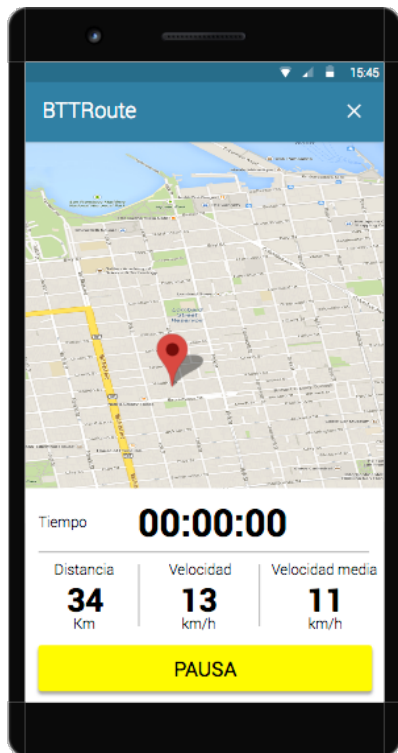


Figura 10. Pantalla de para grabar ruta en funcionamiento.

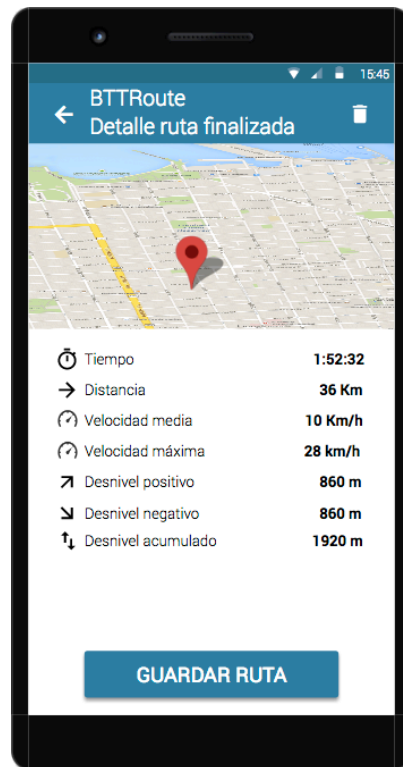


Figura 11. Pantalla de detalle de ruta finalizada.

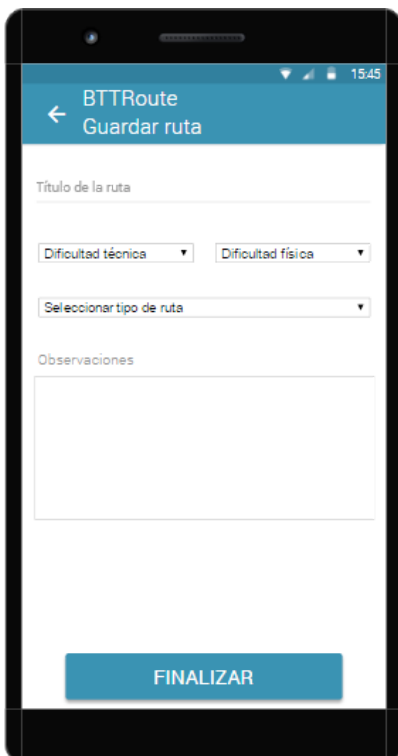


Figura 12. Pantalla para guardar ruta.



Figura 13. Pantalla detalle ruta creada por el usuario.

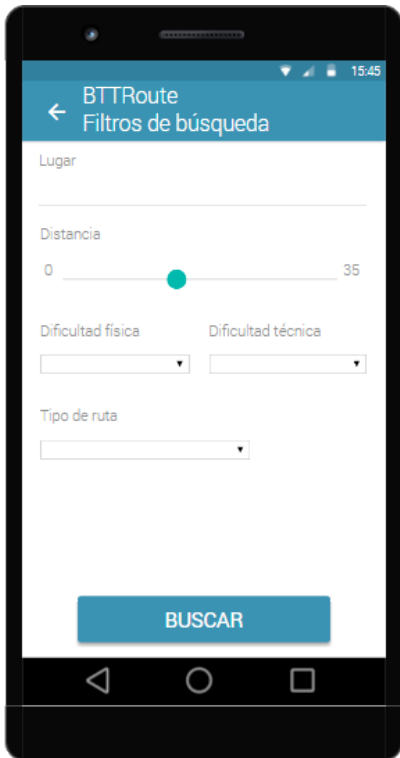


Figura 14. Pantalla filtros de búsqueda de rutas subidas.



Figura 15. Pantalla de listado de rutas subidas.

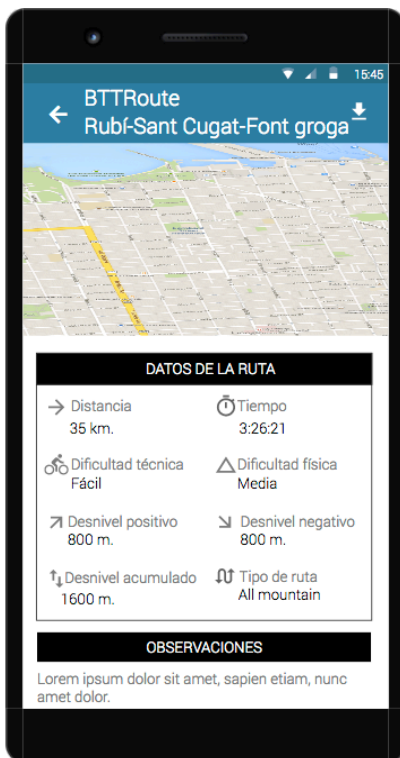


Figura 16. Pantalla detalle de ruta subida por otro usuario.

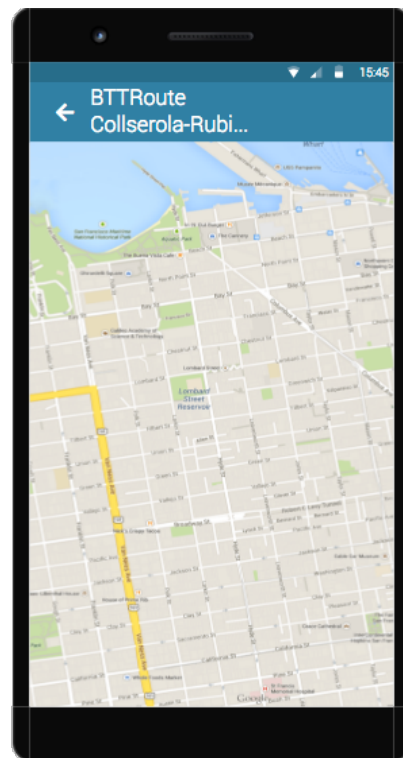


Figura 17. Pantalla visor de ruta ampliada.

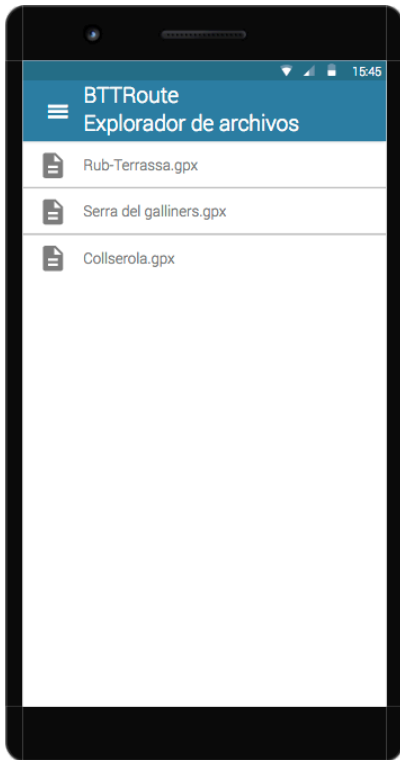


Figura 18. Pantalla de explorador de archivos.



Figura 19. Pantalla de cambio de contraseña.

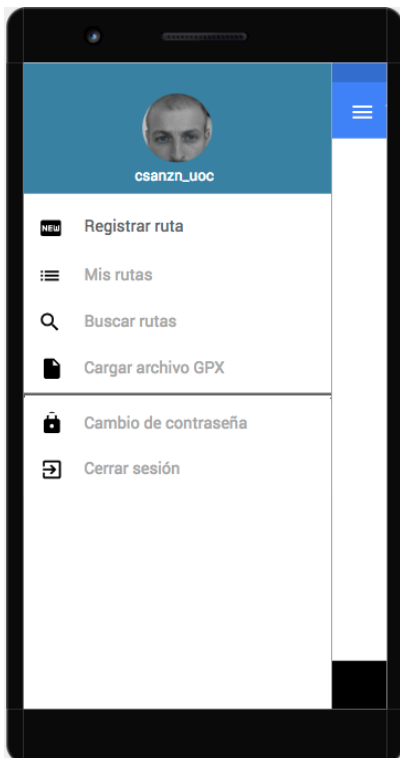


Figura 20. Pantalla menú lateral de la aplicación.

5. Evaluación

En este capítulo se detallarán los casos de uso así como el diseño de la arquitectura seleccionada para desarrollar la aplicación

5.1 Definición de los casos de uso

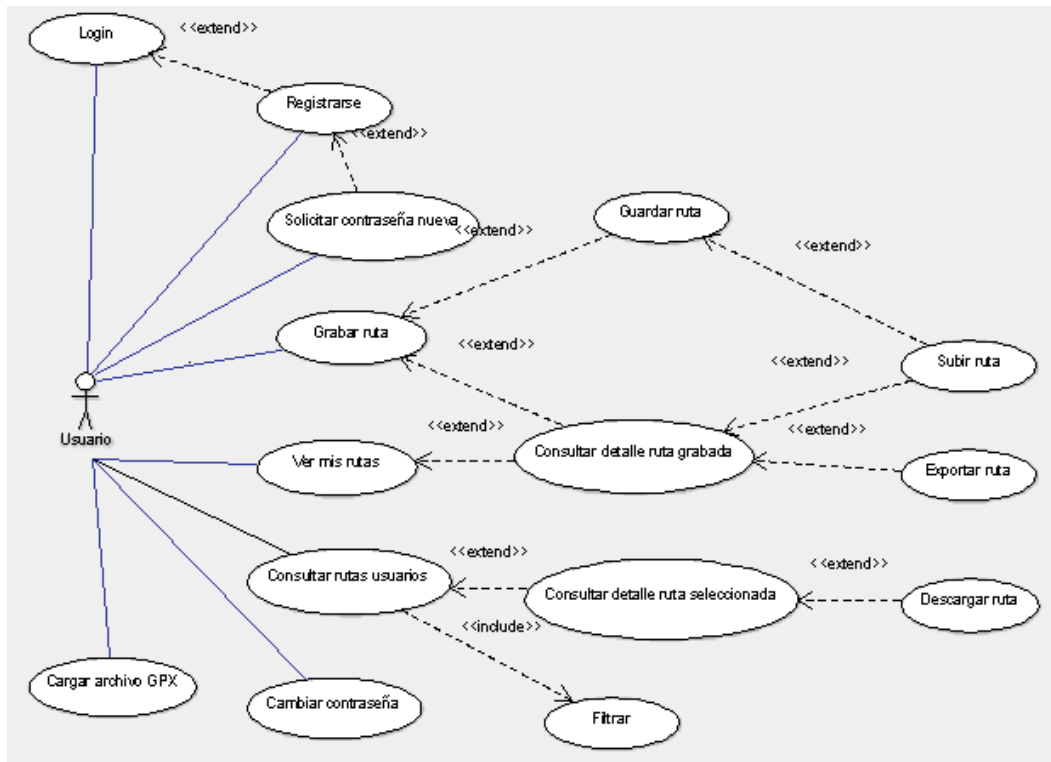


Figura 21. Diagrama de casos de uso.

ID	UC01 - Registrar
ACTORES	Usuario
PRECONDICIONES	Ninguna
FLUJO	<p><i>Flujo principal:</i></p> <ol style="list-style-type: none"> 1. El usuario accede al formulario de registro desde la pantalla de <i>login</i>. 2. El usuario completa todos los campos. 3. El sistema comprueba que no exista el nombre de usuario introducido. 4. El sistema registra al usuario. <p><i>Flujo alternativo 1:</i></p> <ol style="list-style-type: none"> 1. El usuario introducido ya existe en el sistema. 2. Se muestra un mensaje de error. <p><i>Flujo alternativo 2:</i></p> <ol style="list-style-type: none"> 1. Las contraseñas introducidas por el usuario no

	coinciden. Se muestra un mensaje de error.
POSTCONDICION	El usuario está registrado en el sistema.

Figura 22. UC01 – Caso de usuario “Registrar”

ID	UC02 - <i>Login</i>
ACTORES	Usuario
PRECONDICIONES	UC01. Es necesario estar registrado en el sistema
FLUJO	<p><i>Flujo principal:</i></p> <ol style="list-style-type: none"> 1. El usuario introduce el usuario y contraseña. 2. El sistema comprueba si todos los datos están informados. 3. El sistema comprueba si el usuario y contraseña está registrado y si son correctos. 4. El usuario accede dentro de la aplicación. <p><i>Flujo alternativo 1:</i></p> <ol style="list-style-type: none"> 1. El usuario no introduce usuario o contraseña. 2. Se muestra un mensaje de error. <p><i>Flujo alternativo 2:</i></p> <ol style="list-style-type: none"> 1. El usuario introduce el usuario y contraseña. 2. El sistema comprueba si todos los datos están informados. 3. El sistema comprueba si el usuario y contraseña está registrado y si son correctos. 4. Los datos no son correctos y se muestra un mensaje de error.
POSTCONDICION	El usuario accede a dentro de la aplicación.

Figura 23. UC02 – Caso de usuario “Login”

ID	UC03 - Solicitar contraseña nueva
ACTORES	Usuario
PRECONDICIONES	UC01. Es necesario estar registrado en el sistema
FLUJO	<p><i>Flujo principal:</i></p> <ol style="list-style-type: none"> 1. El usuario introduce su nombre de usuario. 2. El sistema comprueba que exista el nombre de usuario. 3. El sistema genera una nueva clave. 4. Se envía al mail del usuario la nueva clave. <p><i>Flujo alternativo 1:</i></p> <ol style="list-style-type: none"> 1. El usuario no introduce ningún usuario. 2. El sistema muestra un mensaje de error. <p><i>Flujo alternativo 2:</i></p>

	<ol style="list-style-type: none"> 1. El usuario introduce un usuario 2. El sistema comprueba que exista el nombre de usuario. 3. El usuario no existe y se muestra un mensaje de error.
POSTCONDICION	El usuario dispone de una nueva clave.

Figura 24. UC03 – Caso de usuario “Solicitar contraseña nueva”

ID	UC04 - Grabar ruta
ACTORES	Usuario
PRECONDICIONES	UC02. Es necesario haber realizado un <i>login</i> .
FLUJO	<p><i>Flujo principal:</i></p> <ol style="list-style-type: none"> 1. El usuario accede a la pantalla de grabar nueva ruta desde el menú lateral o desde la pantalla de “Mis rutas”. 2. El sistema comprueba que el usuario tiene activada la geolocalización y la obtiene. 3. El usuario inicia la grabación. 4. El usuario finaliza la ruta y la guarda en la base de datos del servido en modo privado. <p><i>Flujo alternativo 1:</i></p> <ol style="list-style-type: none"> 1. El sistema detecta que no tiene activada la geolocalización o que no la puede obtener. 2. Muestra un error en pantalla.
POSTCONDICION	Se crea una nueva ruta y se añade a la pantalla de “Mis rutas”

Figura 25. UC04 – Caso de usuario “Grabar ruta”

ID	UC05 – Subir ruta.
ACTORES	Usuario
PRECONDICIONES	UC04. Es necesario haber grabado una ruta y guardarla
FLUJO	<p><i>Flujo principal:</i></p> <ol style="list-style-type: none"> 1. El usuario accede a “Mis rutas”. 2. Selecciona una y accede al detalle. 3. El usuario selecciona la opción de subir ruta y la sube al servidor en modo compartido. <p><i>Flujo alternativo 1:</i></p> <ol style="list-style-type: none"> 1. El usuario intenta subir una ruta procedente del servidor. 2. El sistema le reporta un mensaje advirtiéndole de que esa ruta ya existe en el servidor.
POSTCONDICION	El usuario pone la ruta guardada en modo compartido para que otros usuarios la puedan descargar.

Figura 26. UC05 – Caso de usuario “Subir ruta”

ID	UC06 - Descargar ruta.
ACTORES	Usuario
PRECONDICIONES	UC02. Es necesario haber realizado el <i>login</i> .
FLUJO	<p><i>Flujo principal:</i></p> <ol style="list-style-type: none"> 1. El usuario accede a la pantalla de “buscar rutas”. 2. Selecciona la ruta deseada y accede al detalle. 3. Selecciona la opción de descargar y ésta se descargará en un fichero GPX que se guardará en la ruta de la app.
POSTCONDICION	El usuario dispone de un nuevo archivo GPX en el directorio de la app.

Figura 27. UC06 – Caso de usuario “Descargar ruta”

ID	UC07 - Exportar ruta.
ACTORES	Usuario
PRECONDICIONES	UC02. Es necesario haber realizado el <i>login</i> . UC04. El usuario debe haber grabado una ruta.
FLUJO	<p><i>Flujo principal:</i></p> <ol style="list-style-type: none"> 1. El Usuario accede a “Mis rutas”. 2. Selecciona la ruta deseada y accede al detalle. 3. Selecciona la opción exportar y se generará un fichero GPX que se guardará en la ruta de la app.
POSTCONDICION	El usuario dispone de la ruta en un archivo GPX.

Figura 28. UC07 – Caso de usuario “Exportar ruta”

ID	UC08 - Cambiar contraseña.
ACTORES	Usuario
PRECONDICIONES	UC01. Es necesario estar registrado. UC02. Es necesario haber realizado el <i>login</i> .
FLUJO	<p><i>Flujo principal:</i></p> <ol style="list-style-type: none"> 1. El usuario accede al apartado “cambiar contraseña”. 2. Completa los datos solicitados. 3. El sistema actualiza la contraseña actual. <p><i>Flujo alternativo 1:</i></p> <ol style="list-style-type: none"> 1. El usuario no completa todo los campos. 2. El sistema muestra un mensaje de error. <p><i>Flujo alternativo 2:</i></p> <ol style="list-style-type: none"> 1. El usuario introduce en los campos “contraseña nueva” y “repite contraseña nueva” con valores diferentes. 2. El sistema muestra un mensaje de error. <p><i>Flujo alternativo 3:</i></p>

POSTCONDICION	<ol style="list-style-type: none"> 1. El usuario introduce la contraseña actual y la nueva. 2. El sistema comprueba que la contraseña actual introducida coincida con la que está registrada en el sistema. 3. Si no coincide, el sistema muestra un mensaje de error.
	El usuario dispone de la nueva contraseña elegida.

Figura 29. UC08 – Caso de usuario “Cambiar contraseña”

5.2. Diseño de la arquitectura

La arquitectura seleccionada para el desarrollo de la aplicación estará basada en el patrón modelo-vista-presentador (MVP)^[2]. Esta patrón es una derivación del patrón modelo-vista-controlador (MVC), y, al igual que este último, nos permite separar la capa de presentación de la lógica de la misma.

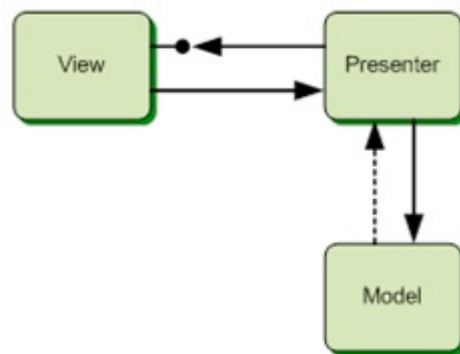


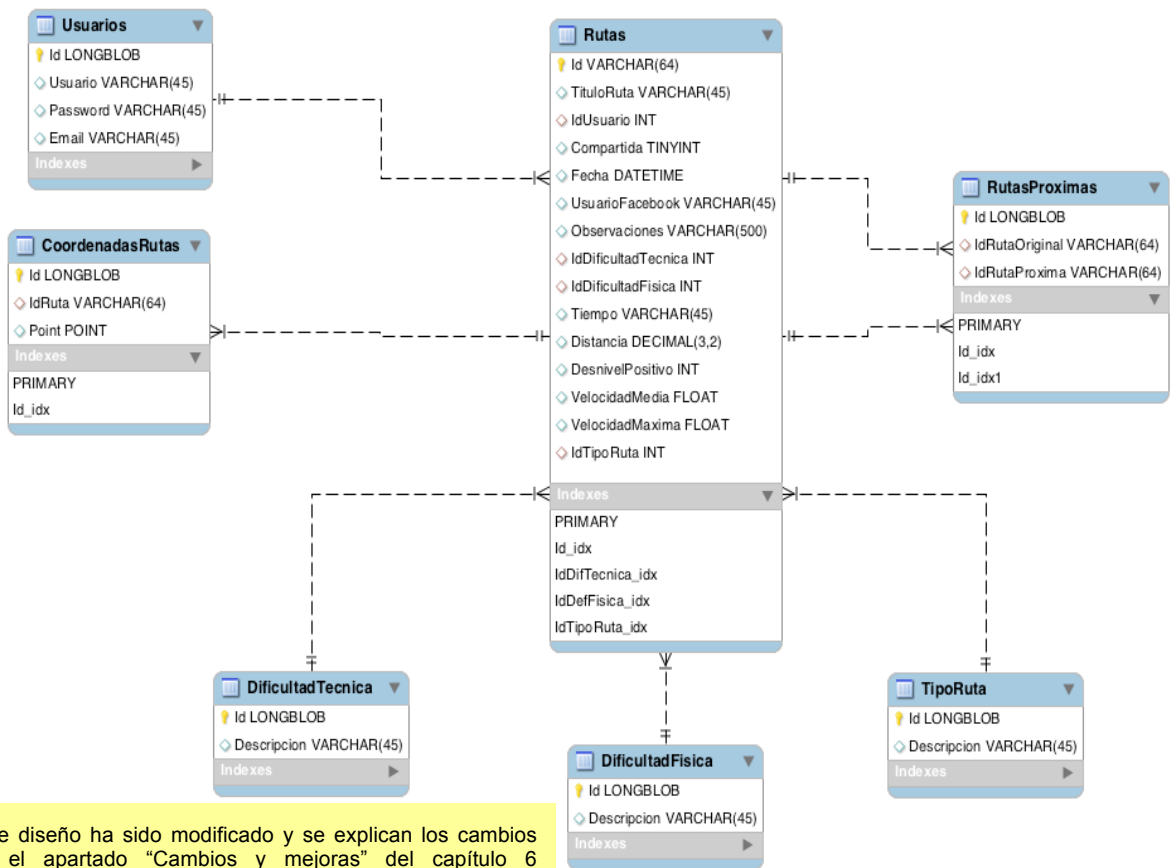
Figura 30. Patrón MVP

- El **modelo** es la capa que gestiona los datos.
- La **vista** es la encargada de mostrar los datos (Fragments, views...)
- El **presentador** ofrecerá a la vista los datos recibidos por el modelo.

Diseño de la base de datos

Para el diseño de la base de datos se ha utilizado Google Cloud SQL, así que toda la información estará ubicada en la nube y podremos acceder a la información a través de la propia API de Google Cloud. Se ha elegido en este caso utilizar MySQL^[3] ya que esta consume menos recursos y ofrece una mayor velocidad en cuanto a consultas se refiere. Google Cloud también nos ofrece la oportunidad de utilizar PostgreSQL.

Una de las ventajas que tenemos de utilizar esta plataforma, es que nos quitará la preocupación de montar nuestro propio servidor SQL y tenerlo que gestionar nosotros mismos.



Este diseño ha sido modificado y se explican los cambios en el apartado "Cambios y mejoras" del capítulo 6 "Implementación"

Figura 31. Diseño de la base de datos.

Diseño de entidades y clases

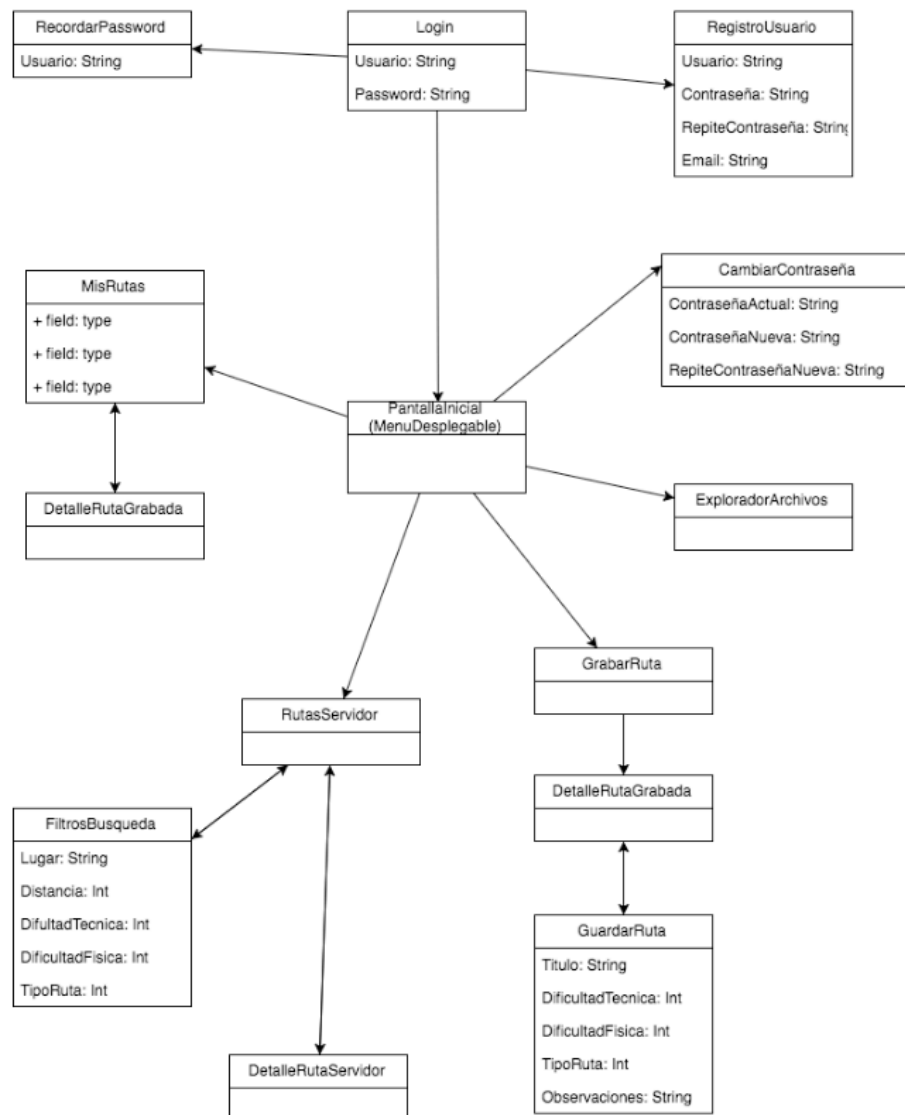


Figura 32. Diagrama de clases y entidades.

Arquitectura del sistema

Para el desarrollo de este proyecto, se ha decidido hacer uso de todas las herramientas que nos ofrece Google Cloud para el despliegue de una aplicación.

En nuestro caso se ha utilizado:

- Cloud SQL^[4] para el almacenamiento de datos.
- Cloud Endpoints^[5] para realizar las llamadas al *backend*.
- App Engine^[6] para alojar el *backend*.

Dentro de nuestro proyecto Android Studio, implementaremos con la API de Google Cloud Endpoints las llamadas a los métodos del *backend*. También estará dentro del proyecto el desarrollo del *backend* que se alojará en el App Engine y que publicaremos por medio de la misma plataforma de desarrollo.

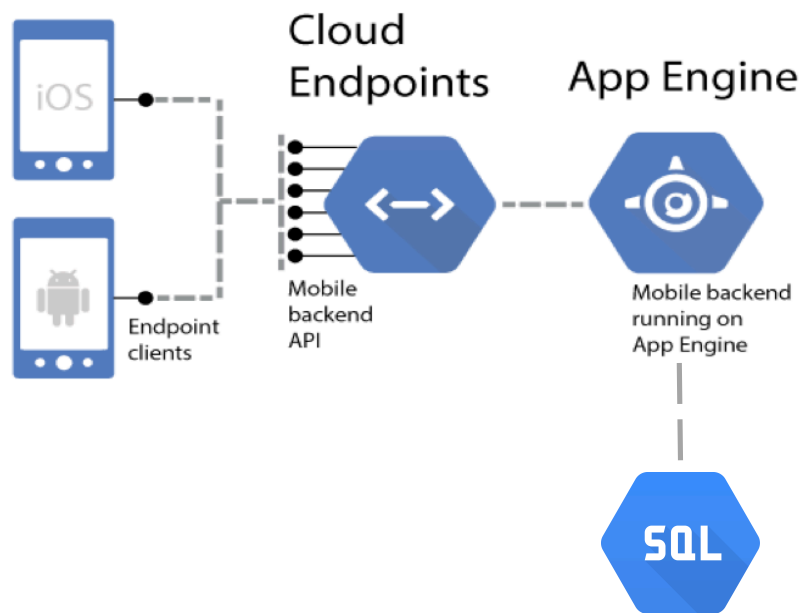


Figura 33. Esquema de la arquitectura de la aplicación.

Tal como se puede observar en el esquema, la aplicación, para acceder a los datos lo hará mediante peticiones realizadas a través de los *endpoints* que se conectarán con el *backend* y éste se encargará de solicitarlos o introducirlos en la base de datos.

Para desarrollar el *backend* podemos elegir entre diferentes lenguajes para su implementación. En este caso, se ha elegido Java debido a que es el mismo lenguaje que se ha utilizado para desarrollar la app.

6. Implementación

6.1 Introducción

Para desarrollar la aplicación se ha optado por un desarrollo nativo únicamente para Smartphones y con soporte para una versión mínima de Android 5.1 (SDK 22). Con esta versión podemos cubrir el 60% de los dispositivos que hay actualmente.

Se ha optado por esta versión debido a que cada vez hay más tendencia de cambiar de dispositivo en periodos de tiempo cada vez más cortos.

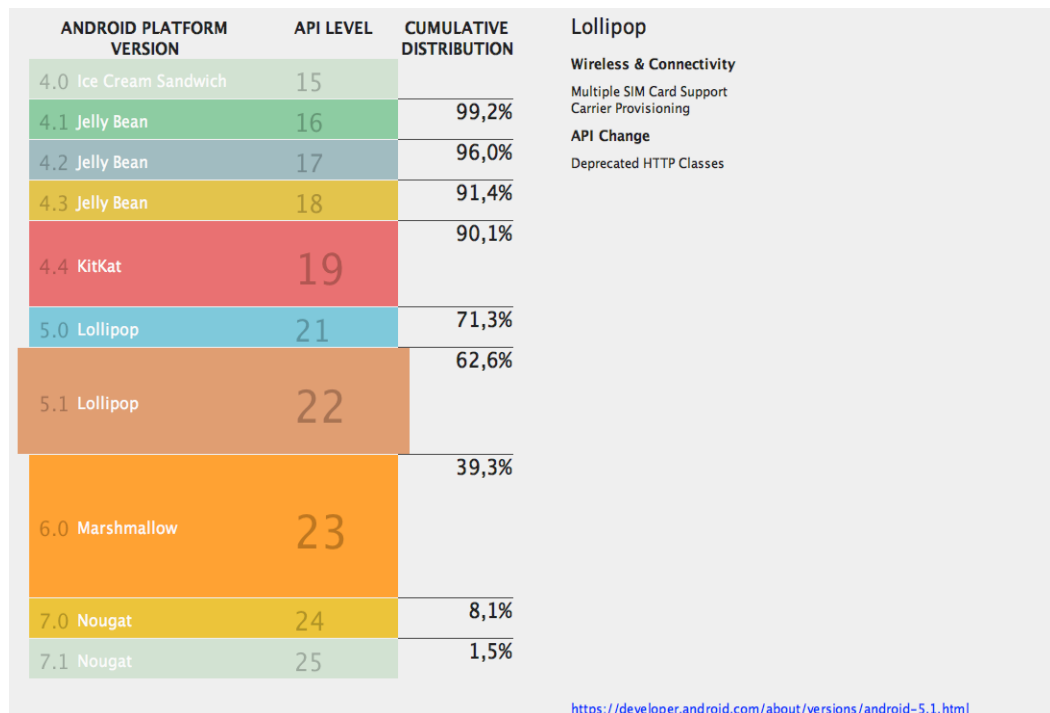


Figura 34. Versiones de Android.

6.2 Herramientas utilizadas

Para el desarrollo de la aplicación se han utilizado las siguientes herramientas.

Android Studio

Como herramienta de desarrollo se ha elegido el IDE Android Studio versión 2.3.3 tanto para el desarrollo de la app como el de *backend* (ambos incluidos en el mismo proyecto). Desde la misma plataforma, se publicará el *backend* a Google Cloud tal como se detalla en los siguientes apartados.

Google Cloud Platform^[7]

Plataforma que nos ofrece todos los servicios necesarios para hacer funcionar una aplicación sin la necesidad de tener nuestro propio servidor físico ya que esta plataforma está dirigida para crear ciertos tipos de soluciones a través de la nube.

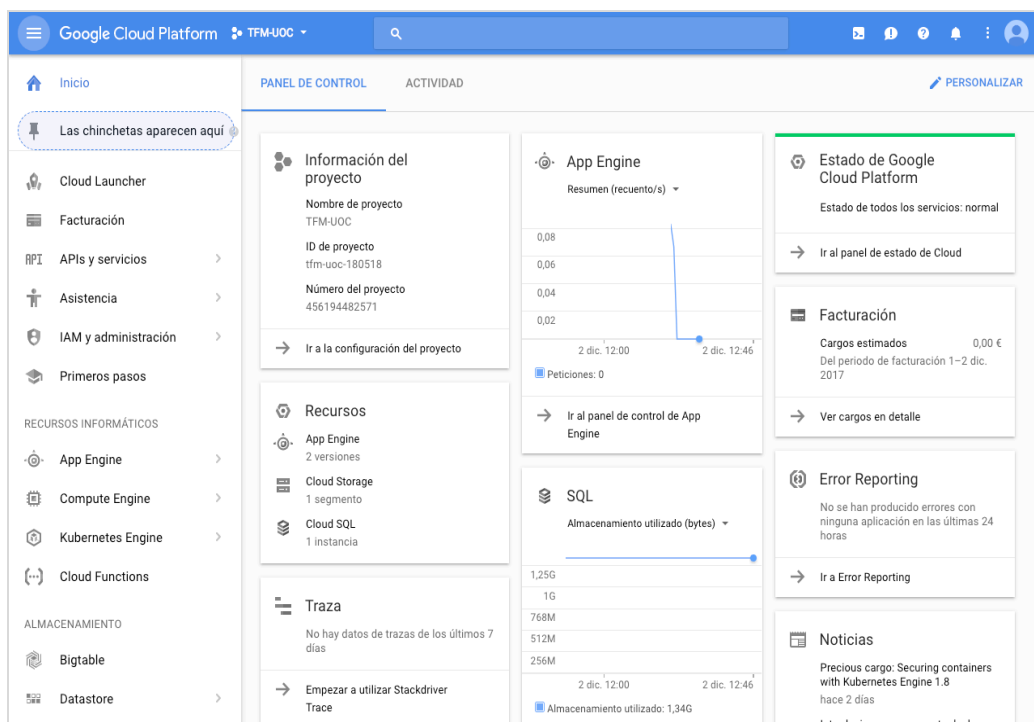


Figura 35. Consola de la plataforma Google Cloud.

De la plataforma solo utilizaremos dos servicios: *App Engine* y *SQL*.

El primer servicio nos permite alojar el *backend* que se encargará de realizar todas las peticiones a la base de datos y ofrecerlos a nuestra aplicación. El segundo servicio lo utilizamos para alojar la base de datos que utilizará la aplicación.

Para poder utilizar la plataforma, debemos disponer de una cuenta de Gmail. Accedemos a la web de Google Cloud (<https://cloud.google.com>) y seleccionamos “Pruébalo gratis”.

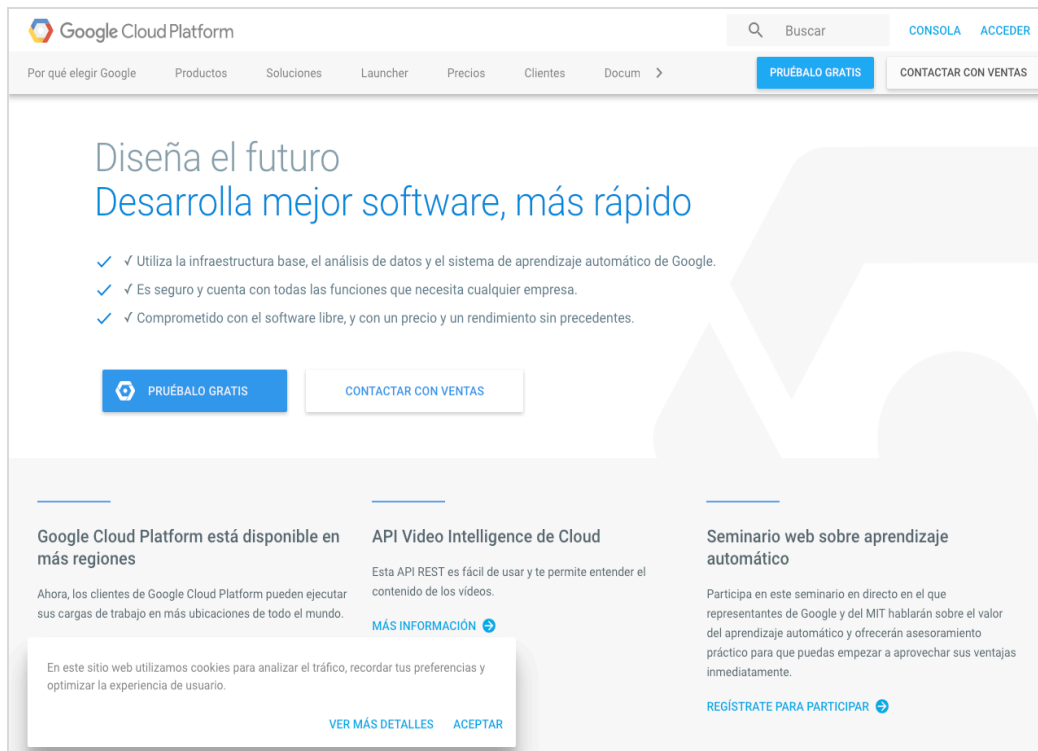


Figura 36. Pantalla inicial de Google Cloud.

El siguiente paso, será realizar el *login* con la cuenta de Gmail y accederemos a la consola principal.

Lo primero de todo, será crear un nuevo proyecto y para ello lo haremos desde la opción “Selecciona un proyecto”.

Se abrirá una nueva ventana donde seleccionaremos la opción de “+” para crear un nuevo proyecto.

En la siguiente pantalla, ponemos el nombre del proyecto y pulsamos sobre el botón “Crear”.



Figura 37. Pantalla introducir el nombre del proyecto.

Y con todos estos pasos ya tendremos acceso a la consola de nuestro nuevo proyecto.

Ahora desde Android Studio ya podemos crear una App Engine^[8] y publicarla en la plataforma de Google Cloud tal como veremos en los siguientes apartados.

Por último, creamos la base de datos donde se guardarán los datos que utilizará la aplicación.

Para poder crear una base de datos, seleccionamos la opción de SQL en el menú lateral. Tendremos que activar la facturación y acto seguido ya podemos crear una instancia nueva.

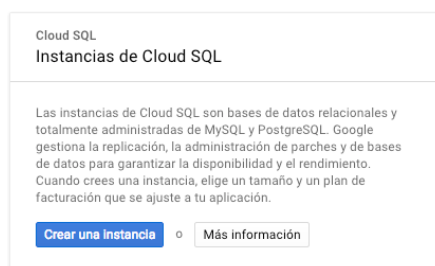


Figura 38. Crear instancia Cloud SQL

Nos dará a elegir entre dos opciones: MySQL y PostgreSQL. En este proyecto se ha utilizado MySQL.

Por último, solo queda completar la información de la instancia, como el identificador, la contraseña de acceso y la ubicación de la misma.

Figura 39. Formulario de creación de una instancia nueva.

Una vez completados todo el proceso, tendremos disponible una instancia nueva y solo quedará crear las tablas necesarias. La creación de las tablas y consultas sobre las mismas, lo podemos hacer desde la misma consola que nos ofrece Google Cloud SQL o sincronizar la instancia con un gestor de base de datos.

MySQL Workbench

El uso de este gestor no es imprescindible. Se ha utilizado para agilizar un poco más el trabajo de creación de tablas y gestión e inserción de datos en la misma.

Este gestor nos permite sincronizar una base de datos de Google Cloud SQL y poder interactuar con ella. Todos los datos que se borren, inserten o actualicen se replicará a la instancia que tengamos en Google Cloud SQL.

Para poder hacer todo esto, necesitamos conocer la IP pública de nuestro equipo. Esto lo podemos hacer, por ejemplo, des de la web <https://cual-es-mi-ip-publica.com/>.

Ahora, dentro de la plataforma de Google Cloud Platform accedemos a nuestra instancia de MySQL creada desde la opción SQL del menú lateral.

Dentro, tendremos que acceder a la opción “Autorización”, tal como se muestra en la imagen.

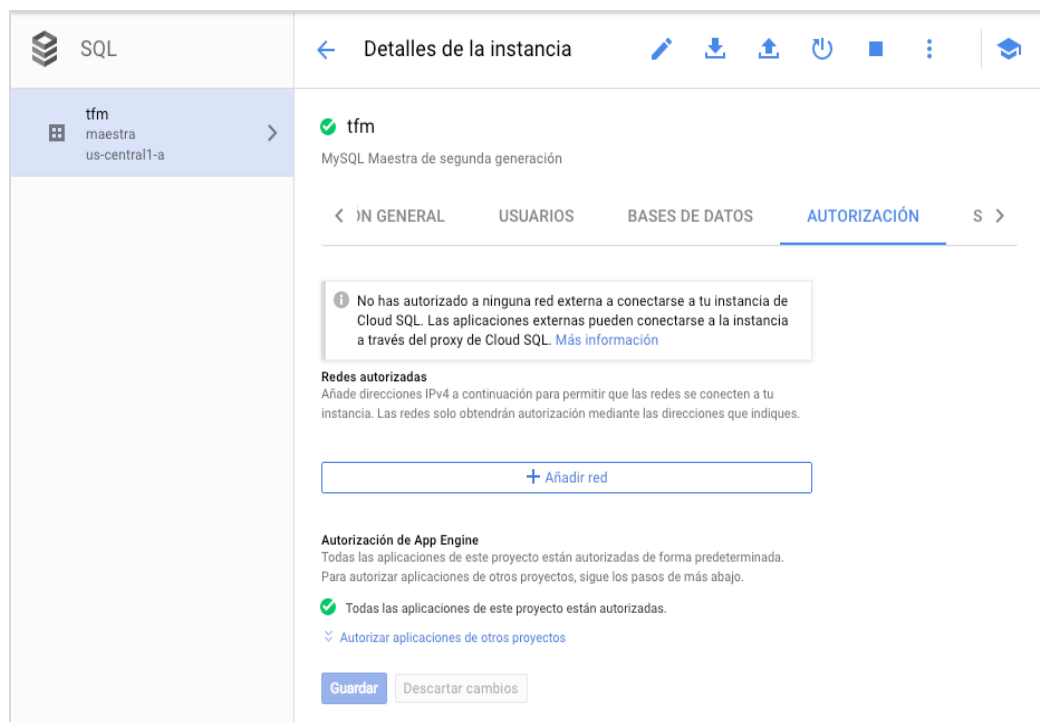


Figura 40. Pantalla de autorización de Google Cloud SQL

Tenemos que seleccionar “Añadir red” y nos solicitará que introduzcamos un nombre para la conexión y la IP que hemos obtenido en la web indicada anteriormente.

Figura 41. Pantalla de añadir nueva red.

Para finalizar, ahora solo queda sincronizar la instancia de Google Cloud SQL con MySQL Workbench.

Primero, necesitamos la IP de la instancia de Cloud SQL. La podemos obtener desde la pantalla principal de la instancia.

Conectarse a esta instancia

Dirección IPv4
35.193.32.115

Nombre de conexión de instancia
tfm-uoc-187812:us-central1:tfm

Conectarse a través de Cloud Shell

Ver todos los métodos de conexión

Configuración

vCPU	Memoria	Almacenamiento: SS
1	3,75 GB	10 GB

- La versión de la base de datos es MySQL 5.7
- El aumento automático del almacenamiento está habilitado
- Ubicada en us-central1-a
- Sin alta disponibilidad (zona)

Figura 42. Pantalla principal de la instancia Cloud SQL.

Después, en la pantalla inicial de MySQL Workbench seleccionamos crear una nueva conexión y en la ventana que nos aparece completamos todos los datos necesarios para crear ésta.

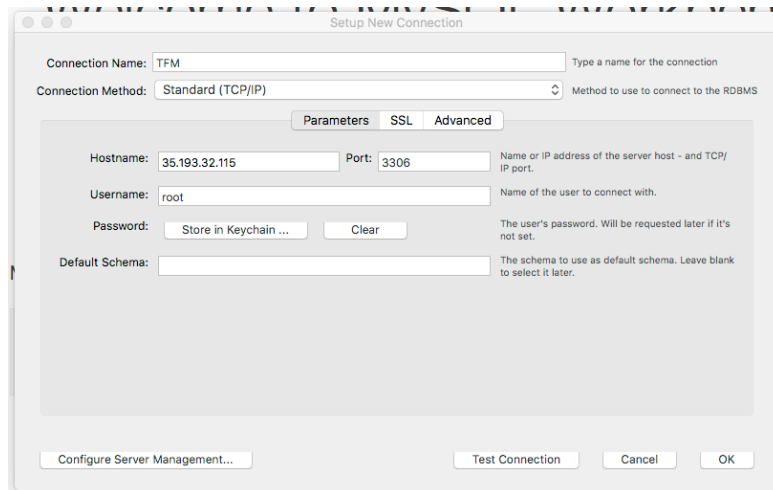


Figura 43. Ventana de configurar nueva conexión MySQL

Y con esto, ya tendremos conexión a nuestra instancia de Google Cloud SQL.

Gitlab

Gitlab es un servicio web de control de versiones basado en Git. En este proyecto se ha utilizado como gestor de repositorios para poder trabajar desde diferentes equipos con el mismo código y así también tener el código fuente en un mismo servidor.

6.3 Librerías y API

Para el desarrollo de BttRoute, se han utilizado varias librerías y a continuación se detallan algunas de las más importantes.

Fabric (<https://fabric.io/home>)

La librería que nos ofrece Fabric nos permite gestionar los errores cuando la aplicación se comporte de una forma incorrecta. Cuando suceda un error en la aplicación, recibiremos un mail en la cuenta configurada en el servicio especificando la clase y línea dónde se ha producido el error.

Osmdroid (<https://github.com/osmdroid/osmdroid>)

Para la muestra de rutas y la creación de estas, se ha decidido utilizar la librería de Osmdroid^[9] que utiliza mapas basados en Open Street Map. Podemos encontrar varias librerías basadas en los mapas de Open Street Map como por ejemplo:

- Mapquest (<https://www.mapquest.com/>)
- Mapbox (<https://www.mapbox.com/>)

El motivo por el que se ha escogido esta librería y no por ejemplo Google Maps, es que nos ofrece la posibilidad de ver caminos de montaña que por ejemplo no podemos ver en Google Maps. A continuación se muestran dos ejemplos de cómo se puede apreciar los detalles de uno y otro mapa:

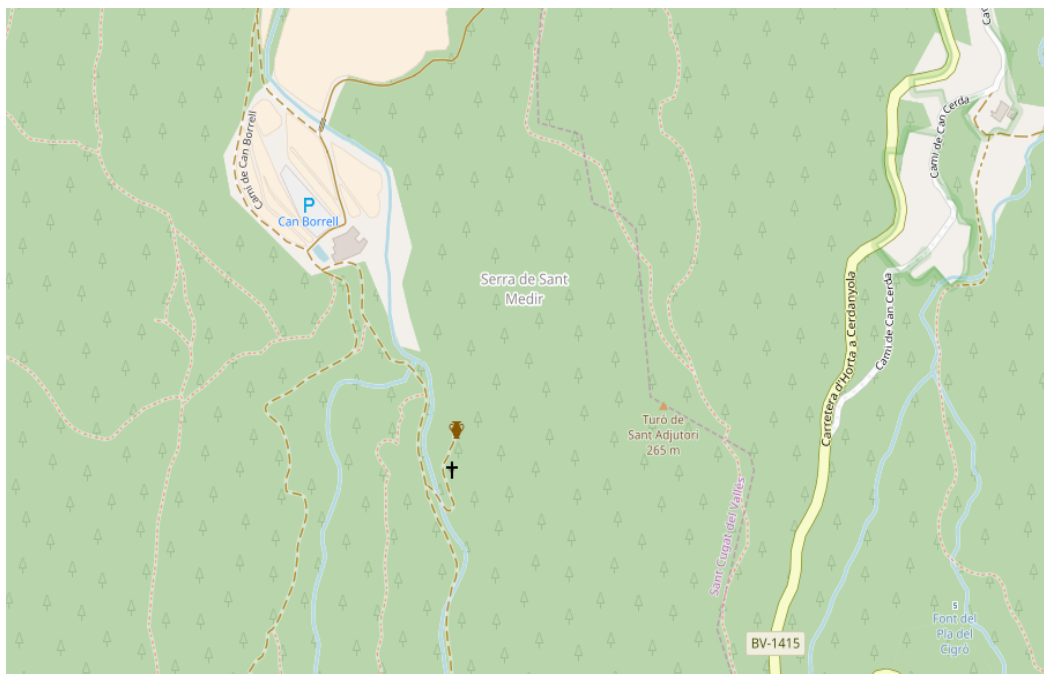


Figura 44. Mapa de Open Street Map.

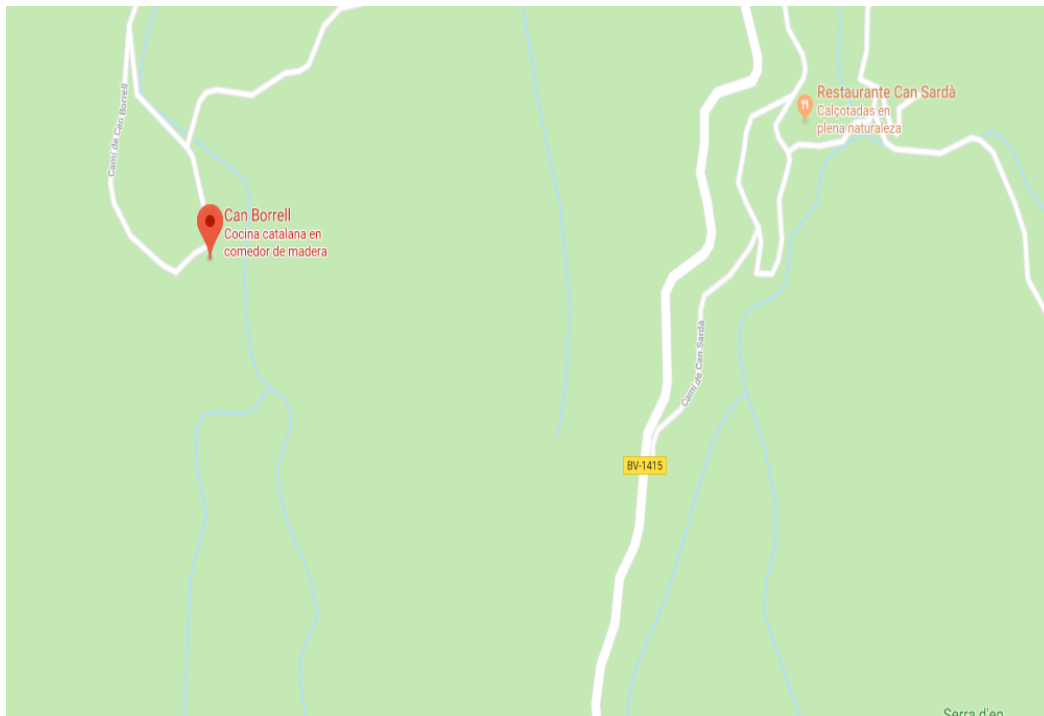


Figura 45. Mapa de Google Maps.

Dado que la app desarrollada en este proyecto se basa en crear rutas de montaña en mountain bike, esta librería se adapta perfectamente a nuestras necesidades.

Picasso (<http://square.github.io/picasso/>)

Esta librería nos permitirá cargar los mapas estáticos que nos proporcionará la API de Mapquest. Se ha elegido esta librería por la sencillez que nos da a la hora de cargar una imagen obtenida a partir de una URL.

Facebook^[10]

Utilizamos la librería de Facebook para implementar la posibilidad de acceder a la aplicación a través de la cuenta de Facebook del propio usuario.

Para poder aplicar el acceso mediante cuenta de Facebook en nuestra aplicación, debemos crear una nueva aplicación en la web de desarrolladores <https://developers.facebook.com/>.

Una vez dentro de la web, accedemos a *Mis aplicaciones* -> *Añadir una nueva aplicación*.

Cuando ya esté creada la aplicación, accedemos a la consola de la misma y solo nos quedaría seleccionar el tipo de producto que queremos, en nuestro caso, "Inicio de sesión con Facebook".

Esto nos llevará a una pantalla donde nos indicará los pasos necesarios para poder implementar un inicio de sesión con una cuenta de Facebook en nuestra app.

Mapquest^[11] (<https://www.mapquest.com/>)

Para mostrar los mapas estáticos que se muestran en alguna de las pantallas de la aplicación, se ha recurrido a la API de Mapquest.

Esta API nos ofrece la posibilidad de a través de una URL y unas coordenadas codificadas, obtener una imagen con la ruta marcada.

Google Location Services

(<https://developer.android.com/reference/android/location/package-summary.html>)

A través de esta API podemos obtener datos y accesos a diferentes servicios del GPS del dispositivo, como por ejemplo:

- Recibir notificaciones cuando hay un cambio de estado en el GPS.^{[12][13]}
- Obtener información sobre la localización del usuario.^{[14][15]}
- Obtener acceso al servicio de localización.^[16]

JavaMail-android (<https://code.google.com/archive/p/javamail-android/downloads>)

Para poder enviar desde el dispositivo un mail a la cuenta del usuario tenemos que utilizar tres librerías que se pueden descargar desde la web indicada. Éstas son: additional.jar, mail.jar y activación.jar. Así nos evitamos tener que utilizar el gestor de correo que tengamos en el dispositivo para enviar un mail.

6.4 Desarrollo

Como se ha comentado al inicio de este capítulo, el proyecto de Android Studio estará compuesto por dos módulos, el de la aplicación y el de *backend*.

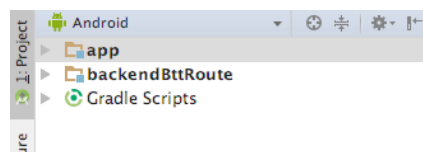


Figura 46. Módulos del proyecto Android Studio.

6.4.1 Estructura del módulo “app”

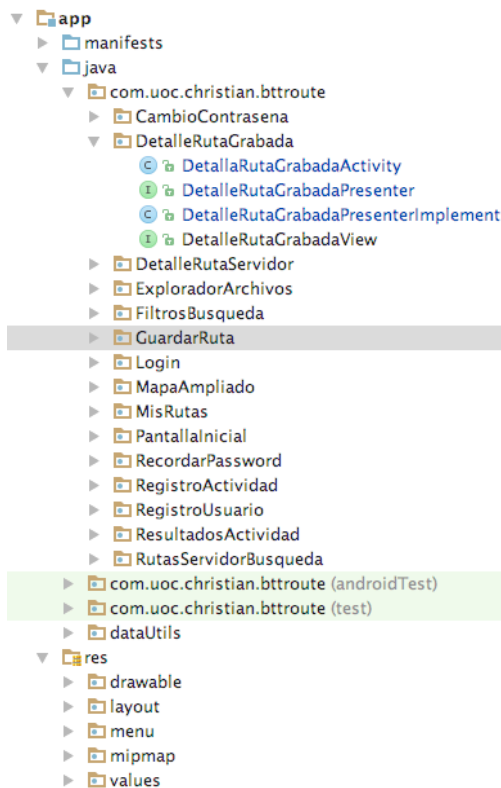


Figura 47. Módulo app

El módulo app contiene todos los archivos necesarios para ejecutar la aplicación, tanto la implementación de la lógica como los *layouts* correspondientes a las pantallas e imágenes. Dentro del módulo, podemos ver el *package* **com.uoc.christian.bttroute** que contiene toda la lógica de negocio. Dentro del *package* podemos ver otro subconjunto de *packages* y cada uno de ellos pertenece a las diferentes pantallas de la aplicación.

Cada uno de los *package* contiene las interfaces y las clases de las distintas pantallas.

A continuación se detallan cada una de las *activities* o *fragments* de la aplicación:

CambioContraseña: En esta *activity* se gestionará la opción de solicitar al usuario una contraseña nueva. El usuario deberá introducir el usuario y automáticamente se creará una contraseña aleatoria y se enviará la email que introdujo el usuario para el alta en el sistema.

DetalleRutaGrabada: En esta *activity* se llevará todo el proceso para mostrar en pantalla los datos de una ruta grabada por el propio usuario. En esta pantalla estará disponible las opciones de compartir ruta y exportar ruta.

DetalleRutaServidor: Al igual que “DetalleRutaGrabada”, en este caso se cargarán los datos de una ruta compartida por otro usuario. En este caso tendremos sólo disponible la opción de exportar ruta.

ExploradorArchivos: Este *fragment* se encargará de mostrar todos los archivos con extensión .GPX que se encuentran dentro de la carpeta de “BttRouteMaps” que se crea cuando se ejecuta por primera vez la aplicación.

FiltrosBusqueda: Esta *activity* se encargará de la gestión de filtros que el usuario puede utilizar para filtrar rutas según sus características. Una vez introducidos los filtros a aplicar, se realiza el proceso de búsqueda.

GuardarRuta: Esta *activity* se encarga de registrar en la base de datos una nueva ruta creada por el usuario a partir de la *activity* “Registro actividad”. La ruta se guardará pero solo será visible para el propio usuario.

Login: Esta es la *activity* principal de la aplicación. Es la que gestiona el acceso a la misma y da acceso a las *activities* de “RecordarPassword” y “RegistroUsuario”. La *activity* podrá validar al usuario a través de una cuenta de Facebook o de una cuenta registrada en el sistema.

MapaAmpliado: La *activity* se encargará de mostrar el track de la ruta en el mapa para que el usuario pueda observarla con detenimiento y con más detalle. Las acciones que podrá hacer el usuario sobre el mapa son: rotar el mapa, zoom y desplazarse por él.

Mis rutas: Este *fragment* se encargará de cargar las rutas creadas por el propio usuario. La pantalla cargará las rutas de cuatro en cuatro, de forma que el usuario podrá cargar más cuando realice un scroll deslizando el dedo desde la parte inferior de la lista hacia arriba. Además, desde esta pantalla seleccionando una ruta, podrá acceder a la *activity* “DetalleRutaGrabada”.

PantallaInicial: Esta *activity* se encargará de gestionar el comportamiento del menú principal de la aplicación y dará acceso al usuario a las diferentes pantallas.

RecordarPassword: Cuando el usuario no recuerde la contraseña, desde la *activity* “Login” podrá acceder a la pantalla para solicitar una contraseña nueva. Se solicitará al usuario el nombre con el que se ha registrado y la **activity** se encargará de validar que el nombre de usuario es correcto, para después enviar a su cuenta de email una contraseña creada aleatoriamente.

RegistroActividad: Desde esta *activity* se crea una nueva ruta. Se controla el tiempo de ruta, distancia, velocidad, velocidad media y la altitud. Además se encarga de pintar el trazado recorrido en el caso de que sea una ruta nueva, así como de mostrar el trazado de una ruta cargada a través de un archivo GPX. Se ha decidido utilizar como proveedor el GPS ya que ofrece mayor precisión que el proveedor NETWORK, aunque esto conlleva a un mayor consumo de batería. Para registrar los puntos de geolocalización, se ha optado por obtener una nueva posición cada segundo y guardar la localización a partir de los 10 metros. A través de la función *requestLocationUpdates* (*proveedor, tiempo, distancia, listener*) se pueden configurar estos parámetros. Pero en nuestro caso, con esta función solo recibiremos la actualización con la variable de tiempo, Si se utilizaba la variable distancia esto provoca que los datos de velocidad que nos ofrece la localización no se actualizaban correctamente en la app. Así, que se ha decidido configurar en esta función únicamente la actualización a través de la variable de tiempo y la de distancia a través de una función propia.

La decisión de guardar la localización a partir de cada 10 metros, se basa principalmente para no tener problemas a la hora de obtener un mapa estático de la ruta. En el caso de realizar una ruta muy larga, obtendremos un número muy elevado de geolocalizaciones. A la hora de codificar éstas, tendremos una cadena muy larga que en ocasiones creará una dirección URL con una longitud superior a la que acepta un navegador, con lo que no nos podrá devolver un mapa estático de la ruta creada.

RegistroUsuario: Con esta *activity* la aplicación registra un nuevo usuario en el sistema. Valida que no exista un usuario con el mismo nombre y que todos los campos estén completados.

ResultadosActividad: Cuando el usuario finaliza una ruta desde “RegistroActividad” accederemos a ésta. Aquí se muestran los datos registrados por el dispositivo: distancia, tiempo, velocidad, velocidad media, desnivel positivo, desnivel negativo y desnivel acumulado. Además, a través de la API de Mapquest obtendremos una imagen estática del track de la ruta. Desde aquí el usuario podrá acceder la *activity* de “GuardarRuta” o eliminar la ruta.

RutasServidorBusqueda: Inicialmente, esta *activity* se encarga de mostrar al usuario todas las rutas que comienzan a una distancia máxima de 5 km desde el punto que se encuentra. Desde esta *activity* se podrá acceder a la de “FiltrosBusqueda” para obtener rutas según unos criterios.

6.4.2 Estructura del módulo “backendBttRoute”

Dentro de este módulo encontramos todas las clases que representan cada una de las entidades definidas de la base de datos y los *endpoints* que se publicarán en la App Engine de Google Cloud.

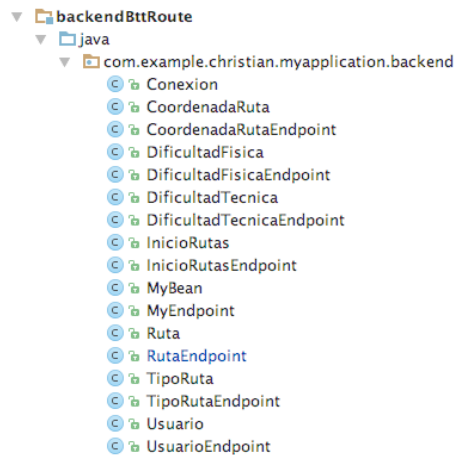


Figura 48. Estructura del módulo “backendBttRoute”.

La versión de Android Studio 2.3.3 nos permite crear un módulo App Engine^[17] que podremos publicar en Google Cloud.

Para hacerlo, se procederá de la siguiente forma:

File => New => New Module

Se mostrará una ventana con los diferentes tipos módulos que podemos implementar y seleccionamos “Google Cloud Module”.

Esto nos creará un nuevo módulo con un pequeño ejemplo “MyBean” y “MyEndpoint”.

A partir de aquí crearemos las entidades necesarias para el funcionamiento de nuestra aplicación. Para ello tendremos que definir cada entidad con la clave `@Entity`.

Ejemplo de la entidad *DificultadFisica*:

```
@Entity
public class DificultadFisica {

    @Id
    Long Id;
    String Descripcion;

    public DificultadFisica(){

    }

    public void setId(Long id) { Id = id; }

    public void setDescripcion(String descripcion) { Descripcion = descripcion; }

    public Long getId() { return Id; }

    public String getDescripcion() { return Descripcion; }

}
```

Figura 49. Entidad correspondiente a la tabla “DificultadFisica”.

Una vez creada la entidad, crearemos su *endpoint* haciendo click con el botón derecho sobre la entidad creada y seleccionando la opción “Generate Cloud Endpoint from Java Class”.

El *endpoint* creado, contendrá todas las operaciones CRUD que podemos realizar.

Ejemplo de *endpoint* perteneciente a la entidad Ruta:

```
public class RutaEndpoint {  
  
    private static final Logger logger = Logger.getLogger(RutaEndpoint.class.getName());  
  
    private static final int DEFAULT_LIST_LIMIT = 4;  
  
    static {  
        // Typically you would register this inside an OfyService wrapper. See: https://code.google.com/p/objectify-appengine/wiki/BestPractices  
        ObjectifyService.register(Ruta.class);  
    }  
  
    /**...*/  
    @ApiMethod(  
        name = "getByIdFacebook",  
        path = "rutaByIdFacebook/{UsuarioFacebook}",  
        httpMethod = ApiMethod.HttpMethod.GET)  
    public List<Ruta> getByIdFacebook(@Named("UsuarioFacebook") String IdFacebook, @Named("NextObjects") Integer nextObjects) throws NotFoundException {...}  
  
    /**...*/  
    @ApiMethod(  
        name = "getById",  
        path = "rutaById/{IdUsuario}",  
        httpMethod = ApiMethod.HttpMethod.GET)  
    public List<Ruta> getById(@Named("IdUsuario") Long IdUsuario, @Named("NextObjects") Integer nextObjects) throws NotFoundException {...}  
  
    /**...*/  
    @ApiMethod(  
        name = "insert",  
        path = "ruta",  
        httpMethod = ApiMethod.HttpMethod.POST)  
    public Ruta insert(Ruta ruta) {...}  
}
```

Figura 50. Ejemplo de clase *endpoint*.

Cuando creamos o modifiquemos alguna entidad o *endpoint* tendremos que actualizar nuestro proyecto accediendo a menú *Build => Clean Project* y después publicarlo en la App Engine de Google Cloud a través del menú *Build => Deploy Module to App Engine*.

Esta acción nos mostrará una ventana en la que se mostrará el módulo de origen, el proyecto que queremos actualizar de Google Cloud y la versión.

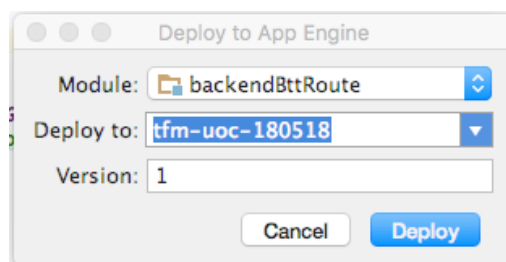


Figura 51. Ventana Deploy to App Engine de Android Studio.

A continuación se detallan los *endpoints* que forman el *backend*:

DificultadFisicaEndpoint: Este *endpoint* contendrá el método que nos devolverá una lista con los registros que contiene la tabla “DificultadFisica”.

DificultadTecnicaEndpoint: Igual que el anterior *endpoint*, pero nos devolverá una lista con los registros que contiene la tabla “DificultadTecnica”.

InicioRutasEndpoint: En este *endpoint* tenemos dos métodos, el que inserta un nuevo registro en la tabla “InicioRutas” y el que nos devuelve el registro que contiene la localización de inicio de una ruta que esté compartida.

RutaEndpoint: Este es el *endpoint* más importante, ya que es el que se encargará de recuperar todos los datos de una ruta, insertar una nueva ruta o recuperar según parámetros de búsqueda o posición inicial del usuario.

TipoRutaEndpoint: Al igual que “DificultadFisicaEndpoint” y “DificultadTecnicaEndpoint”, nos devolverá una lista con todos los registros de la tabla “TipoRuta”.

UsuarioEndpoint: Este *endpoint* tendrá todos los métodos para comprobar si existe un usuario, registrar un nuevo usuario, actualizar la contraseña, o obtener un usuario a partir de su nombre de registro y contraseña.

6.5 Pruebas realizadas

Para realizar las pruebas se ha utilizado un dispositivo físico y el emulador que nos proporciona Android Studio.

Características dispositivo

- Smartphone Samsung S5 neo (SM-G903F) con sistema operativo Android Marshmallow (6.0.1 - API 23)

Características del emulador

- Pixel XL con Android Lollipop (5.1 – API 22).
- Resolución 1440x2560
- 1,5 GB de RAM
- 2 Multi-Core CPU

Se han realizado todo tipo de pruebas para comprobar el funcionamiento de la aplicación y que se detallan a continuación:

Registrar usuario		
Prueba	Detalles	Resultado
Registrar un nuevo usuario	Se completan todos los campos.	OK.
Registrar usuario con un nombre ya registrado.	Se completan todos los campos.	OK. Se devuelve un mensaje de error si el nombre ya existe

Registrar un nuevo usuario	Se deja algún campo vacío.	OK. Se devuelve un mensaje advirtiendo que se deben completar todos los campos.
Registrar un nuevo usuario	Se ponen contraseñas diferentes en los dos campos.	OK. Se devuelve un mensaje advirtiendo de que las contraseñas no coinciden.

Solicitar contraseña nueva		
Prueba	Detalles	Resultado
Obtener una contraseña nueva.	Se introduce el nombre de usuario.	OK. Se genera una nueva contraseña y se envía al mail del usuario
Obtener una contraseña nueva.	No se introduce ningún usuario.	OK. Se devuelve un mensaje de error.
Obtener una contraseña nueva.	Se introduce el nombre de un usuario que no existe.	OK. Se devuelve un mensaje de error informando de que el usuario no existe.

Login		
Prueba	Detalles	Resultado
Login	No se introducen todos los datos.	OK. Se devuelve un mensaje advirtiendo de que se deben completar los datos.
Login	Se introducen todos datos correctos	OK. Se accede a dentro de la aplicación.
Login	Se introducen todos los datos pero incorrectos	OK. Se devuelve un mensaje al usuario advirtiendo de un error de validación.

Cambio contraseña actual		
Prueba	Detalles	Resultado
Cambiar contraseña	Se introducen todos los campos correctamente.	OK.
Cambiar contraseña	No se introducen todos los campos.	OK. Se devuelve un mensaje informando que se deben completar todos los campos.
Cambiar contraseña	Se introducen todos los datos pero la contraseña nueva no coincide con la repetición.	OK. Se devuelve un mensaje informando que la nueva contraseña no es igual a la repetida.
Cambiar contraseña	Se introducen todos los datos pero la contraseña actual no coincide.	OK. Se devuelve un mensaje informando que la contraseña actual no es la misma que la registrada.

Crear ruta nueva		
Prueba	Detalles	Resultado
Crear ruta	Se crea una ruta nueva	OK. Si pinta el camino, se guardan las posiciones y se muestran los datos correctamente.
Crear ruta	Se crea ruta a partir de ruta cargada de un archivo GPX	OK. Se guardan las posiciones y se muestran los datos correctamente.

Guardar ruta		
Prueba	Detalles	Resultado
Grabar ruta creada	Se guarda la ruta completando todos los campos.	OK. Se guarda una nueva ruta en el <i>backend</i> .
Grabar ruta creada	Se guarda la ruta sin poner título.	OK. Se muestra un mensaje advirtiendo de que debe introducir un título a la ruta

Compartir ruta		
Prueba	Detalles	Resultado
Compartir ruta grabada	Se comparte una ruta grabada.	OK.
Compartir ruta grabada	Se intenta compartir una ruta que ya hemos compartido anteriormente.	OK. Se devuelve un mensaje al usuario advirtiendo que esa ruta ya está compartida.
Compartir ruta grabada	Se intenta compartir una ruta grabada a partir de un archivo GPX exportado desde nuestra aplicación.	OK. Se devuelve un mensaje al usuario de que la ruta ya existe en el servidor.

Buscar rutas		
Prueba	Detalles	Resultado
Buscar rutas	Buscar rutas compartidas por otros usuarios sin ningún filtro.	OK. Se obtienen todas las rutas compartidas en el que su inicio está en un radio de 5 km desde nuestra posición.
Buscar rutas	Buscar rutas compartidas por otros usuarios con filtros.	OK. Se obtienen todas las rutas que cumplen con los filtros.

Nota: datos con los lugares que se ha probado en los filtros: Rubí, Sant Cugat, Sant Pere de Riudebitlles, Montserrat, Terrassa, Tibidabo.

Exportar ruta		
Prueba	Detalles	Resultado
Exportar ruta	Se exporta una ruta compartida en el servidor.	OK. Obtenemos un fichero GPX en la ruta generada por la aplicación.

6.6 Cambios y mejoras

A raíz de investigar sobre el tratamiento de las geoposiciones en bases de datos y en otro tipo de aplicaciones se ha observado que el diagrama de la base de datos no era el más apropiado.

En el nuevo modelo, se ha decidido prescindir de dos tablas: “CoordenadasRuta” y “RutasProximas”, modificar la tabla “Ruta” y crear una tabla nueva, “InicioRutas”.

La tabla “CoordenadasRuta” se había creado para guardar todas las localizaciones de una ruta. El inconveniente es que cada ruta podía estar compuesta de 1500, 2000, 1000, 500,... registros. Esto haría que tuviéramos una tabla muy grande y a medida que fuese incrementando el tamaño, las lecturas serían más costosas con lo que relentecería la aplicación.

Buscando por la red, se ha encontrado información de cómo poder codificar todas esas posiciones obteniendo una única cadena de texto. Esta codificación se puede realizar a través de una función que utiliza Google^[18], lo que nos permite que en lugar de tener por ejemplo 2000 registros, tengamos sólo 1.

Como resultado de la eliminación de esta tabla, se ha añadido una nueva columna en la tabla “Ruta” que contiene las localizaciones codificadas.

Posteriormente, se puede descodificar esta cadena y obtener todas las localizaciones.

La tabla “RutasProximas” ha sido eliminada y en su lugar se ha creado la tabla “InicioRutas”. En un principio se pensaba utilizar la primera tabla para guardar los *id*'s de las rutas próximas a otra ruta, pero más tarde se vió que no se podía utilizar ésta para obtener las rutas próximas a la posición de un usuario. Además, se encontró una solución que podía cumplir las exigencias requeridas, obtener las rutas próximas al usuario y a otras rutas.

Esto lo podemos hacer gracias a la función geoespacial **ST_Distance Sphere (POINT (latitud1,longitud1), POINT (latitud2, longitud2))**^[19] que dispone MySQL donde a partir de una coordenada de inicio y otra final podemos obtener la distancia que hay entre las dos.

Por ello, solo nos haría falta guardar el punto de inicio de cada ruta en la tabla “InicioRutas”, y pasar como parámetro la posición del usuario o el punto inicial de otra ruta. Se guardará la posición inicial de una ruta únicamente en el supuesto que el usuario haya decidido compartir.

Con estos cambios, el nuevo diseño de la base de datos quedaría de la siguiente forma:

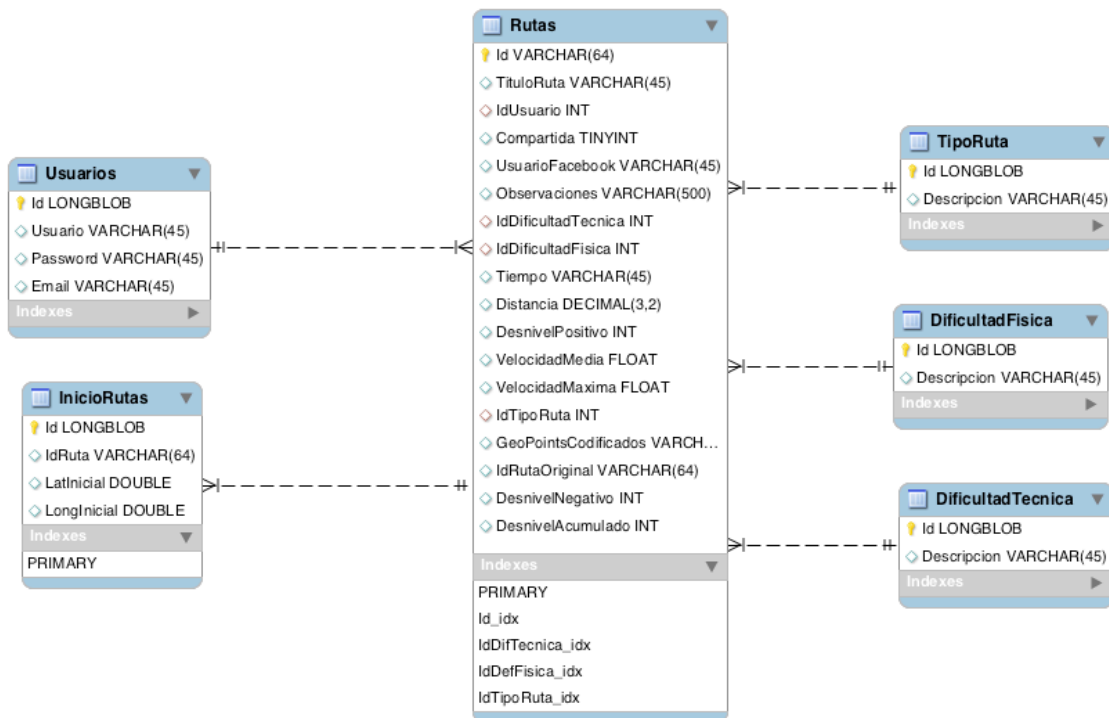


Figura 52. Diseño definitivo de la base de datos.

7. Conclusiones

La realización de este proyecto me ha servido para aplicar los conocimientos adquiridos durante el tiempo que ha durado el Máster. He aprendido cómo planificar un proyecto de desarrollo de una app partiendo desde la base con la toma de requerimientos, análisis de usuarios potenciales, diseño de la arquitectura, diseño de prototipo y finalizando con el desarrollo final. Viendo el resultado final de este trabajo creo que ha cumplido completamente con mis expectativas, a pesar de que en el inicio surgieron dudas sobre si sería posible desarrollar todas las ideas expuestas.

Los objetivos propuestos inicialmente han sido todos cumplidos, aunque a medida que avanzaba el desarrollo, se ha tenido que cambiar la forma de obtener alguno de éstos. Estos cambios eran debido al desconocimiento sobre ciertos campos, y gracias a la investigación he podido conseguir algunos objetivos de una forma más correcta a como se realizaba en un principio.

En cuanto a la planificación, las primeras fases y última se han seguido estrictamente. En cambio en la fase de desarrollo se han incrementado el número de horas necesarias.

La metodología elegida ha sido la adecuada para el desarrollo y ha posibilitado realizar las entregas en los tiempos marcados. Esta metodología, ha permitido dividir el desarrollo de la aplicación en diferentes fases:

- Fase 1: Gestión de usuarios.
- Fase 2: Desarrollo creación y carga de *tracks*.
- Fase 3: Gestión de rutas en la aplicación y base de datos (proceso de guardar, compartir, descargar).

Tal como se ha comentado en el anterior punto, para conseguir algunos de los objetivos ha sido necesario introducir pequeños cambios en el modelo de datos. Por parte de la aplicación, no ha habido cambios importantes, sino que han sido más en cuanto a apariencia, además de incluir algún parámetro más sobre las rutas, como por ejemplo mostrar el desnivel negativo y el acumulado.

Desde el inicio del proyecto se quedaron sin explorar algunas ideas que ayudarían a mejorar nuestra aplicación, así como otras que surgieron durante el desarrollo. En un futuro se podrían añadir siguientes ideas:

- Permitir cargar/exportar archivos KML. Actualmente, sólo se pueden exportar y cargar archivos GPX.
- Implementar la posibilidad de realizar el *login* con una cuenta de Google.
- Mostrar el sentido de la dirección de la ruta cargada. Este es un punto importante ya que cuando cargamos un archivo GPX se muestra únicamente la ruta, aparte del punto de partida y final de la ruta. Dependiendo como sea la ruta puede ocurrir que en algunos momentos,

diferentes puntos se crucen en un determinado tramo, y este hecho puede llegar a confundir al usuario para decidir que camino seleccionar.

- Otra mejora que se podría implementar es variar el color de la ruta cargada para saber qué tramos son más exigentes físicamente. Esto se podría hacer comprobando la altitud en cada punto de localización del archivo GPX y comprobar la variación. Esta funcionalidad podría ayudar al usuario a dosificar mejor las fuerzas y planificarse mejor.
- Poder eliminar un archivo GPX del directorio. El explorador que dispone la aplicación tan sólo permite ejecutar archivos que han sido descargados del servidor. O también, cargándolos conectando el dispositivo a un PC y copiando los archivos en el directorio. La idea sería que, dentro del explorador, si mantenemos pulsado un archivo se diera la posibilidad de borrarlo.
- Posibilidad de que un usuario introduzca comentarios. En estos momentos solo se almacenan las observaciones que introduce un usuario cuando guarda una ruta. Se podría incorporar la funcionalidad que permitiera a otros usuarios escribir comentarios sobre otras rutas compartidas.
- Desactivar la compartición de una ruta. Cuando un usuario decide compartir una ruta, ésta se queda compartida para siempre. Puede suceder que en algún momento, un usuario decida preferir dejar de compartir una ruta en concreto.

8. Glosario

App: Application.

GPX: (GPS Exchange Format) Formato de datos XML para el intercambio de datos GPS entre aplicaciones y Servicios Web en Internet.

GPS: (Global Positioning System) Sistema de Posicionamiento Global.

Framework: Conjunto de librerías, herramientas o esquemas que podemos utilizar desarrollar nuestra aplicación.

API: (Application Programming Interface) Conjunto de subrutinas, funciones y procedimientos que ofrece una biblioteca.

SQL: (Structured Query Language) Lenguaje de consulta estructurada. Lenguaje específico que da acceso a un sistema de gestión de base de datos.

IT: Tecnología de la información.

SDK: (Software Development Kit) Kit de desarrollo de software. Conjunto de herramientas de desarrollo que permite al programador crear una aplicación para un sistema concreto.

IDE: (Integrated Development Environment) Entorno de Desarrollo Integrado. Aplicación informática que proporciona servicios integrales para facilitar al desarrollador el desarrollo de software.

CRUD: (Create, Read, Update and Delete) Crear, Leer, Actualizar y Borrar. Se refiere a las funciones básicas en bases de datos o la capa de persistencia de un software.

RAM: (Random Access Memory) Memoria de acceso aleatorio que se utiliza como memoria de trabajo de computadoras para el sistema operativo, programas y software.

CPU: (Central Processing Unit) Unidad central de procesamiento, es el hardware dentro de un ordenador u otros dispositivos programables, que interpretan las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.

9. Bibliografía

- [1] www.gartner.com/newsroom/id/3725117
- [2] <http://www.develapps.com/es/noticias/modelo-vista-presentador-mvp-en-android> (20/10/2017)
- [3] <https://www.bisente.com/documentos/mysql-postgres.html> (20/10/2017)
- [4] <https://cloud.google.com/sql/docs/mysql/> (23/10/2017)
- [5] <https://cloud.google.com/endpoints/docs/frameworks/legacy/v1/java/> (23/10/2017)
- [6] <https://cloud.google.com/appengine/docs/standard/java/?hl=es> (23/10/2017)
- [7] <https://cloud.google.com/getting-started/?hl=es> (25/10/2017)
- [8] <https://www.3pillarglobal.com/insights/building-backend-applications-with-google-app-engine-google-cloud-endpoints-and-android-studio> (25/10/2017)
- [9] <https://github.com/osmdroid/osmdroid/wiki/How-to-use-the-osmdroid-library> (29/10/2017)
- [10] https://developers.facebook.com/docs/facebook-login/android/?locale=es_ES (30/10/2017)
- [11] <https://developer.mapquest.com/documentation/static-map-api/v5/map/#adding-route> (06/11/2017)
- [12] Para API inferior a level 24
<https://developer.android.com/reference/android/location/GpsStatus.Listener.html>
- [13] Para API igual o superior a level 24
<https://developer.android.com/reference/android/location/GnssStatus.Callback.html>
- [14] <https://developer.android.com/reference/android/location/LocationListener.html>
- [15] <https://developer.android.com/reference/android/location/Location.html>

[16] <https://developer.android.com/reference/android/location/LocationManager.html>

[17] <https://www.3pillarglobal.com/insights/building-backend-applications-with-google-app-engine-google-cloud-endpoints-and-android-studio>

[18] <https://developers.google.com/maps/documentation/static-maps/intro#Paths> (05/11/2017)

[19] <https://blogs.oracle.com/jsmyth/calculating-distances-with-mysqls-spatial-data-extensions> (13/11/2017)

10. Anexos

10.1 Compilación y ejecución

Para compilar y ejecutar la aplicación desde el entorno de desarrollo, necesitamos la librerías indicadas en el capítulo 6 “Implementación” y que ya se incluyen en el código fuente.

También se debe tener instalado en Android Studio:

- *SDK Platform-tools 27.0.0*
- *SDK Build -Tools 27.0.1*
- *Google Play services 46*
- *SDK Tools 26.1.1*
- *Android Emulator 26.1.4*
- *Paquete completo de Support Repository*
- *SDK Platforms - Android 5.1 API 22*

En el caso de tener alguno de estos componentes instalados, la propia plataforma nos lo comunicará y nos dará la posibilidad de instalarlo.

Si la aplicación se ejecuta bajo el entorno de desarrollo de Android Studio, la funcionalidad de hacer el *login* a través de una cuenta de Facebook no será posible. Esto es debido a que en cada máquina que ejecute el código, se necesita generar un *hash key* que se debe introducir en la consola de desarrollo de Facebook.

Dentro del código fuente de la aplicación, exactamente en la clase “LoginActivity”, tenemos un método que realiza esta operación, creándonos el *hash* necesario.

```
//Generamos una hash key para el login de Facebook. Este hash lo tenemos que introducir en
la consola de desarrollo de facebook en el caso de que estemos debugando la aplicación.
try {
    PackageInfo info = getPackageManager().getPackageInfo(
        "com.uoc.christian.bttroute",
        PackageManager.GET_SIGNATURES);
    for (Signature signature : info.signatures) {
        MessageDigest md = MessageDigest.getInstance("SHA");
        md.update(signature.toByteArray());
        Log.d("KeyHash:", Base64.encodeToString(md.digest(), Base64.DEFAULT));
    }
} catch (PackageManager.NameNotFoundException e) {
    e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

Una vez ejecutada la depuración, en el terminal de Android Studio, buscamos la clave “KeyHash” y se mostrará la línea que contiene el *hash key*.

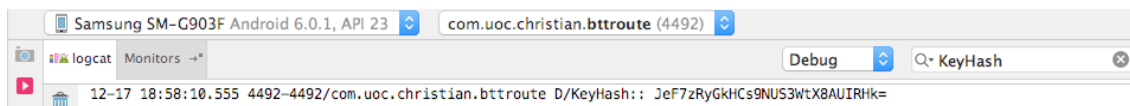


Figura 53. Terminal de Android Studio.

Si se desea probar la funcionalidad de *login* con cuenta Facebook depurando el código, se puede enviar el *hash key* obtenido al email csanzn@uoc.edu y se introducirá en la consola de desarrollo de Facebook.

10.2 Instalación

Para realizar la instalación del APK que se adjunta con el proyecto, primero deberemos activar en nuestro dispositivo la opción que nos permite instalar aplicaciones procedentes de fuentes desconocidas.

Nos dirigimos a “Ajustes => Pantalla Bloqueo y Seguridad => Fuentes desconocidas”.

Finalmente, habrá que copiar el archivo ejecutable en el dispositivo y la instalación se realizará automáticamente.

10.3 Manual de usuario

10.3.1 Registro de un nuevo usuario

Para acceder a la aplicación es necesario haber creado un usuario en el sistema o tener una cuenta de Facebook. Si el usuario decide darse de alta, desde la pantalla de *login* puede acceder al formulario de alta.

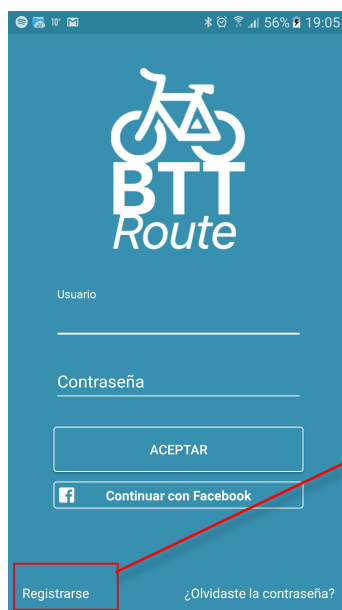


Figura 54. Manual de usuario-Acceso al formulario de registro.

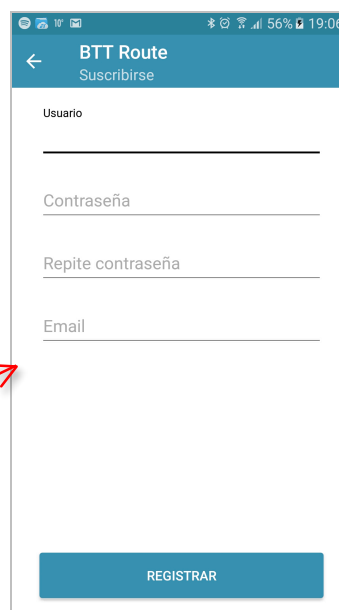


Figura 55. Manual de usuario-Pantalla registro de usuario.

10.3.2 Login

Para realizar el *login* se puede hacer de dos formas, a través de una cuenta dada de alta o con un usuario de Facebook.



Figura 56. Manual de usuario - Tipo de accesos a la aplicación.

1. Acceso a través de una cuenta dada de alta.
2. Acceso a través de una cuenta de Facebook.

Como parte de las pruebas realizadas se ha creado un usuario con el que poder acceder a la aplicación:

Usuario: **esan**
Password: **asd**

10.3.3 Recuperar contraseña

Cuando no nos acordemos de la contraseña, podremos solicitar a la aplicación que nos cree una de nueva y nos la envíe a la cuenta de *mail* con la que nos registramos.

Desde la pantalla de *login*, podemos acceder al formulario para solicitar una nueva.



Figura 57. Manual de usuario- Acceso para solicitar una contraseña nueva.

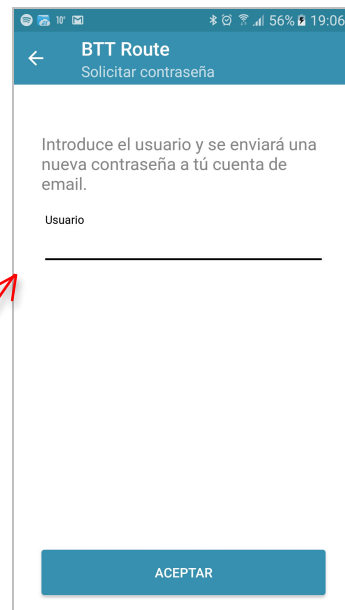


Figura 58. Manual de usuario-Formulario para solicitar una nueva contraseña.

10.3.4 Cambio de contraseña

El usuario, siempre que lo desee, podrá cambiar la contraseña de acceso a la aplicación. Esta opción es recomendable realizarla cuando el usuario haya solicitado una nueva al sistema, y solamente estará disponible si se hace con una cuenta registrada.

Desde el menú lateral, podemos acceder al formulario para cambiar la contraseña.

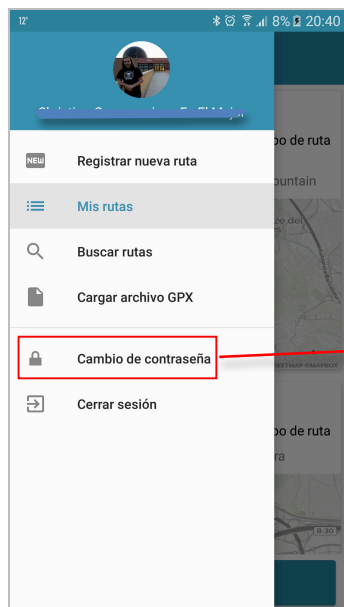


Figura 59. Manual de usuario- Acceso a la pantalla de cambio de contraseña.

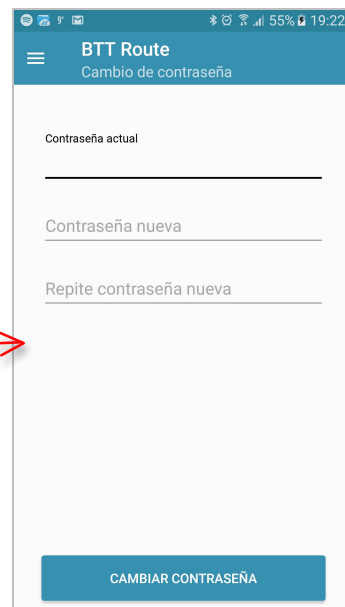


Figura 60. Manual de usuario-Pantalla cambio de contraseña.

10.3.5 Crear una ruta nueva

Para crear una ruta nueva se puede hacer de dos formas:

- Crear una ruta por iniciativa del usuario.
- A partir de un archivo GPX cargado en el dispositivo.

El usuario podrá acceder a la pantalla que registra la actividad desde tres puntos diferentes de la aplicación:



Figura 61. Manual de usuario-Acceso a grabar ruta I

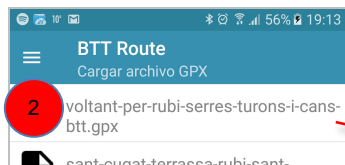


Figura 62. Manual de usuario-Acceso a grabar ruta II

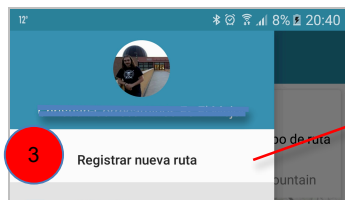


Figura 63. Manual de usuario-Acceso a grabar ruta III

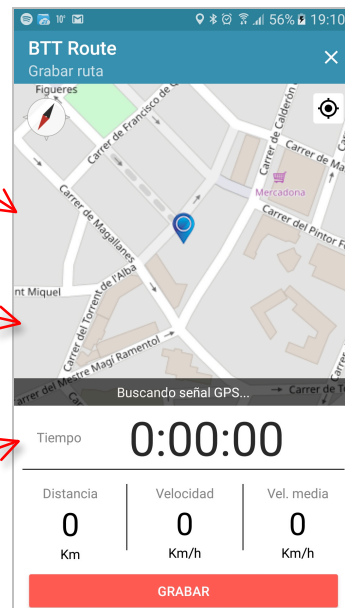



Figura 64. Manual de usuario - Pantalla de grabar ruta.

1. Desde la pantalla inicial de "Mis rutas", presionando sobre el botón "Nueva ruta" accederá a la pantalla de "Grabar ruta".
2. Desde la pantalla de "Cargar archivo GPX", seleccionando un archivo accederá directamente a la pantalla de "Grabar ruta" y además se marcará la ruta.
3. Desde el menú lateral, seleccionando la opción *Registrar nueva ruta* accederá a la pantalla de "Grabar ruta".

10.3.6 Exportar rutas

Desde la propia aplicación el usuario puede exportar en formato GPX tanto sus rutas creadas como las rutas compartidas por otros usuarios. Esta opción la tendrá disponible en la pantalla de detalle de la ruta pulsando sobre el icono .

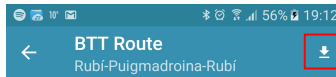



Figura 65. Manual de usuario - Opción exportar ruta.

10.3.7 Compartir ruta

El usuario puede compartir sus rutas creadas con otros usuarios. Para ello, tendrá que acceder al detalle de una ruta propia de la pantalla "Mis rutas" y pulsar sobre el icono .

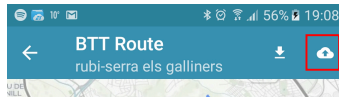


Figura 66. Manual de usuario - Opción compartir ruta.

10.3.8 Ampliar mapa

Cuando el usuario esté en la pantalla de detalle, tanto de una ruta propia como de una compartida por otro usuario, podrá acceder a un mapa ampliado donde podrá analizar el *track* de la ruta. Pulsando sobre el mapa estático accederá a la pantalla que muestra el mapa ampliado y manipulable.


Además, el icono  en la parte inferior derecha del mapa, le indicará que el mapa se puede ampliar.



Figura 67. Manual de usuario - Acceso a mapa ampliado.

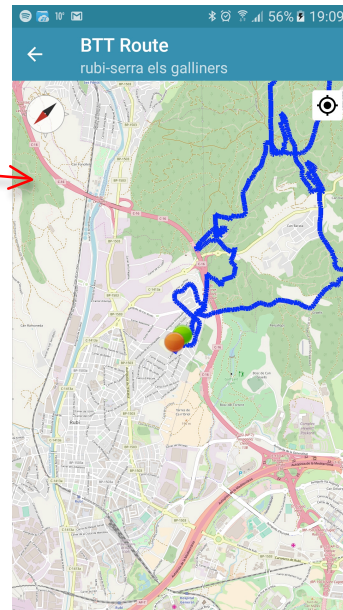



Figura 68. Manual de usuario - Mapa ampliado de ruta.

10.3.9 Filtrar rutas

Cuando el usuario accede a la pantalla de rutas compartidas por otros usuarios, se muestran por defecto las que se encuentran a una determinada posición desde la ubicación del usuario. Desde la pantalla de filtros se podrán obtener las rutas que cumplan con unos determinados requisitos. Para acceder a esta pantalla, pulsaremos sobre el icono  que tenemos disponible en la pantalla de “Buscar rutas”.

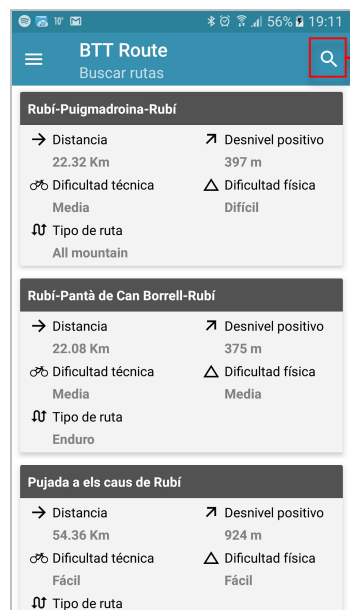


Figura 69. Manual de usuario-Opción de acceso a filtros.



Figura 70. Manual de usuario-Pantalla de filtros de búsqueda.