UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Ingeniería del Software

GoSykel: una aplicación móvil para promover el uso de la bicicleta en la población

GoSykel: a mobile application to boost bicycle's usage among citizens

Realizado por
María Gálvez Manceras

Tutorizado por
Gabriel Jesús Luque Polo

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, julio de 2022

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA DEL SOFTWARE

GoSykel: una aplicación móvil para promover el uso de la bicicleta en la población

GoSykel: a mobile application to boost bicycle's usage among citizens

Realizado por
María Gálvez Manceras

Tutorizado por
Gabriel Jesús Luque Polo

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JULIO DE 2022

Fecha defensa: julio de 2022

Dedicatoria

Hoy pongo fin a una etapa larga y bonita a la par que dura. Y es gracias a las personas de mi alrededor que el camino ha sido más llevadero. Gracias por escucharme, por aguantarme y por vivir conmigo esta etapa.

Me gustaría dedicar este trabajo a mis padres y a mi hermano, han sido mis principales puntos de apoyo durante este viaje, siempre han estado ahí, pase lo que pase, me han apoyado para no tirar la toalla y seguir.

A Noel, por ayudarme, escucharme y animarme cuando más lo necesitaba durante estos dos últimos años.

A Daniel Pérez, por habernos ayudado y compartir su conocimiento sin pedir nada a cambio durante estos 4 años, el mundo necesita más gente como él.

Mis amigos de la universidad, sois lo mejor que me llevo de esta etapa. Nuestras meriendas de 2.50€, las barbacoas, los cafés, las tardes de biblioteca y academia en época de exámenes o las quedadas en Discord durante el confinamiento para estudiar. Adri, Antonio, David G, David C, Javi y Jose, os llevo en mi corazón.

Por último, y no menos importante, al profesor que aceptó tutelarme y que ha hecho que este trabajo vea la luz. A Gabriel, eres y serás un ejemplo a seguir de pura dedicación y de pasión por el trabajo. Ojalá más profesores como tú.

Abstract

Nowadays, climate change is a worldwide threat. We usually go to work or school in our car or motorbike which negatively impacts us when we try to mitigate climate change's threat. To take care of our planet and breathe cleaner air, we need to change how we move to a more sustainable mobility. Using our bicycle in our daily movements is an excellent approach to help our planet reduce emissions at the same time as we exercise.

With this work, we want to develop a mobile application where users can create routes and make other users' routes using their bicycles. Our users can see the available bicycle lanes in Málaga's center and also add new bicycle lanes that have not been added yet. To boost bicycle usage and contribute to the environment GoSykel gives points to our users when they do different actions inside the application such as registering, creating new routes or adding new bicycle lanes.

Keywords: GoSykel, mobile application, sustainable mobility, routes, bicycle lanes

Resumen

En los tiempos que corren el cambio climático es una amenaza a nivel mundial. La forma en que nos transportamos, habitualmente mediante el uso de vehículos a motor como coches o motos, contribuye negativamente a su mitigación. Para cuidar del planeta donde vivimos y respirar un aire más limpio tenemos que cambiar la forma en que nos movemos hacia una movilidad más limpia. En este caso, hacer uso de la bicicleta en los desplazamientos es una buena forma de ayudar al planeta reduciendo las emisiones al mismo tiempo que hacemos deporte.

Mediante la realización de este trabajo pretendemos desarrollar una aplicación móvil que permita a los usuarios crear rutas y realizar las rutas creadas por otros usuarios mediante el uso de la bicicleta. Nuestros usuarios también pueden consultar los carriles bici disponibles en Málaga capital así como añadir nuevos tramos de carril bici. Además, GoSykel anima a usar la bicicleta y a contribuir al medio ambiente ya que los usuarios ganan puntos al realizar diferentes acciones dentro de la aplicación.

Palabras clave: GoSykel, aplicación móvil, movilidad sostenible, rutas, carriles bici

Índice

1. Introducción **11**

1.1. Motivación 11

1.2. Objetivos 12

1.3. Estructura del documento 12

2. Tecnologías y herramientas **15**

2.1. Tecnologías *backend* 15

2.1.1. Python 15

2.1.2. Flask 16

2.1.3. MongoDB 16

2.1.4. Firebase 17

2.1.5. Heroku 17

2.2. Tecnologías *frontend* 18

2.2.1. JavaScript 18

2.2.2. React Native 19

2.2.3. Expo 19

2.3. Bibliotecas de desarrollo 20

2.3.1. Bibliotecas usadas en el *backend* 20

2.3.2. Bibliotecas usadas en el *frontend* 21

2.4. Herramientas de desarrollo 23

2.4.1. Visual Studio Code 23

2.4.2. Postman 23

2.4.3. MoonModeler 24

2.4.4. MongoDBCompass 24

2.5. Herramientas de gestión 24

2.5.1. Git/GitHub 24

2.5.2. Cloudinary 25

2.5.3. Trello 25

2.5.4. Google Drive . . . 26
2.5.5. Telegram . . . 26
2.6. Herramientas de documentación . . . 26
2.6.1. Google Docs . . . 26
2.6.2. Overleaf . . . 27
2.7. Herramientas adicionales . . . 27
2.7.1. Google Forms . . . 27
2.7.2. Marvel . . . 28

3. Metodología **29**
3.1. Metodología ágil . . . 29
3.2. Gestión del proyecto . . . 30
3.3. Iteraciones realizadas . . . 31
3.3.1. Iteración 0 . . . 31
3.3.2. Iteraciones 1-4 . . . 32

4. Análisis **35**
4.1. Requisitos . . . 35
4.1.1. Requisitos funcionales . . . 35
4.1.2. Requisitos no funcionales . . . 38
4.2. Casos de uso . . . 38

5. Modelado y diseño **61**
5.1. Modelo de datos . . . 61
5.1.1. Colección *User* . . . 61
5.1.2. Colección *Item* . . . 63
5.1.3. Colección *Route* . . . 64
5.1.4. Colección *Bycicle lane* . . . 64
5.1.5. Colección *Token* . . . 65
5.2. Diagramas de navegación . . . 65
5.3. Proceso de diseño . . . 65
5.3.1. Línea de diseño . . . 68

5.3.2. Prototipado interfaz de usuario 70

6. Desarrollo e implementación **77**

6.1. Iteraciones realizadas 77

6.1.1. Iteración 0 77

6.1.2. Iteración 1 79

6.1.3. Iteración 2 81

6.1.4. Iteración 3 85

6.1.5. Iteración 4 88

6.2. Pruebas realizadas 92

6.2.1. Pruebas de la API REST usando Postman 93

6.2.2. Pruebas en dispositivos reales 93

6.2.3. *Test* de usabilidad 97

7. Conclusiones y trabajo futuro **101**

7.1. Conclusiones 101

7.2. Trabajo futuro 102

Referencias **105**

Apéndice A. Manual de usuario **109**

A.1. Introducción 109

A.2. ¿Cómo acceder a GoSykel? 109

A.3. Principales secciones de GoSykel 110

A.4. Sección: Carriles bici 112

A.5. Sección: Rutas 113

A.6. Sección: *Ranking* 117

A.7. Sección: Cuenta 117

A.7.1. Subsección: Mi perfil 118

A.7.2. Subsección: Mis rutas 118

A.7.3. Subsección: Tienda 120

A.7.4. Subsección: Sobre GoSykel 122

A.7.5. Subsección: Cerrar sesión 122

Apéndice B. Manual de despliegue del *backend* **123**

B.1. Introducción 123

B.2. Pre-requisitos 123

B.3. Instalación de dependencias 123

B.4. Configuración de Firebase 124

B.5. Uso 125

B.6. Despliegue en la nube 126

Apéndice C. Manual de instalación y ejecución de la aplicación móvil en un dispositivo físico **129**

C.1. Introducción 129

C.2. Pre-requisitos 129

C.3. Instalación de dependencias 130

C.4. Configuración de Firebase 130

C.5. Uso de la aplicación en local 131

1 Introducción

En este capítulo describiremos los motivos que nos llevaron a realizar este proyecto y los objetivos que hemos perseguido durante el desarrollo y la futura defensa. También describiremos la estructura de la memoria y resumiremos brevemente los entregables que podemos encontrar en los apéndices.

1.1. Motivación

En los últimos años hemos ido notando los efectos del cambio climático en el planeta tierra: períodos de sequía, tormentas más intensas, temperaturas más elevadas... Entre las diversas causas que provocan estos acontecimientos se encuentra el transporte y la forma en que las personas nos movemos cotidianamente. Desde 1970 las emisiones de gases de efecto invernadero han aumentado un 80 % debido a los vehículos que se mueven por carretera [22].

Para cuidar del planeta tierra y tener un aire de mejor calidad que nos permita continuar viviendo en él, tenemos que sumergirnos en nuevas formas de movilidad: en este caso, la movilidad sostenible, y la bicicleta es el medio de transporte sostenible por excelencia. Además, tiene numerosas ventajas como sus bajas emisiones, promueve la actividad física y contribuye a mantener la distancia de seguridad, necesaria en estos tiempos de pandemia [17].

Por tanto, para motivar a la población a que use la bicicleta con más frecuencia, una buena alternativa sería promover su uso mediante la gamificación de forma que los usuarios habituales de las bicicletas se sientan motivados para contribuir al medio ambiente [1]. Aquí entra en juego GoSykel para promover el uso de la bicicleta mediante una aplicación móvil. Esta aplicación permitirá interactuar con el usuario y recopilar información sobre sus recorridos.

1.2. Objetivos

Para poder motivar a los usuarios y hacerles usar su bicicleta desarrollamos GoSykel, una aplicación móvil donde nos planteamos diferentes objetivos a cubrir y que listaremos a continuación.

A alto nivel, los usuarios podrán crear su perfil y gestionarlo, permitiéndoles editar el perfil, borrar su cuenta o iniciar sesión entre otros. Podrán ganar puntos por añadir nuevas rutas usando su bicicleta, realizando rutas creadas por otros usuarios o añadiendo nuevos carriles bici.

En cuanto a la visualización de la información, sabemos que es importante que los usuarios circulen por entornos seguros. Por ello, podrán explorar los diferentes carriles bici disponibles en el conjunto de datos abiertos del Ayuntamiento de Málaga [5] y visualizar las rutas que han añadido otros usuarios. Además de visualizar todas las rutas añadidas, podrán filtrarlas atendiendo diferentes criterios entre los que destacamos el autor o (parte del) nombre de la ruta.

Además de contar con una aplicación móvil, dispondremos de un servidor donde procesar los datos facilitados por el usuario a través de la recogida de coordenadas para añadir nuevas rutas, carriles bici o realizar otras rutas. Para resolver este problema diseñaremos un algoritmo de filtrado de coordenadas que nos permitirá obtener estadísticas de la ruta almacenada. Algunas de las estadísticas son el tiempo que ha tardado, la distancia recorrida y el número de paradas que ha hecho el usuario durante el recorrido.

1.3. Estructura del documento

En esta memoria queremos plasmar las diferentes fases realizadas para completar con éxito el trabajo y para ello la hemos dividido en diferentes capítulos cuyo contenido resumiremos a continuación.

En el Capítulo 1 planteamos el por qué del proyecto, resaltando la motivación y objetivos del mismo, un resumen de lo que encontraremos a lo largo del documento y mencionamos los diferentes entregables que podemos encontrar en los apéndices. De este modo, el lector tiene una idea general del proyecto que planteamos.

Continuamos con el Capítulo 2 donde destacamos las tecnologías empleadas en el desa-

rrollo. Dichas tecnologías incluyen desde lenguajes de programación hasta herramientas de gestión.

Tras tener claras qué tecnologías vamos a utilizar, pasamos al Capítulo 3 donde mencionamos la metodología que hemos empleado, cómo hemos gestionado el proyecto y el trabajo que hemos realizado en las diferentes iteraciones hasta completar los objetivos iniciales.

Pasamos al Capítulo 4 donde listamos las funcionalidades que tiene el producto desarrollado y las características del mismo apoyándonos en los requisitos funcionales, no funcionales y los correspondientes casos de uso.

En el Capítulo 5 explicamos la definición de la base de datos y los diagramas de navegación. También mencionamos aquellos aspectos interesantes en términos de diseño como son las líneas de diseño que hemos seguido y los bocetos realizados.

Por su parte, el Capítulo 6 menciona el proceso de desarrollo que hemos seguido, dividiendo el trabajo en iteraciones y explicando los requisitos implementados en cada iteración así como las decisiones tomadas en cada caso. Adicionalmente explicaremos las pruebas realizadas a lo largo del proceso de desarrollo.

En el Capítulo 7 recogemos las conclusiones finales que obtenemos tras realizar un proyecto de esta envergadura y presentamos diferentes propuestas de mejora a realizar en el futuro.

Hemos incluido 3 apéndices:

1. **Manual de usuario:** está pensado para usuarios nuevos en la aplicación y su propósito es ayudarles a familiarizarse con las funcionalidades implementadas dando una serie de pasos a seguir.

2. **Manual de despliegue del *backend*:** proporciona los pasos a seguir para poner en funcionamiento el servidor tanto en local como en la nube.

3. **Manual de instalación y ejecución de la aplicación móvil en un dispositivo físico:** da los pasos a seguir para poder configurar y ejecutar la aplicación en un teléfono móvil real.

2

Tecnologías y herramientas

Este capítulo nos centramos en las tecnologías elegidas para el desarrollo de la aplicación y las diferentes herramientas utilizadas a lo largo del proyecto. Más tarde mostraremos un listado exhaustivo de las bibliotecas que hemos usado y para qué, con el objetivo de facilitar el desarrollo a otros desarrolladores que estén interesados en desarrollar elementos similares. De este modo, cuentan con una pequeña guía para saber por dónde empezar.

2.1. Tecnologías *backend*

En esta sección repasaremos las tecnologías usadas para implementar el servidor, la base de datos, la autenticación de usuarios en la aplicación y el despliegue del servidor.

2.1.1. Python

Python [36] es uno de los lenguajes de programación *open-source* más populares. Estamos ante un lenguaje interpretado de alto nivel y multiparadigma ya que soporta tanto la programación orientada a objetos, la programación imperativa y en menor medida la programación funcional.

Igualmente Python destaca por ser un lenguaje multiplataforma, siendo posible instalarlo tanto en Windows como en Linux o Mac OS. Una de las ventajas de Python es su sintaxis de muy alto nivel y la gran cantidad de bibliotecas disponibles que permiten solventar fácilmente diferentes problemas.

Python se ha convertido en un lenguaje muy popular a lo largo de los años, tanto es así que es muy utilizado en proyectos relacionados con la Inteligencia Artificial, representación de datos o desarrollo web entre otros. Además, ha creado una comunidad de

usuarios bastante grande que es muy útil cuando queremos encontrar información o solventar los problemas que nos puedan surgir.

Hemos empleado Python para desarrollar el *backend* de la aplicación debido a que contamos con experiencia previa en este lenguaje así como por la gran cantidad de recursos disponibles en Internet para buscar ayuda.

2.1.2. Flask

Flask [33] es un *microframework* que permite desarrollar rápidamente aplicaciones web en Python usando el Modelo Vista Controlador (MVC).

Con la instalación simple de Flask tenemos las herramientas necesarias para comenzar a desarrollar una web funcional. No obstante, si necesitamos una funcionalidad que no se puede desarrollar usando las herramientas predeterminadas se pueden instalar numerosas bibliotecas adicionales que nos ayudarán a implementar la funcionalidad.

Una de las principales ventajas de Flask es que incluye un servidor web de desarrollo por lo que es posible ver los resultados mientras estamos desarrollando.

Al igual que Python, hemos elegido este *framework* ya que es muy ligero y facilita el desarrollo, haciendo que éste sea rápido. Igualmente, ofrece todas las características necesarias para resolver las funcionalidades que planteamos. En nuestro caso, hemos empleado Flask para desarrollar la API REST de GoSykel.

2.1.3. MongoDB

MongoDB [21] es un sistema gestor de base de datos no relacional (NoSQL) que guarda los datos en estructuras BSON con un esquema que puede ir cambiando dinámicamente. A diferencia de las bases de datos relacionales, en MongoDB los datos se almacenan en colecciones y cada colección tiene diferentes documentos. Si bien es cierto que nos ofrece la posibilidad de almacenar los datos localmente o en la nube (usando MongoDB Atlas), para el desarrollo del proyecto hemos optado por usar la versión en la nube.

Se caracteriza por ser un sistema veloz donde se alcanza un buen balance entre el rendimiento y la funcionalidad gracias a la forma en que se realizan las consultas.

Una de las ventajas de MongoDB es que está escrito en C++ lo que nos permite instalarlo en Windows, Linux y Mac OS. Igualmente, al estar en constante evolución, con cada versión nos ofrecen nuevas mejoras. Desde el punto de vista del desarrollador, la forma de añadir nuevos documentos a una colección es más flexible que en otros sistemas gestores de bases de datos.

En nuestro sistema hemos usado MongoDB para almacenar los datos de los usuarios, las rutas, los carriles bici, etc. Si bien es cierto que existen otros sistemas gestores de bases de datos relacionales alojados en la nube, hemos elegido MongoDB porque cumple los requisitos que necesita nuestro sistema y se integra de forma muy sencilla con el resto de tecnologías que estamos utilizando. Especialmente, la biblioteca PyMongo (que veremos más adelante) facilita su uso desde el lenguaje Python, utilizado en el *backend*.

2.1.4. Firebase

Firebase [9] es una plataforma en la nube creada por Google para desarrollar rápidamente aplicaciones web y móvil de gran calidad en las diferentes plataformas: iOS, Android y web.

Desde el equipo de Google ponen a nuestra disposición una amplia gama de herramientas de fácil uso dentro de una misma plataforma. Sus herramientas están agrupadas en 4 categorías diferentes: desarrollo, crecimiento, monetización y análisis.

Firebase es de gran utilidad en proyectos que son pequeños y que no tendrán gran flujo de datos ya que podemos comenzar usando el plan gratuito sin incurrir en gastos adicionales. Otro aspecto interesante de Firebase es que nos permite centrarnos en otros aspectos más interesantes del desarrollo como puede ser el *frontend* dejando en segundo plano al *backend*.

Concretamente hemos usado Firebase para autenticar a los usuarios, delegando a este servicio aspectos como el almacenamiento de la contraseña de acceso del usuario y las operaciones relacionadas con la creación de las cuentas de los usuarios, el inicio de sesión de los mismos, etc. Nos hemos decantado por Firebase debido a su facilidad para conectarlo con el *frontend* y el *backend*.

2.1.5. Heroku

Heroku [37] es una plataforma alojada en la nube que nos permite compilar, desplegar y monitorizar aplicaciones rápidamente. Proporcionan toda la infraestructura necesaria y ofrece

soporte para aplicaciones escritas en los lenguajes más populares como Node, Ruby, Java o Scala entre otros.

Cuando desplegamos una aplicación en Heroku, ésta se aloja en un *dyno* (contenedor) cuyo sistema operativo subyacente es Linux (Ubuntu). Es posible añadir funcionalidades adicionales al contenedor mediante la instalación de extensiones.

Entre las ventajas de Heroku destacan: seguridad en las comunicaciones ya que los despliegues se hacen sobre HTTPS; escalabilidad, inicialmente la aplicación se aloja en un contenedor de la capa gratuita y en caso de necesidad los contenedores se pueden escalar vertical y horizontalmente; despliegues ágiles, es posible desplegar una aplicación rápidamente usando varios comandos.

A lo largo del desarrollo hemos usado Heroku para desplegar los nuevos cambios realizados en el *backend* usando Flask. Decidimos usar esta tecnología ya que es gratuita y de fácil configuración para hacer los primeros despliegues. Una vez la configuramos sólo hay que emplear un comando para desplegar los nuevos cambios.

2.2. Tecnologías *frontend*

En esta sección repasaremos las tecnologías y los *frameworks* usados para implementar la cara visible de GoSykel, la aplicación móvil.

2.2.1. JavaScript

JavaScript [16] es el lenguaje de programación que hemos elegido para el desarrollo de GoSykel. Diseñado por Mozilla, sigue el estándar ECMAScript y se caracteriza por ser un lenguaje ligero, interpretado y compilado en tiempo de ejecución.

JavaScript nos ofrece soporte para la programación orientada a objetos, imperativa y declarativa. Destaca por facilitar un desarrollo rápido ya que, al ser interpretado, se puede probar de forma inmediata sin ningún proceso intermedio de compilación y la simplicidad en su sintaxis ya que está inspirada principalmente en la de Java y en menor medida en la de C/C++ lo que facilita su aprendizaje.

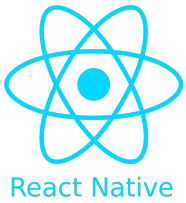
Con el paso de los años su uso se ha incrementado y con ello su popularidad, incrementan-

do así el soporte disponible en Internet para este lenguaje. Tiene una comunidad de usuarios muy activa en StackOverflow y en GitHub.

Hemos seleccionado JavaScript como lenguaje de programación para desarrollar el *frontend* de la aplicación ya que es uno de los lenguajes compatibles con Expo, plataforma que explicaremos más tarde. Utilizar JavaScript ha sido un reto puesto que contábamos con poca experiencia en este lenguaje. Si bien es cierto que existen otras alternativas como TypeScript, usar JavaScript nos ha servido para romper el hielo y coger experiencia de cara al futuro ya que estamos ante un lenguaje cada vez más demandado.

2.2.2. React Native

React Native [27] es un *framework* JavaScript desarrollado por Facebook que nos permite crear aplicaciones nativas para Android e iOS. Está basado en React y sigue el mismo paradigma a la hora de construir las aplicaciones.



Una de las ventajas principales de React Native frente a otros *frameworks* de desarrollo de aplicaciones móviles es su curva de aprendizaje, sencilla para desarrolladores sin experiencia. Al estar basado en JavaScript es sencillo de leer y aprender pues nos ofrece una gran variedad de elementos como mapas, filtros...

React Native destaca por tener bibliotecas compatibles tanto con Android como con iOS permitiendo crear aplicaciones que más tarde funcionarán en ambos sistemas operativos. Igualmente, nos permite desarrollar y ejecutar aplicaciones complejas de forma suave mejorando el rendimiento de las aplicaciones nativas.

El motivo por el que hemos seleccionado React Native como *framework* para desarrollar el *frontend* es por su curva de aprendizaje y fácil adaptación al mismo. Igualmente ha sido una buena elección ya que cuenta con un gran número de bibliotecas que nos facilitan las tareas de desarrollo.

2.2.3. Expo

Expo [7] es una plataforma de código abierto formada por herramientas, bibliotecas y servicios que nos permiten crear aplicaciones móviles nativas en iOS y Android usando JavaScript.

Internamente Expo usa Expo SDK que es una biblioteca de JavaScript que ofrece acceso nativo a la cámara del teléfono, a los contactos o almacenamiento local sin editar código a bajo nivel. Esto permite que podamos ejecutar y probar el proyecto en cualquier dispositivo que tenga la aplicación Expo Go instalada. También es posible visualizar los cambios en el dispositivo a medida que estamos desarrollando la aplicación.

Entre las ventajas de Expo destacan:

- Sencillez de uso, no hace falta tener instalado Xcode o Android Studio.
- Proporciona componentes propios para la interfaz de usuario que no están disponibles en React Native.
- Permite acceder fácilmente a servicios complejos pero necesarios en muchas aplicaciones móviles como las notificaciones *push* o el almacenamiento interno.

Nos hemos decantado por Expo para el desarrollo de la aplicación móvil ya que incluye todo lo necesario para comenzar a desarrollar rápidamente sin tener que realizar una configuración exhaustiva del proyecto ya que de eso se encarga Expo. Esto hace que sea la alternativa perfecta para desarrolladores sin experiencia en la configuración de proyectos en React Native. Igualmente hay que destacar que estamos ante una plataforma donde la visualización de los cambios en el código es bastante rápida sin necesidad de perder tiempo configurando nuestro dispositivo.

2.3. Bibliotecas de desarrollo

2.3.1. Bibliotecas usadas en el *backend*

De cara a ayudarnos con el desarrollo del servidor de la aplicación, hemos usado diferentes bibliotecas disponibles en Python. A continuación, listamos las más importantes y la funcionalidad que ha aportado su uso:

- Firebase Admin [10] es una biblioteca del lado del servidor que nos permite interactuar con Firebase con privilegios. Concretamente, hemos usado Firebase Admin para eliminar la cuenta del usuario en Firebase cuando desea eliminar su perfil en GoSykel. Hemos empleado esta tecnología en el *backend* ya que es la misma tecnología que más tarde empleamos en el *frontend*.

- GeoPy [34] es una biblioteca que nos permite acceder a servicios de geocodificación como OpenStreetMap, Google Geocoding API o MapBox. Concretamente, hemos usado GeoPy para calcular la distancia entre dos coordenadas. La facilidad de uso y la documentación disponible en Internet nos han ayudado a decantarnos por ella.

- PyJWT [35] es una biblioteca de Python que nos permite codificar y decodificar *JSON Web Tokens* (JWT). JWT es un estándar que permite difundir la identidad del usuario entre el cliente y el servidor de forma segura mediante el uso de información cifrada. Hemos empleado esta biblioteca para garantizar la seguridad en las comunicaciones, garantizando que el usuario es quien dice ser. Entre las razones por las que hemos usado esta biblioteca destacan: su popularidad y el gran soporte que encontramos en Internet.

- PyMongo [26] es una distribución de Python que contiene las herramientas necesarias para trabajar con MongoDB desde Python. Hemos usado esta biblioteca para operar con la base de datos insertando, editando, eliminando o filtrando documentos. Esta biblioteca nos proporciona las funcionalidades adecuadas para el manejo de la base de datos y no hemos encontrado otra biblioteca que nos permita interactuar con MongoDB.

2.3.2. Bibliotecas usadas en el *frontend*

De cara a ayudarnos con el desarrollo de la aplicación móvil hemos usado diferentes bibliotecas de React Native y Expo. A continuación, listamos las más importantes y la funcionalidad que ha aportado su uso:

- Expo Google Fonts [8] es una biblioteca de código abierto de Expo que nos permite incorporar diferentes fuentes disponibles en Google Fonts en una aplicación desarrollada con Expo. Esta biblioteca nos ha sido útil para incorporar diferentes fuentes en la aplicación y algunas de sus variantes. Esta biblioteca nos proporciona las funcionalidades adecuadas para manejar cómodamente las fuentes que usamos en la aplicación y no hemos encontrado otra biblioteca similar.

- Expo App Loading [3] se encarga de mostrar la pantalla de bienvenida mientras elementos como las fuentes, logos e imágenes están cargando. Concretamente hemos usado esta biblioteca para mostrar dicha pantalla en caso de que los elementos no estuvieran car-

gados. Al tratarse de una biblioteca de Expo es la opción que mejor se ajusta a nuestras necesidades.

- Expo Location [18] es un paquete que nos permite leer datos relacionados con la geolocalización del dispositivo. También se encarga de gestionar los permisos necesarios para poder leer los datos de geolocalización. En nuestro caso hemos usado esta biblioteca para conocer la ubicación en tiempo real del usuario cada cierto intervalo de tiempo. La documentación disponible en línea hace que esta biblioteca sea la candidata ideal para obtener la ubicación del usuario fácilmente.

- React Native Maps [29] proporciona los componentes necesarios para poder incluir mapas en las aplicaciones Android e iOS. Nos proporciona elementos como: marcadores, mapas, *polylines* (líneas para dibujar rutas)... Durante el desarrollo de GoSykel hemos usado esta biblioteca para mostrar la ruta a seguir por los usuarios y mostrar la ubicación actual del usuario en el mapa. A diferencia de otras bibliotecas disponibles para este tipo de proyectos, ésta destaca por su alta compatibilidad con los mapas de ambos sistemas operativos.

- React Native Paper [14] nos proporciona un conjunto de componentes listos para ser usados en aplicaciones desarrolladas con React Native siguiendo las guías de diseño de *Material Design*. Durante el desarrollo de la interfaz de usuario de GoSykel hemos usado esta biblioteca ya que ofrece una gran variedad de componentes altamente personalizables. Esta biblioteca nos proporciona los elementos necesarios para construir una interfaz de usuario agradable y no hemos encontrado otra biblioteca similar que nos ayude a conseguir dicho objetivo.

- React Navigation [28] nos permite gestionar la navegación y el enrutamiento dentro de aplicaciones desarrolladas con Expo y React Native. Hemos empleado esta biblioteca para gestionar la navegación entre las diferentes pantallas de la aplicación móvil. Si bien es cierto que existen diferentes alternativas a esta biblioteca, nos hemos centrado en ella ya que proporciona diferentes métodos de navegación.

- Firebase [9], en concreto el módulo *authentication* nos permite gestionar el registro e inicio de sesión de los usuarios de la aplicación así como se encarga de actualizar la

contraseña de acceso del usuario o su dirección de correo electrónico entre otros. Nos hemos ayudado de esta biblioteca para manejar la autenticación de los usuarios. Como hemos comentado anteriormente, hemos seleccionado esta biblioteca por su sencilla integración con el proyecto en JavaScript.

2.4. Herramientas de desarrollo

De cara a ayudarnos con el desarrollo de la aplicación móvil hemos usado diferentes herramientas. A continuación veremos cada una con detalles.

2.4.1. Visual Studio Code

Visual Studio Code [32] es el IDE que hemos elegido para desarrollar tanto la aplicación móvil como el servidor. Destaca por ser el IDE más empleado a nivel mundial y debido a su éxito cuenta con un gran número de herramientas y extensiones además de la gran cantidad de documentación disponible en línea.

Hemos usado este IDE debido a que ya teníamos experiencia previa en su uso y mediante la instalación de diferentes extensiones se convierte en el IDE perfecto para desarrollar código en Python o JavaScript. Igualmente se integra muy bien con git por lo que tenemos todo lo que necesitamos en un mismo lugar.

2.4.2. Postman

Postman [25] es una plataforma para construir y usar APIs. Trata de cumplir diferentes propósitos como probar APIs tanto para usarlas en *frontend* como en *backend* o generar documentación de la API que estamos desarrollando.

Durante el desarrollo hemos empleado Postman para comprobar que las peticiones al servidor funcionan cuando el servidor se ejecuta en local y cuando lo desplegamos en la nube. Haber usado Postman en trabajos anteriores ha hecho que se convierta en el candidato ideal para comprobar las peticiones rápidamente. Si bien es cierto que existen otras alternativas a Postman, creemos que esta plataforma es bastante completa y se ajusta bastante bien a las necesidades del proyecto.

2.4.3. MoonModeler

MoonModeler [6] es una herramienta cuya funcionalidad principal es dibujar diagramas Entidad Relación (ER) para diferentes tipos de bases de datos como MongoDB, Mongoose, PostgreSQL, MySQL, MariaDB, SQLite y GraphQL. Cabe destacar que es capaz de hacer ingeniería inversa a partir de una base de datos ya existente.

Al comienzo del proyecto esta herramienta nos sirvió para diseñar el modelo de datos que más tarde seguiríamos en la aplicación. Tras investigar otras herramientas para construir dichos diagramas observamos que esta es la más completa ya que permite hacer ingeniería inversa a partir de bases de datos ya existentes. No obstante, no es perfecta, para usar algunas características hay que pagar. Aun así, cubre las necesidades del proyecto.

2.4.4. MongoDBCompass

MongoDBCompass [20] es una herramienta interactiva para consultar los datos almacenados en una base de datos de MongoDB. Es fácil de usar pues la interfaz de usuario es sencilla, intuitiva y amigable.

A lo largo del desarrollo hemos empleado esta herramienta para comprobar los datos almacenados y detectar irregularidades o datos erróneos así como para filtrar datos. La facilidad para conectarnos con la base de datos usando esta herramienta hace que sea perfecta para visualizar los datos en un par de *clics*.

2.5. Herramientas de gestión

Un aspecto muy importante de los proyectos de desarrollo de software es la gestión de los mismos. De cara a llevar una buena gestión del proyecto, hemos empleado diferentes herramientas que veremos a continuación.

2.5.1. Git/GitHub

Git [11] es el sistema de control de versiones más utilizado en todo el mundo. Permite restaurar una versión anterior del proyecto en caso de ser necesario o crear nuevas ramas dentro de un mismo proyecto para no interferir al resto de desarrolladores.

Habitualmente se usa Git en combinación con GitHub [4] un portal que actúa como repositorio de código en línea. El propósito de esta plataforma es conectar a los desarrolladores pudiendo ver proyectos de otros usuarios así como aportar sus ideas a otros repositorios.

Durante el desarrollo se han empleado estas herramientas para gestionar el control de versiones de ambos proyectos y monitorizar el historial de *commits* realizados. Hemos elegido GitHub ya que su interfaz es intuitiva y fácil de usar. Una de sus puntos fuertes es que nos permite revisar el estado del repositorio rápidamente.

2.5.2. Cloudinary

Cloudinary [15] nos permite gestionar las imágenes y contenido multimedia que más tarde se mostrará en aplicaciones web y móviles. Es de gran utilidad cuando se desea cargar, almacenar, manipular, optimizar y entregar imágenes y vídeos de manera eficiente y efectiva.

En nuestro caso hemos usado Cloudinary para almacenar las imágenes correspondientes a los avatares, insignias y encabezados disponibles en GoSykel. Esta herramienta en línea es muy fácil de usar y añadir nuevos elementos a la colección es tan fácil como arrastrar y soltar en la carpeta donde deseamos guardarlos. Pese a tener funcionalidades de pago, con la capa gratuita es más que suficiente para cubrir los objetivos del proyecto.

2.5.3. Trello

Trello [31] es una herramienta muy útil para gestionar proyectos y organizar el trabajo mediante tableros, listas y tarjetas fomentando así el trabajo colaborativo entre los diferentes miembros del equipo. Además, cuenta con una gran variedad de extensiones que permiten tener toda la información importante en un mismo lugar.

Trello nos ha resultado muy útil para gestionar las tareas pendientes durante el ciclo de vida del proyecto y anotar aspectos importantes del mismo. Al igual que con otras herramientas, teníamos experiencia previa en su uso y la incorporación de la extensión de Google Drive fue la clave para aumentar la productividad. Respecto a otras herramientas del mercado esta es más completa ya que permite personalizar distintos aspectos de los tableros como las columnas o establecer una fecha límite a las tarjetas.

2.5.4. Google Drive

Google Drive [24] nos permite alojar contenido en diferentes formatos y acceder a él desde cualquier parte así como compartirlo con otros usuarios al mismo tiempo que fomenta el trabajo en equipo.

Desde el comienzo del proyecto hemos usado Google Drive para almacenar y compartir archivos importantes con el tutor. Si bien es cierto que existen otras herramientas para fomentar el trabajo colaborativo hemos elegido Google Drive porque no hace falta instalar programas adicionales, la versión web es muy completa.

2.5.5. Telegram

Telegram [30] es una aplicación de mensajería instantánea disponible en versión web y móvil que permite la comunicación entre personas de cualquier parte del mundo. Cuenta con diversas funcionalidades como crear y destruir conversaciones o incorporar *bots* a los grupos y conversaciones individuales.

Cuando empezamos el proyecto comenzamos a usar esta aplicación para comunicarnos de forma rápida y fluida y solventar problemas que pudieran surgir. Una de las características de Telegram que nos llevó a su uso es la privacidad de los usuarios, para registrarnos en Telegram no tenemos que aportar nuestro número de teléfono. Es suficiente con disponer de un nombre de usuario para que los demás puedan contactar con nosotros.

2.6. Herramientas de documentación

Ya que el proyecto ha sido relativamente largo (3 o 4 meses) es importante documentar cualquier decisión que hayamos tomado sobre el mismo para recuperar dicha información más tarde y tenerla en cuenta. Para llevar a cabo esta tarea nos hemos ayudado de diferentes herramientas.

2.6.1. Google Docs

Google Docs [12] es una herramienta de procesamiento de textos creada por Google que permite elaborar documentos de forma sencilla y rápida.

En este trabajo lo hemos utilizado como bloc de notas para anotar diferentes decisiones e información relevante que más tarde trasladaremos a este documento LaTeX. Al haber usado Google Drive para compartir diversos archivos entre nosotros, Google Docs es el candidato ideal para crear documentos fácilmente accesibles dentro de carpetas compartidas con otros usuarios.

2.6.2. Overleaf

Overleaf [23] es un editor web de LaTeX que permite generar documentos científicos, artículos o resultados de trabajos de investigación con gran calidad y que permite la colaboración entre diferentes usuarios.

Hemos empleado esta herramienta para desarrollar el contenido de la memoria del Trabajo Fin de Grado (TFG) ya que nos proporciona las herramientas necesarias para compilar código LaTeX sin necesidad de instalar LaTeX en local.

2.7. Herramientas adicionales

A lo largo de la realización de este trabajo hemos usado otras herramientas adicionales para facilitar algunas de las tareas. A continuación las repasaremos.

2.7.1. Google Forms

Google Forms [13] es una herramienta de Google que nos permite realizar y compartir cuestionarios en línea. Igualmente nos permite procesar los resultados y visualizarlos a través de diferentes gráficas.

Concretamente hemos empleado esta herramienta para obtener los resultados del *test* de usabilidad realizado a diferentes usuarios. Es la herramienta de cuestionarios ideal para integrar dentro del sistema de archivos compartidos de Google Drive que hemos comentado anteriormente, así disponemos de los resultados del *test* en el mismo sitio que el resto de archivos.

2.7.2. Marvel

Marvel [19] es una herramienta que nos permite realizar prototipos de diferentes tipos de aplicaciones rápidamente. Pone a disposición de los usuarios múltiples elementos típicos de páginas web, aplicaciones móviles Android o iOS, entre otros.

En nuestro caso particular hemos usado Marvel para realizar un esbozo inicial de la interfaz de GoSykel y plantear una primera versión de la navegación entre los diferentes elementos de la aplicación. Si bien es cierto que hay diferentes herramientas que nos permiten realizar bocetos en Internet, hemos escogido esta ya que sus elementos son prácticamente iguales a los elementos que nos proporcionan los *frameworks* de desarrollo, por lo que el boceto es casi una aplicación real.

3 Metodología

La metodología define cómo desarrollaremos el proyecto y sienta las bases de actuación sobre el mismo. En este capítulo explicamos las decisiones tomadas en cuanto a la metodología que hemos seguido y detallaremos aspectos importantes sobre ella.

3.1. Metodología ágil

Al inicio del proyecto planteamos usar una metodología ágil donde poder adaptarnos a los cambios que aparezcan durante el desarrollo. En tecnologías tan recientes como Flask, Firebase, React Native o Expo son habituales los cambios de versiones provocando que la forma de proceder cambie de un momento para otro. Esto tiene sus consecuencias dentro del proyecto pues tenemos que adaptar los requisitos a la nueva metodología establecida por estas tecnologías.

Hemos elegido una metodología iterativa basada en Scrum [2] y adaptada a un proyecto conformado por una única persona que cubre todos los roles (analista, diseñador, desarrollador, probador y documentalista). No obstante, el rol de *Product Owner* lo ha ocupado el tutor.

Una de las características de Scrum es que el proyecto está conformado por diferentes iteraciones cuya duración habitual es de 15 días. En nuestro caso, hemos realizado algunas iteraciones algo más largas así como diferentes reuniones periódicas tras acabar la iteración. En el transcurso de las iteraciones se han realizado consultas y reuniones para resolver dudas y plantear diferentes soluciones.

Cabe destacar que ambos (el tutor y el estudiante) hemos participado en todo el proceso verificando los resultados obtenidos en las diferentes iteraciones durante las reuniones programadas con el tutor donde mostramos el progreso hasta la fecha y definimos los siguientes pasos a seguir.

3.2. Gestión del proyecto

De cara a gestionar y organizar el trabajo pendiente por hacer, el trabajo que está en progreso y el trabajo concluido hemos usado Trello como hemos comentado en el capítulo anterior.

Concretamente ha sido un tablero compartido entre el tutor y el estudiante. Dicho tablero tiene diferentes columnas como podemos ver en la Figura 1.

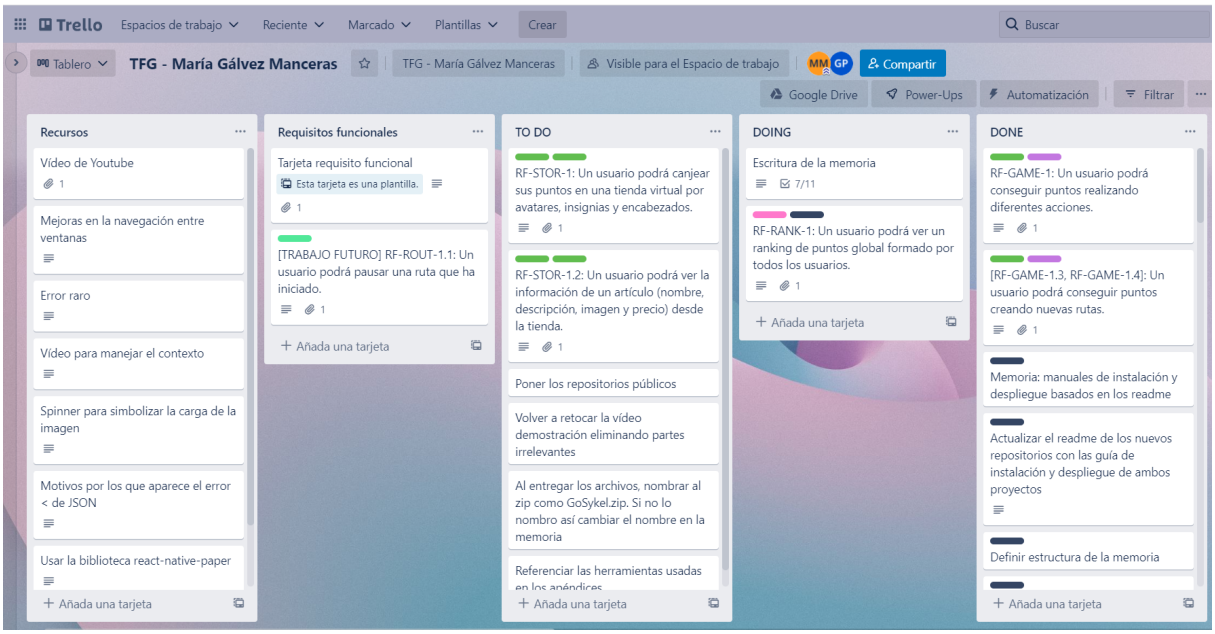


Figura 1: Tablero de Trello donde hemos organizado el proyecto

Las columnas más importantes del tablero son *TO DO, DOING* y *DONE* pues son las que marcan el trabajo que hay que completar todavía, el trabajo que está en progreso y el trabajo completado respectivamente. Adicionalmente hemos añadido las columnas "Recursos" y "Requisitos funcionales". La primera de ellas son pequeñas anotaciones que hemos ido realizando durante el desarrollo y contiene principalmente tarjetas con información para resolver diferentes problemas. Por su parte, la columna "Requisitos funcionales" simboliza la pila de funcionalidades pendientes que aún no han sido planificadas.

Inicialmente planteamos cada requisito funcional como una tarjeta de la columna "Requisitos funcionales". Aprovechamos la potencia de Trello y creamos una tarjeta que serviría como plantilla para el resto de requisitos funcionales y que está conectada (gracias a una extensión de Trello) a la carpeta compartida de Google Drive haciendo que sea accesible con un par de *clics* desde cualquier tarjeta.

Otro aspecto interesante de la gestión del proyecto es que mientras que hemos completado las funcionalidades hemos anotado aquellos aspectos relevantes de las mismas aprovechando la funcionalidad que nos ofrece Trello de añadir descripciones a las tarjetas.

3.3. Iteraciones realizadas

El proyecto lo hemos realizado siguiendo los procedimientos establecidos en el marco de trabajo de la Ingeniería del Software. Uno de los elementos de este marco de trabajo es la división del proyecto en diferentes iteraciones (que se corresponden a los *sprints* de Scrum) cuyo contenido detallamos en las siguientes líneas.

3.3.1. Iteración 0

Esta iteración inicial está formada por varias etapas imprescindibles que nos han servido para adquirir los conocimientos necesarios en aquellas tecnologías desconocidas (Firebase, React Native y Expo) hasta el momento así como para planificar y desarrollar las diferentes funcionalidades a implementar durante las iteraciones posteriores. En concreto las etapas que conformaron esta iteración son:

- Estudio y adaptación a las nuevas tecnologías mediante la elaboración de una pequeña aplicación de prueba.
- Obtención, clasificación y validación de requisitos así como su división en diferentes iteraciones.
- Elaboración de la planificación que más tarde seguiremos, puesta a punto del tablero de Trello donde agrupamos los requisitos que implementaremos en cada iteración y configuración de los proyectos *backend* y *frontend*.
- Elaboración de bocetos para determinar el diseño preliminar de la interfaz de usuario.
- Diseño de la base de datos que más tarde empleamos.

Como resultado de esta fase obtuvimos la planificación mostrada en la Figura 2. Contar con una planificación antes de empezar a ejecutar el proyecto nos ha ayudado a llevar un orden y cumplir estrictamente la planificación inicial si bien es cierto que se han dado algunos cambios puntuales donde movimos algunos requisitos entre iteraciones.

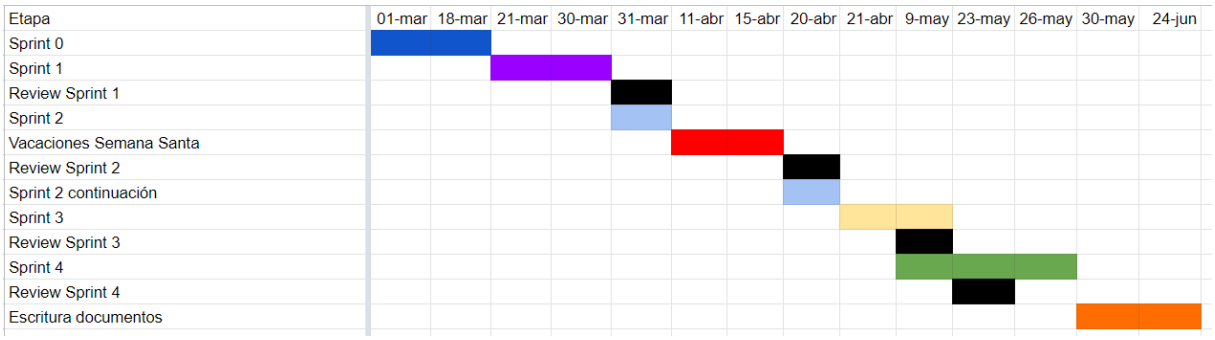


Figura 2: Resultado de la iteración 0

3.3.2. Iteraciones 1-4

Tras finalizar la Iteración 0 y con la planificación establecida damos comienzo al desarrollo de la aplicación dividido en 4 iteraciones que empiezan y terminan una tras otra. Al final de cada iteración nos reunimos para mostrar el trabajo realizado. El objetivo es implementar toda la funcionalidad acordada a lo largo de las 4 iteraciones propuestas.

En la **Iteración 1** cuya duración fue de 8 días naturales implementamos la visualización de los carriles bici, el registro y autenticación de usuarios así como el cambio de contraseña y cierre de sesión en la aplicación.

A lo largo de los 13 días naturales que duró la **Iteración 2** completamos la funcionalidad relativa a la gestión de usuarios permitiendo editar, eliminar y visualizar el perfil del usuario dentro de la aplicación. Al final de esta iteración nuestros usuarios ya podían añadir y descartar nuevos tramos de carril bici así como iniciar, descartar y guardar nuevas rutas.

Durante la **Iteración 3** que nos ocupó 16 días naturales implementamos la visualización del perfil de otros usuarios, distintos aspectos de la visualización de las rutas creadas por los usuarios, la realización de una ruta creada por otro usuario y la edición/eliminación de rutas que son propiedad de un usuario.

Además cubrimos algunos de los requisitos relacionados con la gamificación dentro de la aplicación, concretamente aquellos relacionados con el *ranking* de usuarios. Durante esta iteración realizamos diferentes mejoras, por ejemplo, añadimos encabezados a aquellas pantallas que no disponían de ellos, añadimos nuevos componentes que reutilizamos en diferentes pantallas y una pantalla de carga que mostramos cuando extraemos la información del servidor.

Finalmente, a lo largo de la **Iteración 4** que nos ocupó 15 días naturales completamos

las funcionalidades relativas a la gamificación y obtención de puntos por realizar diferentes acciones dentro de la aplicación. También pusimos a disposición de los usuarios una tienda donde canjear puntos y poder visualizar los elementos disponibles.

Al igual que en la iteración anterior realizamos diferentes mejoras en términos de seguridad implementando un mecanismo de *tokens* para verificar que el usuario es quien dice ser y generamos ficheros con datos de prueba que más tarde utilizamos en la demostración de la aplicación. Al final de esta iteración damos los últimos retoques y grabamos el vídeo demostración.

4
Análisis

En este capítulo nos centraremos en los requisitos que serán la base del diseño y desarrollo que realizaremos posteriormente así como el análisis de cada uno de ellos mediante la elaboración de los casos de uso.

4.1. Requisitos

A la hora de analizar los requisitos principalmente tratamos de encontrar respuesta a la pregunta, ¿qué debe hacer nuestra aplicación? para obtener un conjunto de funcionalidades que nos permitirán alcanzar los objetivos establecidos al inicio del proyecto.

Así pues tendremos dos conjuntos bien diferenciados de requisitos. Por un lado, requisitos funcionales que son funcionalidades que debe ofrecer la aplicación y por otro lado tenemos requisitos no funcionales que definen el comportamiento del sistema y las características del mismo.

4.1.1. Requisitos funcionales

En este apartado nos centramos en los requisitos funcionales de la aplicación y los clasificamos según el módulo al que pertenecen.

Requisitos relacionados con la gestión de usuarios

- RF-USER-1: Un usuario que no estuviera registrado anteriormente podrá registrarse facilitando un *email*, *nickname* y contraseña de 6 caracteres como mínimo.
- RF-USER-2: Un usuario podrá gestionar su perfil.
- RF-USER-2.1: Un usuario podrá iniciar sesión usando su *email* y contraseña.
- RF-USER-2.2: Un usuario podrá cerrar sesión en la aplicación.

- RF-USER-2.3: Un usuario podrá ver la información de su perfil.
- RF-USER-2.4: Un usuario podrá actualizar la información de su perfil.
- RF-USER-2.5: Un usuario podrá cambiar su contraseña antes de iniciar sesión en caso de que la haya olvidado.
- RF-USER-2.6: Un usuario podrá eliminar su perfil.
- RF-USER-3: Un usuario podrá ver el perfil de otro usuario.

Requisitos relacionados con la gestión de las rutas

- RF-ROUT-1: Un usuario podrá iniciar una nueva ruta.
- RF-ROUT-1.1: Un usuario podrá descartar una ruta que ha iniciado.
- RF-ROUT-1.2: Un usuario podrá guardar una nueva ruta asignándole nombre, visibilidad (pública o privada) y nivel de dificultad (de 1 a 5).
- RF-ROUT-2: Un usuario podrá realizar una ruta creada por otro usuario.
- RF-ROUT-3: Un usuario podrá editar el nombre o la visibilidad de una de sus rutas tras crearla.
- RF-ROUT-4: Un usuario podrá eliminar sus rutas creadas y/o completadas.

Requisitos relacionados con la visualización de las rutas

- RF-VSRT-1: Un usuario podrá ver todas las rutas registradas hasta la fecha, ordenadas de más reciente a más antigua.
- RF-VSRT-2: Un usuario podrá filtrar las rutas registradas según diferentes criterios: creador, (parte del) nombre, visibilidad, rutas más frecuentes y rutas creadas por mí.
- RF-VSRT-3: Un usuario podrá ver las rutas públicas creadas por otro usuario.
- RF-VSRT-4: Un usuario podrá ver información sobre la ruta seleccionada.
- RF-VSRT-4.1: Un usuario podrá ver el recorrido a seguir para completar una ruta.

- RF-VSRT-4.2: Un usuario podrá ver los usuarios más rápidos en realizar una ruta.
- RF-VSRT-5: Un usuario podrá ver desde su perfil las rutas que ha creado y las que ha completado.

Requisitos relacionados con los carriles bici

- RF-BICI-1: Un usuario podrá ver los carriles bici disponibles.
- RF-BICI-2: Un usuario podrá añadir un nuevo tramo de carril bici.
- RF-BICI-3: Un usuario podrá descartar un nuevo tramo de carril bici.

Requisitos relacionados con la obtención de puntos

- RF-GAME-1: Un usuario podrá conseguir puntos por registrarse.
- RF-GAME-2: Un usuario podrá conseguir puntos por iniciar sesión diariamente.
- RF-GAME-3: Un usuario podrá conseguir puntos creando nuevas rutas.
- RF-GAME-4: Un usuario podrá conseguir puntos completando una ruta creada por otro usuario.
- RF-GAME-5: Un usuario podrá conseguir puntos realizando una ruta de otro usuario frecuentemente.
- RF-GAME-6: Un usuario podrá conseguir puntos registrando carriles bici.

Requisitos relacionados con el *ranking* de usuarios

- RF-RANK-1: Un usuario podrá ver un *ranking* de puntos global formado por todos los usuarios.
- RF-RANK-2: Un usuario podrá ver su posición en el *ranking* de puntos global.
- RF-RANK-3: Un usuario podrá ver la posición de otro usuario en el *ranking*.

Requisitos relacionados con la tienda de avatares, insignias y encabezados

- RF-STOR-1: Un usuario podrá ver los avatares, insignias y encabezados disponibles.

- RF-STOR-2: Un usuario podrá ver la información de un artículo (nombre, descripción, imagen y precio) desde la tienda.
- RF-STOR-3: Un usuario podrá comprar un avatar, insignia o encabezado con los puntos conseguidos hasta el momento.

4.1.2. Requisitos no funcionales

En este apartado nos centramos en los requisitos no funcionales de la aplicación donde definimos el comportamiento del sistema y las características del mismo.

- RNF-1: La aplicación móvil funcionará adecuadamente en dispositivos que dispongan de una conexión GPS activa y una localización válida.
- RNF-2: La aplicación móvil funcionará correctamente en dispositivos que dispongan de conexión a Internet para visualizar el recorrido realizado o que se está realizando sobre un mapa.
- RNF-3: Las pruebas realizadas sobre la aplicación móvil se llevarán a cabo en un dispositivo real.
- RNF-4: El código desarrollado será modular, permitiendo reutilizar componentes y funciones, evitando así la duplicación de código.
- RNF-5: El registro, inicio y cierre de sesión así como el cambio de contraseña se implementarán usando *Firebase Authentication*.
- RNF-6: Los datos se almacenarán en una base de datos no relacional como MongoDB.
- RNF-7: El servidor (*backend*) se desarrollará en Python usando Flask.
- RNF-8: La aplicación móvil (*frontend*) se desarrollará usando React Native y Expo.

4.2. Casos de uso

En las siguientes tablas detallamos los casos de uso, teniendo en cuenta tanto el escenario principal como el alternativo. Cada caso de uso tiene un título para saber la acción que estamos realizando, una descripción para conocer más detalles, el/los requisito/s que cubre dicho caso

de uso, las pre y post condiciones y la secuencia de pasos tanto en el escenario principal como en el alternativo.

Tabla 1: CU01 - Registro de un usuario.

Título	Registro de un usuario.
Descripción	Un usuario que no estuviera registrado anteriormente podrá registrarse facilitando un *email*, *nickname* y contraseña de 6 caracteres como mínimo.
Requisitos	RF-USER-1
Pre-condición	El usuario ha iniciado GoSykel.
Post-condición	El usuario se ha registrado correctamente.
Escenario principal	
1. El usuario pulsa "¿Aún no tienes cuenta? Regístrate". 2. El sistema muestra el formulario de creación de una nueva cuenta. 3. El usuario rellena el campo nombre de usuario. 4. El sistema muestra el campo nombre de usuario con la información escrita por el usuario. 5. El usuario rellena el campo correo electrónico con una dirección de correo electrónico válida. 6. El sistema muestra el correo electrónico con la dirección introducida por el usuario. 7. El usuario rellena el campo contraseña introduciendo una contraseña válida, es decir, que tiene más de 6 caracteres. 8. El sistema muestra la contraseña con caracteres ocultos (*). 9. El usuario rellena el campo confirma tu contraseña introduciendo una contraseña válida, es decir, que tiene más de 6 caracteres y coincide con la contraseña introducida anteriormente. 10. El sistema muestra la contraseña repetida con caracteres ocultos (*). 11. El usuario pulsa el botón "¡A pedalear!". 12. El sistema comprueba la validez de los datos introducidos. 13. El sistema crea una cuenta al usuario y lo redirige a la pantalla principal donde se muestran los carriles bici.	
Escenario alternativo	
Correo no válido: 12. El usuario rellena el campo correo electrónico con una dirección de correo electrónico inválida.	

13. El sistema indica al usuario que la dirección de correo electrónico no es válida.

14. El usuario continúa con el paso 5 del escenario principal.

Correo electrónico en uso:

12. El usuario rellena el campo correo electrónico con una dirección de correo electrónico ya registrada.

13. El sistema indica al usuario que la dirección de correo electrónico está en uso.

14. El usuario continúa con el paso 5 del escenario principal.

Contraseña no válida:

12. El usuario rellena el campo contraseña introduciendo una contraseña que tiene menos de 6 caracteres.

13. El sistema indica al usuario que la contraseña no cumple los requisitos.

14. El usuario continúa con el paso 7 del escenario principal.

Contraseñas no coinciden:

12. El usuario rellena el campo confirma tu contraseña introduciendo una contraseña que no coincide con la contraseña introducida en el paso 7.

13. El sistema indica al usuario que las contraseñas no coinciden.

14. El usuario continúa con el paso 9 del escenario principal.

Campos sin rellenar:

12. El usuario no cumplimenta todos los campos del formulario.

13. El sistema indica al usuario que debe rellenar todos los campos.

14. El usuario continúa con el paso 2 del escenario principal.

Tabla 2: CU02 - Inicio de sesión.

Título	Inicio de sesión.
Descripción	Un usuario podrá iniciar sesión usando su *email* y contraseña.
Requisitos	RF-USER-2, RF-USER-2.1
Pre-condición	El usuario ha iniciado GoSykel. El usuario está registrado en GoSykel.
Post-condición	El usuario inicia sesión correctamente.
Escenario principal	
1. El usuario pulsa “Iniciar sesión”.	

2. El sistema muestra el formulario de inicio de sesión.

3. El usuario rellena el campo correo electrónico con una dirección de correo electrónico válida y ya registrada.

4. El sistema muestra el correo electrónico con la dirección introducida por el usuario.

5. El usuario rellena el campo contraseña introduciendo una contraseña de acceso válida y ya registrada.

6. El sistema muestra la contraseña con caracteres ocultos (*).

7. El usuario pulsa el botón “Iniciar sesión”.

8. El sistema comprueba la validez de los datos introducidos.

9. El sistema inicia la sesión del usuario y lo redirige a la pantalla principal donde se muestran los carriles bici.

Escenario alternativo

Correo no válido:

8. El usuario rellena el campo correo electrónico con una dirección de correo electrónico inválida.

9. El sistema indica al usuario que la dirección de correo electrónico no es válida.

10. El usuario continúa con el paso 3 del escenario principal.

Correo electrónico no registrado:

8. El usuario rellena el campo correo electrónico con una dirección de correo electrónico no registrada.

9. El sistema indica al usuario que la dirección de correo electrónico no está registrada.

10. El usuario continúa con el paso 3 del escenario principal.

Contraseña no válida:

8. El usuario rellena el campo contraseña introduciendo una contraseña que tiene menos de 6 caracteres.

9. El sistema indica al usuario que la contraseña no cumple los requisitos.

10. El usuario continúa con el paso 5 del escenario principal.

Contraseña incorrecta:

8. El usuario rellena el campo contraseña introduciendo una contraseña que no coincide con la contraseña de la cuenta del usuario.

9. El sistema indica al usuario que la contraseña es incorrecta.

10. El usuario continúa con el paso 5 del escenario principal. **Campos sin rellenar:** 8. El usuario no cumplimenta todos los campos del formulario. 9. El sistema indica al usuario que debe rellenar todos los campos. 10. El usuario continúa con el paso 2 del escenario principal.

Tabla 3: CU03 - Cierre de sesión.

Título	Cierre de sesión.
Descripción	Un usuario podrá cerrar sesión en la aplicación.
Requisitos	RF-USER-2, RF-USER-2.2
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección “Carriles bici”.
Post-condición	El usuario cierra sesión correctamente.
Escenario principal	
1. El usuario pulsa “Cuenta”. 2. El sistema muestra la sección de gestión de la cuenta. 3. El usuario pulsa “Cerrar sesión”. 4. El sistema cierra la sesión del usuario y lo redirige a la pantalla de bienvenida.	
Escenario alternativo	
Cierre de sesión incorrecto: 4. El sistema falla al cerrar la sesión y lo indica al usuario. 5. El sistema redirige al usuario a la pantalla de bienvenida.	

Tabla 4: CU04 - Ver la información del perfil.

Título	Ver la información del perfil.
Descripción	Un usuario podrá ver la información de su perfil.
Requisitos	RF-USER-2, RF-USER-2.3
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección “Carriles bici”.

Post-condición	El usuario visualiza la información de su perfil.
Escenario principal	
1. El usuario pulsa “Cuenta”.	
2. El sistema muestra la sección de gestión de la cuenta.	
3. El usuario pulsa “Mi perfil”.	
4. El sistema muestra la información del perfil del usuario.	
Escenario alternativo	
Error al cargar los datos del perfil:	
4. El sistema falla al cargar los datos del perfil y lo indica al usuario.	
5. El sistema muestra datos por defecto.	

Tabla 5: CU05 - Actualizar la información del perfil.

Título	Actualizar la información del perfil.
Descripción	Un usuario podrá actualizar la información de su perfil.
Requisitos	RF-USER-2, RF-USER-2.4
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección “Carriles bici”.
Post-condición	El usuario edita la información de su perfil correctamente.
Escenario principal	
1. El usuario pulsa “Cuenta”.	
2. El sistema muestra la sección de gestión de la cuenta.	
3. El usuario pulsa “Mi perfil”.	
4. El sistema muestra la información del perfil del usuario.	
5. El usuario pulsa “Editar perfil”.	
6. El sistema muestra la pantalla de edición del perfil.	
7. El usuario cambia su nombre de usuario (si procede).	
8. El sistema muestra el nuevo nombre de usuario.	
9. El usuario elige su avatar.	
10. El sistema muestra el avatar elegido.	

11. El usuario elige su encabezado. 12. El sistema muestra el encabezado elegido. 13. El usuario rellena el campo contraseña actual. 14. El sistema muestra la contraseña actual con caracteres ocultos (*). 15. El usuario rellena el campo nueva contraseña (si procede). 16. El sistema muestra la nueva contraseña con caracteres ocultos (*) (si procede). 17. El usuario pulsa “Guardar cambios”. 18. El sistema comprueba la validez de los datos introducidos. 19. El sistema actualiza el perfil del usuario y lo redirige a la información de su perfil.
Escenario alternativo
Avatar no seleccionado: 18. El usuario no ha seleccionado un avatar. 19. El sistema indica al usuario que hay campos sin rellenar. 20. El usuario continúa con el paso 9 del escenario principal. **Encabezado no seleccionado:** 18. El usuario no ha seleccionado un encabezado. 19. El sistema indica al usuario que hay campos sin rellenar. 20. El usuario continúa con el paso 11 del escenario principal. **Contraseña actual no introducida:** 18. El usuario no ha rellenado el campo contraseña actual. 19. El sistema indica al usuario que hay campos sin rellenar. 20. El usuario continúa con el paso 13 del escenario principal.

Tabla 6: CU06 - Cambio de contraseña antes de iniciar sesión.

Título	Cambio de contraseña antes de iniciar sesión.
Descripción	Un usuario podrá cambiar su contraseña antes de iniciar sesión en caso de que la haya olvidado.
Requisitos	RF-USER-2, RF-USER-2.5
Pre-condición	El usuario ha iniciado GoSykel. El usuario está registrado en GoSykel.
Post-condición	El usuario cambia su contraseña correctamente.

Escenario principal
1. El usuario pulsa “Iniciar sesión”. 2. El sistema muestra el formulario de inicio de sesión. 3. El usuario pulsa “¿Has olvidado tu contraseña?”. 4. El sistema muestra el formulario de cambiar contraseña. 5. El usuario rellena el campo correo electrónico. 6. El sistema muestra el correo electrónico introducido por el usuario. 7. El usuario pulsa el botón “Cambiar contraseña”. 8. El sistema comprueba la validez de los datos introducidos. 9. El sistema envía un correo electrónico con los pasos a seguir y lo redirige al formulario de inicio de sesión.
Escenario alternativo
Correo no válido: 8. El usuario rellena el campo correo electrónico con una dirección de correo electrónico inválida. 9. El sistema indica al usuario que la dirección de correo electrónico no es válida. 10. El usuario continúa con el paso 5 del escenario principal. **Correo electrónico no registrado:** 8. El usuario rellena el campo correo electrónico con una dirección de correo electrónico no registrada. 9. El sistema indica al usuario que la dirección de correo electrónico no está registrada. 10. El usuario continúa con el paso 5 del escenario principal. **Campos sin rellenar:** 8. El usuario no rellena el campo correo electrónico. 9. El sistema indica al usuario que debe rellenar el campo correo electrónico. 10. El usuario continúa con el paso 4 del escenario principal.

Tabla 7: CU07 - Eliminar el perfil.

Título	Eliminar el perfil.

Descripción	Un usuario podrá eliminar su perfil.
Requisitos	RF-USER-2, RF-USER-2.6
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección “Carriles bici”.
Post-condición	El usuario elimina su perfil correctamente.
Escenario principal	
1. El usuario pulsa “Cuenta”. 2. El sistema muestra la sección de gestión de la cuenta. 3. El usuario pulsa “Mi perfil”. 4. El sistema muestra la información del perfil del usuario. 5. El usuario pulsa “Editar perfil”. 6. El sistema muestra la pantalla de edición del perfil. 7. El usuario pulsa “Eliminar perfil”. 8. El sistema muestra un mensaje de confirmación. 9. El usuario selecciona “Sí”. 10. El sistema elimina el perfil del usuario y lo redirige a la pantalla de bienvenida.	
Escenario alternativo	
No elimina el perfil: 9. El usuario selecciona “No”. 10. El sistema oculta el mensaje de confirmación. 11. El usuario continúa con el paso 6 del escenario principal.	

Tabla 8: CU08 - Ver el perfil de otro usuario.

Título	Ver el perfil de otro usuario.
Descripción	Un usuario podrá ver el perfil y las rutas públicas creadas por otro usuario.
Requisitos	RF-USER-3, RF-VSRT-3
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección “Carriles bici”. GoSykel dispone de al menos 1 ruta.
Post-condición	El usuario ve el perfil y las rutas públicas de otro usuario correctamente.

Escenario principal
1. El usuario pulsa “Rutas”.
2. El sistema muestra la lista de rutas registradas.
3. El usuario pulsa “Ver ruta”.
4. El sistema muestra los detalles de la ruta seleccionada.
5. El usuario pulsa “Ver perfil” de uno de los usuarios más rápidos.
6. El sistema muestra la información del perfil del usuario y sus rutas públicas.

Tabla 9: CU09 - Añadir una nueva ruta.

Título	Añadir una nueva ruta.
Descripción	Un usuario podrá iniciar, guardar o descartar una nueva ruta.
Requisitos	RF-ROUT-1, RF-ROUT-1.1, RF-ROUT-1.2
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección “Carriles bici”. El usuario ha permitido a la aplicación tomar datos de su geoposición.
Post-condición	El usuario añade la ruta correctamente.
Escenario principal	
1. El usuario pulsa “Rutas”.	
2. El sistema muestra la lista de rutas registradas.	
3. El usuario pulsa “+ NUEVA RUTA”.	
4. El sistema muestra un mapa con la ubicación actual del usuario.	
5. El sistema recoge y almacena datos de la ubicación en tiempo real del usuario periódicamente.	
6. El usuario pulsa sobre “Guardar” cuando considera que la ruta ha acabado.	
7. El sistema muestra el formulario para guardar la ruta.	
8. El usuario rellena el campo nombre de la ruta.	
9. El sistema muestra el campo nombre de la ruta con la información del usuario.	
10. El usuario selecciona la visibilidad de la ruta.	
11. El sistema muestra el campo visibilidad de la ruta con la opción seleccionada por el usuario.	

12. El usuario indica el nivel de dificultad de la ruta.

13. El sistema muestra el nivel de dificultad de la ruta con la opción indicada por el usuario.

14. El usuario pulsa “Guardar”.

15. El sistema comprueba la validez de los datos introducidos.

16. El sistema almacena la nueva ruta y redirige al usuario a la pantalla de rutas.

Escenario alternativo

Descartar la ruta (en cualquier momento entre los pasos 4 y 5):

* El usuario pulsa sobre el icono situado en la parte superior izquierda para descartar la ruta.

* El sistema muestra un mensaje de confirmación.

* El usuario selecciona “Sí”.

* El sistema detiene la toma de datos y redirige al usuario a la pantalla de rutas.

Continuar con la ruta (en cualquier momento entre los pasos 4 y 5):

* El usuario pulsa sobre el icono situado en la parte superior izquierda para descartar la ruta.

* El sistema muestra un mensaje de confirmación.

* El usuario selecciona “No”.

* El sistema continúa tomando datos y oculta el mensaje.

Nombre de la ruta vacío:

8. El usuario deja el campo nombre de la ruta vacío.

9. El sistema indica al usuario que debe rellenar el nombre de la ruta.

10. El usuario continúa por el paso 7 del escenario principal.

Continuar añadiendo la ruta:

14. El usuario pulsa “Cancelar”.

15. El sistema continúa tomando datos y oculta el formulario para guardar la ruta.

Tabla 10: CU10 - Realizar una ruta.

Título	Realizar una ruta.
Descripción	Un usuario podrá realizar una ruta creada por otro usuario.
Requisitos	RF-ROUT-2
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la pantalla de detalles de una ruta.

Post-condición	El usuario realiza el recorrido de la ruta.
Escenario principal	
1. El usuario pulsa “Realizar ruta”. 2. El sistema muestra un mapa con la ubicación actual del usuario y el recorrido a seguir. 3. El sistema recoge y almacena datos de la ubicación en tiempo real del usuario periódicamente. 4. El usuario pulsa “Finalizar recorrido”. 5. El sistema comprueba que el usuario haya realizado la ruta. 6. El sistema almacena el recorrido y redirige al usuario a la pantalla de rutas.	
Escenario alternativo	
Descartar el recorrido (en cualquier momento entre los pasos 3 y 4): * El usuario pulsa sobre el icono situado en la parte superior izquierda para descartar el recorrido. * El sistema muestra un mensaje de confirmación. * El usuario selecciona “Sí”. * El sistema detiene la toma de datos y redirige al usuario a la pantalla de información de la ruta. **Continuar con el recorrido (en cualquier momento entre los pasos 3 y 4):** * El usuario pulsa sobre el icono situado en la parte superior izquierda para descartar el recorrido. * El sistema muestra un mensaje de confirmación. * El usuario selecciona “No”. * El sistema continúa tomando datos y oculta el mensaje.	

Tabla 11: CU11 - Editar la información de una ruta.

Título	Editar la información de una ruta.
Descripción	Un usuario podrá editar el nombre o la visibilidad de una de sus rutas tras crearla.
Requisitos	RF-ROUT-3

Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la pantalla “Mis rutas” dentro de la sección “Cuenta”. El usuario ha creado al menos 1 ruta.
Post-condición	El usuario edita la información de la ruta.
Escenario principal	
1. El usuario pulsa “Editar ruta” sobre la ruta que desea editar. 2. El sistema muestra el formulario de editar la ruta. 3. El usuario cambia el nombre de la ruta (si procede). 4. El sistema muestra el nuevo nombre de la ruta. 5. El usuario cambia la visibilidad de la ruta (si procede). 6. El sistema muestra la nueva visibilidad de la ruta. 7. El usuario pulsa “Guardar cambios”. 8. El sistema comprueba la validez de los datos introducidos. 9. El sistema actualiza la información de la ruta y redirige al usuario a la pantalla “Mis rutas”.	
Escenario alternativo	
Campos sin rellenar: 3. El usuario deja vacío el campo nombre de la ruta. 4. El sistema indica al usuario que debe rellenar el campo nombre de la ruta. 5. El usuario continúa con el paso 2 del escenario principal.	

Tabla 12: CU12 - Eliminar una ruta creada por el usuario.

Título	Eliminar una ruta creada por el usuario.
Descripción	Un usuario podrá eliminar sus rutas creadas.
Requisitos	RF-ROUT-4
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la pantalla “Mis rutas” dentro de la sección “Cuenta”. El usuario ha creado al menos 1 ruta.
Post-condición	El usuario elimina una de sus rutas creadas.
Escenario principal	

1. El usuario pulsa “Editar ruta” sobre la ruta que desea eliminar. 2. El sistema muestra el formulario de editar la ruta. 3. El usuario pulsa “Eliminar ruta”. 4. El sistema muestra un mensaje de confirmación. 5. El usuario selecciona “Sí”. 6. El sistema elimina la ruta de las rutas del usuario y lo redirige a la pantalla “Mis rutas”.
Escenario alternativo
No elimina la ruta: 5. El usuario selecciona “No”. 6. El sistema oculta el mensaje de confirmación. 7. El usuario continúa con el paso 2 del escenario principal.

Tabla 13: CU13 - Eliminar una ruta completada por el usuario.

Título	Eliminar una ruta completada por el usuario.
Descripción	Un usuario podrá eliminar sus rutas completadas.
Requisitos	RF-ROUT-4
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la pantalla “Mis rutas” dentro de la sección “Cuenta”. El usuario ha completado al menos 1 ruta.
Post-condición	El usuario elimina una de sus rutas completadas.
Escenario principal	
1. El usuario pulsa “Eliminar ruta” sobre la ruta que desea eliminar. 2. El sistema muestra un mensaje de confirmación. 3. El usuario selecciona “Sí”. 4. El sistema elimina la ruta de sus rutas completadas y lo redirige a la pantalla “Mis rutas”.	
Escenario alternativo	
No elimina la ruta: 3. El usuario selecciona “No”. 4. El sistema oculta el mensaje de confirmación.	

Tabla 14: CU14 - Ver las rutas disponibles.

Título	Ver las rutas disponibles.
Descripción	Un usuario podrá ver todas las rutas registradas hasta la fecha, ordenadas de más reciente a más antigua.
Requisitos	RF-VSRT-1
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección "Carriles bici". GoSykel tiene al menos 1 ruta.
Post-condición	El usuario ve las rutas disponibles.
Escenario principal	
1. El usuario pulsa "Rutas". 2. El sistema muestra la lista de rutas disponibles.	
Escenario alternativo	
Error al cargar las rutas: 2. El sistema falla al cargar las rutas y muestra un mensaje de error.	

Tabla 15: CU15 - Filtrar las rutas disponibles.

Título	Filtrar las rutas disponibles.
Descripción	Un usuario podrá filtrar las rutas registradas según diferentes criterios: creador, (parte del) nombre, visibilidad, rutas más frecuentes y rutas creadas por mí.
Requisitos	RF-VSRT-2
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección "Carriles bici". GoSykel tiene al menos 1 ruta.
Post-condición	El usuario filtra las rutas disponibles.
Escenario principal	
1. El usuario pulsa "Rutas". 2. El sistema muestra la lista de rutas disponibles. 3. El usuario selecciona el filtro deseado. 4. El sistema muestra las rutas que coinciden con el filtro aplicado.	

Escenario alternativo
No hay rutas que coinciden con el filtro aplicado: 4. El sistema muestra un mensaje informando de lo sucedido.

Tabla 16: CU16 - Ver la información de la ruta seleccionada.

Título	Ver la información de la ruta seleccionada.
Descripción	Un usuario podrá consultar el recorrido, las estadísticas y los usuarios más rápidos de la ruta seleccionada.
Requisitos	RF-VSRT-4, RF-VSRT-4.1, RF-VSRT-4.2
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario está en la sección "Rutas" (ver CU14). GoSykel tiene al menos 1 ruta.
Post-condición	El usuario visualiza la información de la ruta seleccionada.
Escenario principal	
1. El usuario pulsa "Ver ruta". 2. El sistema muestra el recorrido, las estadísticas y los usuarios más rápidos de la ruta seleccionada.	
Escenario alternativo	
El usuario selecciona una ruta privada que no ha sido creada por él: 2. El sistema muestra un mensaje informando de lo sucedido.	

Tabla 17: CU17 - Ver las rutas creadas y completadas por el usuario.

Título	Ver las rutas creadas y completadas por el usuario.
Descripción	Un usuario podrá ver desde su perfil las rutas que ha creado y las que ha completado.
Requisitos	RF-VSRT-5
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección "Cuenta". El usuario ha creado al menos 1 ruta. El usuario ha completado al menos 1 ruta.

Post-condición	El usuario visualiza la lista de rutas creadas y completadas.
Escenario principal	
1. El usuario pulsa “Mis rutas”. 2. El sistema muestra las rutas creadas y completadas por el usuario.	
Escenario alternativo	
Error al cargar las rutas: 2. El sistema falla al cargar las rutas y muestra un mensaje de error.	

Tabla 18: CU18 - Ver los carriles bici disponibles.

Título	Ver los carriles bici disponibles.
Descripción	Un usuario podrá ver los carriles bici disponibles.
Requisitos	RF-BICI-1
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección “Rutas”.
Post-condición	El usuario ve los carriles bici disponibles.
Escenario principal	
1. El usuario pulsa “Carriles bici”. 2. El sistema muestra el mapa con los carriles bici disponibles.	
Escenario alternativo	
Error al cargar los carriles bici: 2. El sistema falla al cargar los carriles bici y muestra un mensaje de error.	

Tabla 19: CU19 - Añadir un nuevo carril bici.

Título	Añadir un nuevo carril bici.
Descripción	Un usuario podrá guardar o descartar un nuevo carril bici.
Requisitos	RF-BICI-2, RF-BICI-3

Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección "Carriles bici". El usuario ha permitido a la aplicación tomar datos de su geoposición.
Post-condición	El usuario añade el carril bici correctamente.

Escenario principal

1. El usuario pulsa "+ CARRIL BICI".
2. El sistema muestra un mapa con la ubicación actual del usuario.
3. El sistema recoge y almacena datos de la ubicación en tiempo real del usuario periódicamente.
4. El usuario pulsa sobre "Guardar" cuando considera que el carril bici ha acabado.
5. El sistema muestra un mensaje de confirmación.
6. El usuario selecciona "Guardar".
7. El sistema almacena el nuevo tramo de carril bici y redirige al usuario a la pantalla "Carriles bici".

Escenario alternativo

Descartar el carril bici (en cualquier momento entre los pasos 3 y 4):

* El usuario pulsa sobre el icono situado en la parte superior izquierda para descartar el carril bici.
* El sistema muestra un mensaje de confirmación.
* El usuario selecciona "Sí".
* El sistema detiene la toma de datos y redirige al usuario a la pantalla "Carriles bici".

Continuar con el carril bici (en cualquier momento entre los pasos 3 y 4):

* El usuario pulsa sobre el icono situado en la parte superior izquierda para descartar el carril bici.
* El sistema muestra un mensaje de confirmación.
* El usuario selecciona "No".
* El sistema continúa tomando datos y oculta el mensaje.

Continuar añadiendo el carril bici:

6. El usuario pulsa "Cancelar".
7. El sistema continúa tomando datos y oculta el mensaje.

Tabla 20: CU20 - Conseguir puntos por registrarse.

Título	Conseguir puntos por registrarse.
Descripción	Un usuario podrá conseguir puntos por registrarse.
Requisitos	RF-GAME-1
Pre-condición	El usuario no dispone de una cuenta en GoSykel.
Post-condición	El usuario obtiene puntos tras registrarse.
Escenario principal	
1. El usuario se registra en la aplicación. 2. El sistema comprueba que no estuviera registrado. 3. El sistema registra al usuario. 4. El sistema modifica los puntos del nuevo usuario.	

Tabla 21: CU21 - Conseguir puntos por iniciar sesión diariamente.

Título	Conseguir puntos por iniciar sesión diariamente.
Descripción	Un usuario podrá conseguir puntos por iniciar sesión diariamente.
Requisitos	RF-GAME-2
Pre-condición	El usuario está registrado en GoSykel.
Post-condición	El usuario obtiene puntos tras iniciar sesión una vez al día.
Escenario principal	
1. El usuario inicia sesión en la aplicación. 2. El sistema comprueba el último acceso del usuario. 3. El sistema modifica los puntos del usuario (si procede).	

Tabla 22: CU22 - Conseguir puntos por crear nuevas rutas.

Título	Conseguir puntos por crear nuevas rutas.
Descripción	Un usuario podrá conseguir puntos creando nuevas rutas.
Requisitos	RF-GAME-3
Pre-condición	El usuario está registrado en GoSykel.

Post-condición	El usuario obtiene puntos tras crear una nueva ruta.
Escenario principal	
1. El usuario crea una nueva ruta. 2. El sistema modifica los puntos del usuario.	

Tabla 23: CU23 - Conseguir puntos por completar rutas de otros usuarios.

Título	Conseguir puntos por completar rutas de otros usuarios.
Descripción	Un usuario podrá conseguir puntos completando una ruta creada por otro usuario.
Requisitos	RF-GAME-4
Pre-condición	El usuario está registrado en GoSykel. El usuario visualiza la información de la ruta a realizar (ver CU16).
Post-condición	El usuario obtiene puntos tras realizar la ruta seleccionada.
Escenario principal	
1. El usuario realiza el recorrido de la ruta seleccionada. 2. El sistema comprueba que el usuario ha realizado el recorrido de la ruta. 3. El sistema modifica los puntos del usuario (si procede).	

Tabla 24: CU24 - Conseguir puntos por realizar una ruta frecuentemente.

Título	Conseguir puntos por realizar una ruta frecuentemente.
Descripción	Un usuario podrá conseguir puntos realizando una ruta de otro usuario frecuentemente.
Requisitos	RF-GAME-5
Pre-condición	El usuario está registrado en GoSykel. El usuario visualiza la información de la ruta a realizar (ver CU16).
Post-condición	El usuario obtiene puntos tras realizar la ruta frecuentemente.
Escenario principal	
1. El usuario realiza el recorrido de la ruta seleccionada. 2. El sistema comprueba que el usuario ha realizado la ruta frecuentemente.	

3. El sistema modifica los puntos del usuario.

Tabla 25: CU25 - Conseguir puntos por añadir nuevos carriles bici.

Título	Conseguir puntos por añadir nuevos carriles bici.
Descripción	Un usuario podrá conseguir puntos registrando carriles bici.
Requisitos	RF-GAME-6
Pre-condición	El usuario está registrado en GoSykel.
Post-condición	El usuario obtiene puntos tras registrar un nuevo carril bici.
Escenario principal	
1. El usuario registra un nuevo carril bici. 2. El sistema modifica los puntos del usuario.	

Tabla 26: CU26 - Ver los usuarios del *ranking*.

Título	Ver los usuarios del *ranking*.
Descripción	Un usuario podrá ver un *ranking* de puntos global donde aparece su posición actual en el *ranking* y la posición del resto de usuarios.
Requisitos	RF-RANK-1, RF-RANK-2, RF-RANK-3
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección "Carriles bici". GoSykel tiene al menos 5 usuarios.
Post-condición	El usuario ve el *ranking* de usuarios, su posición y la posición del resto de miembros.
Escenario principal	
1. El usuario pulsa *"Ranking"*. 2. El sistema muestra el *ranking* de usuarios, la posición del usuario actual y la posición del resto de los usuarios.	
Escenario alternativo	
Error al cargar el *ranking*: 2. El sistema falla al cargar los datos y muestra un mensaje de error.	

Tabla 27: CU27 - Ver los artículos de la tienda.

Título	Ver los artículos de la tienda.
Descripción	Un usuario podrá ver los avatares, insignias y encabezados disponibles.
Requisitos	RF-STOR-1
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario se encuentra en la sección "Cuenta". GoSykel tiene artículos disponibles.
Post-condición	El usuario ve los artículos de la tienda.
Escenario principal	
1. El usuario pulsa "Tienda". 2. El sistema muestra los avatares, insignias y encabezados disponibles en la tienda.	
Escenario alternativo	
Error al cargar los artículos: 2. El sistema falla al cargar los artículos y muestra un mensaje de error.	

Tabla 28: CU28 - Ver la información de un artículo de la tienda y comprarlo.

Título	Ver la información de un artículo de la tienda y comprarlo.
Descripción	Un usuario podrá ver la información de un artículo de la tienda y comprarlo con los puntos conseguidos hasta el momento.
Requisitos	RF-STOR-2, RF-STOR-3
Pre-condición	El usuario ha iniciado sesión en GoSykel. El usuario dispone de suficientes puntos para comprar el artículo. El usuario se encuentra en la tienda (ver CU27).
Post-condición	El usuario ve la información de un artículo y lo compra.
Escenario principal	
1. El usuario selecciona un artículo no comprado para conocer la información. 2. El sistema muestra la información del artículo. 3. El usuario pulsa sobre el carrito de la compra. 4. El sistema muestra un mensaje de confirmación de la compra del artículo. 5. El usuario selecciona "Sí".	

6. El sistema canjea los puntos del usuario, muestra un mensaje de confirmación y lo redirige a la tienda.
Escenario alternativo
No compra el artículo: 5. El usuario selecciona “No”. 6. El sistema oculta el mensaje de confirmación. 7. El usuario continúa con el paso 2 del escenario principal. **Elige un artículo ya comprado:** 1. El usuario selecciona un artículo comprado para conocer la información. 2. El sistema muestra la información del artículo y un mensaje que indica que ya ha sido comprado.

5

Modelado y diseño

Una vez la planificación concluye o al menos gran parte de la misma dependiendo de la metodología que hayamos empleado, el proyecto entra en la fase de modelado y diseño. En este capítulo analizaremos aspectos importantes del diseño de la aplicación como son el modelo de datos elegido y las líneas de diseño que hemos tomado para el desarrollo de la aplicación.

5.1. Modelo de datos

En la Figura 3 mostramos el modelo de datos que sigue nuestra aplicación. Como podemos observar está conformado por 5 colecciones: *User, Item, Route, Bycicle lane* y *Token*. Cada colección tiene determinados atributos que explicaremos a continuación.

Debemos tener en cuenta que al haber empleado una base de datos NoSQL las claves externas aparecen como atributos de otras colecciones. Esto podemos apreciarlo en la colección *Route* que almacena al autor de la ruta con un valor de tipo *ObjectId* que es el tipo predefinido en MongoDB para los identificadores.

5.1.1. Colección *User*

La colección *User* almacena la información de los usuarios de la aplicación. De cada usuario guardamos la siguiente información:

- `_id`: identificador del documento.
- `email`: correo electrónico del usuario que se ha registrado.
- `nickname`: nombre de usuario del usuario que se ha registrado.
- `points`: puntos conseguidos por el usuario hasta la fecha.

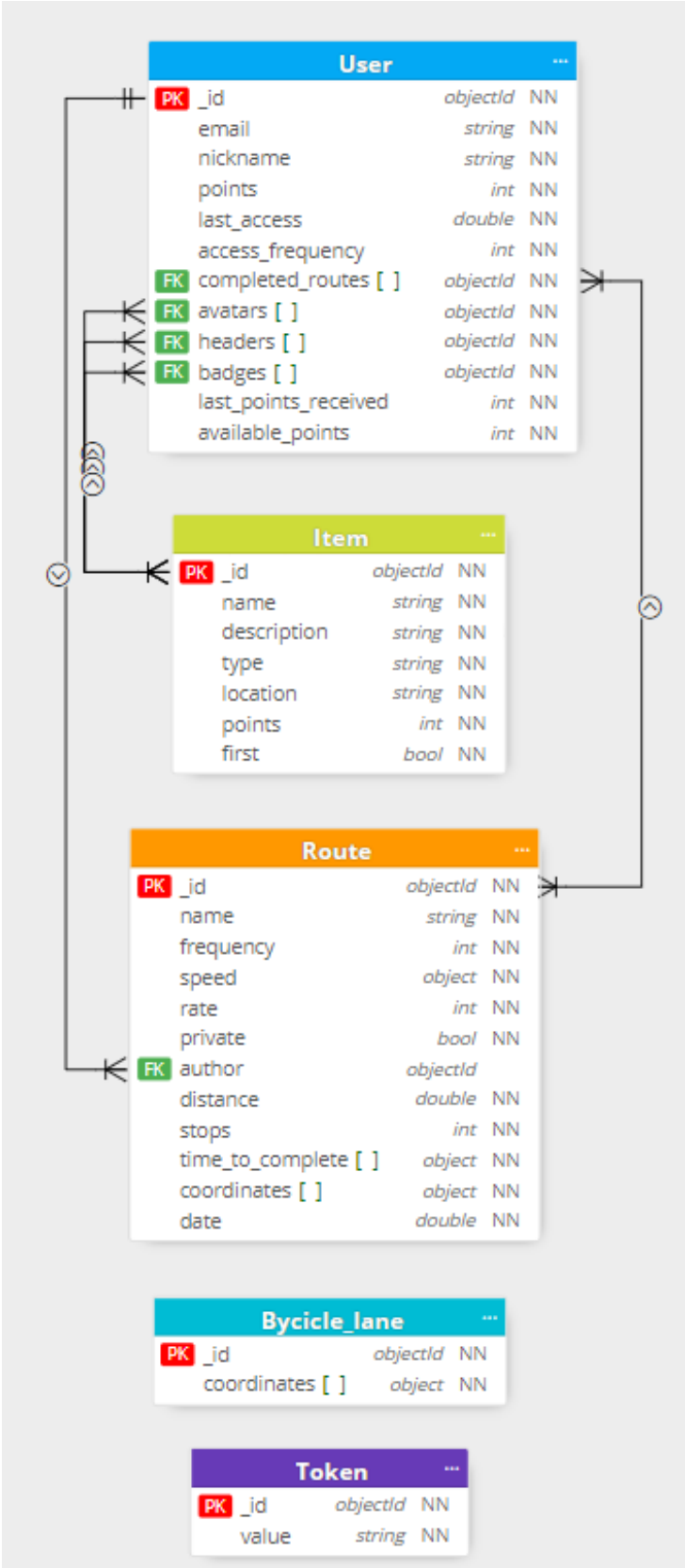


Figura 3: Modelo de datos de la aplicación

- `last_access`: representa la última vez que accedió el usuario a la aplicación en formato POSIX.
- `access_frequency`: frecuencia de acceso, nos ayuda a determinar si un usuario accede diariamente a la aplicación o no.
- `completed_routes`: almacena los identificadores de las rutas que ha completado el usuario. No obstante, si un usuario crea una nueva ruta, ésta no se añade a esta lista.
- `avatars`: almacena los identificadores de los avatares que ha obtenido el usuario.
- `headers`: almacena los identificadores de los encabezados que ha obtenido el usuario.
- `badges`: almacena los identificadores de las insignias que ha obtenido el usuario.
- `last_points_received`: almacena los últimos puntos que ha recibido el usuario por iniciar sesión o añadir nuevas rutas.
- `available_points`: guarda los puntos disponibles para gastar en la tienda de avatares, insignias y encabezados.

5.1.2. Colección *Item*

La colección *Item* almacena la información de los objetos disponibles en la tienda. De cada objeto guardamos la siguiente información:

- `_id`: identificador del documento.
- `name`: nombre del objeto.
- `description`: descripción del objeto.
- `type`: tipo del objeto, *AVATAR, HEADER* o *BADGE.*
- `location`: URL donde se encuentra almacenado el objeto.
- `points`: puntos que debe tener el usuario para obtener dicho objeto.

5.1.3. Colección *Route*

La colección *Route* almacena la información de las rutas creadas por los usuarios en GoSykel. De cada ruta guardamos la siguiente información:

- `_id`: identificador del documento.
- `name`: nombre de la ruta.
- `frequency`: frecuencia de realización de la ruta.
- `speed`: velocidad del usuario que ha creado la ruta. Este objeto almacena dos velocidades, una que considera las paradas que ha hecho (*general*) y otra que representa el tiempo que ha estado moviéndose el usuario (*moving*).
- `rate`: valoración de la ruta en una escala de 1 a 5.
- `private`: booleano que representa la visibilidad de la ruta. Si es *true* la ruta es privada, en otro caso, es pública.
- `author`: almacena el identificador del autor de la ruta.
- `distance`: almacena la distancia de la ruta.
- `stops`: número de paradas realizadas por el creador de la ruta durante el transcurso de la misma.
- `time_to_complete`: lista de objetos que almacena el tiempo que han tardado en completar la ruta los diferentes usuarios que la realizan.
- `coordinates`: almacena las coordenadas que componen la ruta.
- `date`: fecha en que se creó la ruta en formato POSIX.

5.1.4. Colección *Bycicle lane*

La colección *Bycicle lane* almacena la información de los carriles bici añadidos por los usuarios de GoSykel. De cada carril bici guardamos la siguiente información:

- `_id`: identificador del documento.
- `coordinates`: almacena las coordenadas que componen el carril bici.

5.1.5. Colección *Token*

La colección *Token* almacena la información encriptada de los usuarios que tienen la sesión iniciada en un momento concreto. De cada *token* guardamos la siguiente información:

- `_id`: identificador del documento.
- `value`: valor encriptado del identificador del usuario y la fecha de acceso del mismo.

5.2. Diagramas de navegación

Mientras que el usuario está navegando por GoSykel atraviesa diferentes pantallas, relacionadas entre sí. En este sentido, hemos tratado de hacer la navegación lo más sencilla posible, dando lugar a 20 pantallas diferentes dentro de la aplicación móvil. Hay que tener en cuenta que es posible navegar hacia atrás en aquellas pantallas que ofrezcan la flecha de navegación. En la Figura 4 podemos observar el flujo de navegación dentro de la aplicación móvil.

Por otro lado, durante el desarrollo creamos otro diagrama de navegación donde se muestran los diferentes elementos que permiten la navegación dentro de la aplicación. A medida que el proyecto iba cogiendo forma, era necesario recoger en una imagen el esquema de navegación con el diseño actual para futuras explicaciones y/o aclaraciones que pudieran surgir. De este modo, si cualquier persona necesita entender la estructura interna del proyecto podría hacerlo.

Concretamente, React Native dispone de *Tab Navigators*, *Stack Navigators* y *Screens*. Los *Tab Navigators* proporcionan la barra inferior de navegación que permite visitar las diferentes secciones. En cambio, los *Stack Navigators* los usamos para conocer qué *Screens* son las que pertenecen a un *Tab Navigator* concreto. Debemos tener en cuenta que, como en toda jerarquía en árbol, hay una sola raíz de la que colgarán múltiples hijos. En la Figura 5 podemos observar el esquema que se comenzó a hacer al principio del desarrollo y que durante las diferentes iteraciones hemos ido refinando hasta conseguir la versión que mostramos.

5.3. Proceso de diseño

Al usar una metodología iterativa basada en Scrum, el diseño es una etapa que siempre se ejecuta en cada iteración ya que tenemos que diseñar nuevas interfaces de usuario o elegir el

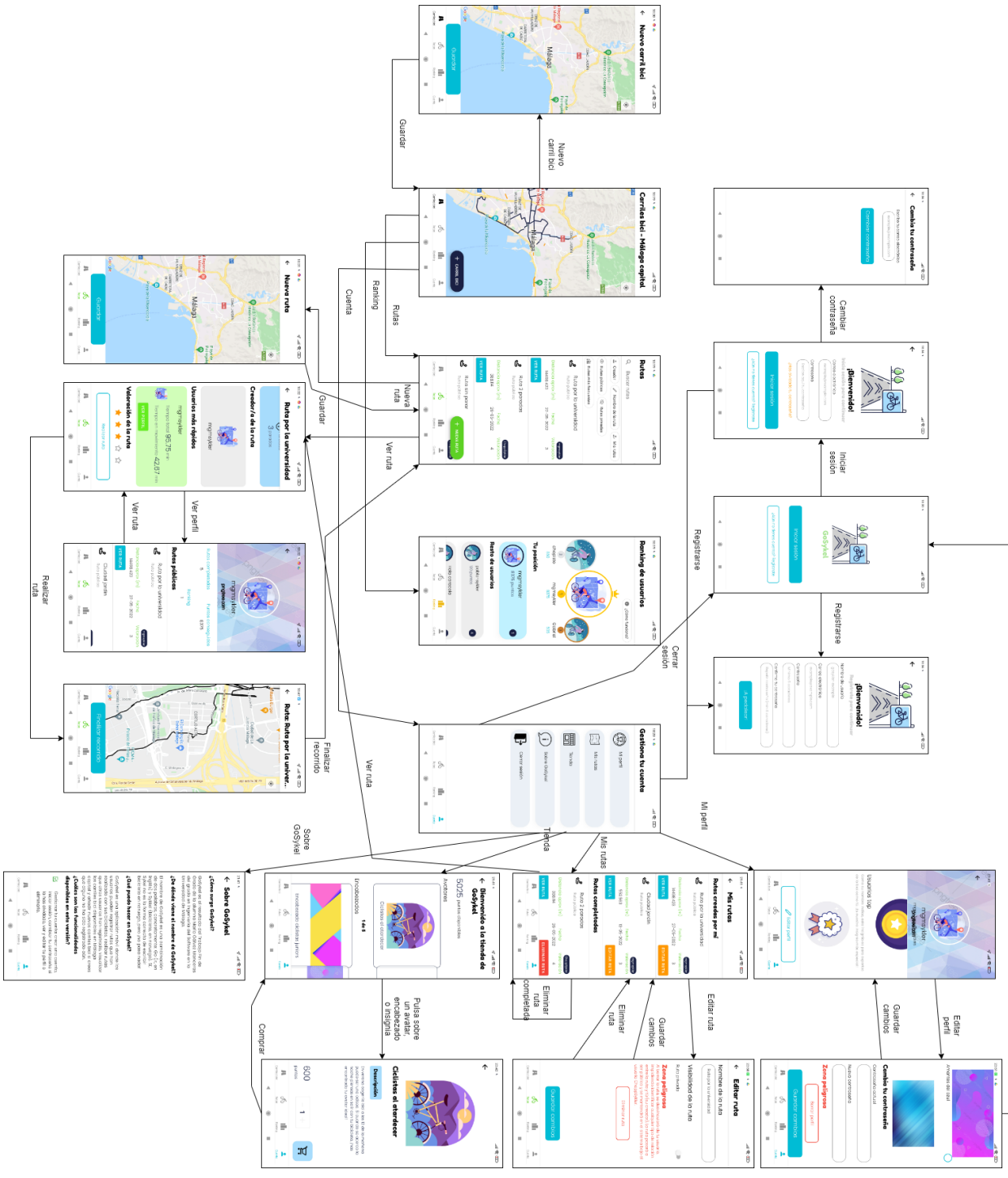


Figura 4: Diagrama de navegación de GoSykel

Figura 5: Diagrama de navegación realizado durante el desarrollo de GoSykel

esquema de colores que emplearemos a lo largo de la aplicación, dependiendo de la fase del proyecto donde nos encontremos.

Por ello, en los siguientes párrafos, repasaremos la línea de diseño que hemos seguido durante el desarrollo de la aplicación centrándonos en aspectos como el icono, los colores principales y el nombre de la misma. Finalmente, nos centraremos en el diseño de la interfaz de usuario que realizamos al principio del proyecto.

5.3.1. Línea de diseño

Hemos procurado que el diseño sea sencillo y que el usuario comprenda fácilmente lo que le estamos presentando. Igualmente, hemos empleado iconos y figuras fáciles de entender para minimizar el número de fallos que cometen los usuarios y proporcionar una buena experiencia de usuario.

Icono de la aplicación

Hemos desarrollado la aplicación teniendo en mente principalmente dos colores, el azul y el verde, en tonalidades similares a las que aparecen en el icono de GoSykel (Figura 6) que ha sido generado a partir de la imagen obtenida de esta página.

Figura 6: Icono de GoSykel

Elegimos este icono ya que representa los valores principales de GoSykel mediante la bicicleta y su correspondiente carril bici haciendo gala de la sostenibilidad y de la importancia de la misma para reducir las emisiones de carbono y contribuir a un aire más limpio dentro de las ciudades.

Otro de los motivos por el cual seleccionamos este icono es por la gama de colores. Tanto el azul, como el verde y el resto de colores azul marino, negro y gris son muy suaves y nos

ayudan a romper con las tonalidades tan conocidas de estos colores.

Colores principales

La plantilla de colores empleada abarca diferentes tipos de colores, aunque en la aplicación se usan más colores, aquí nos centramos en los 3 principales (en hexadecimal):

- 00bcd4 como color primario de la aplicación. Principalmente lo usamos en elementos interactivos como botones o *checboxes*. Elegimos este color porque destaca y además se ve perfectamente en el dispositivo del usuario final. Debemos tener en cuenta que usar este tipo de colores es beneficioso porque ayuda a distinguir el fondo del resto de elementos interactivos.

- 5dd124 es otro de los colores principales de la aplicación. En línea con el icono de la aplicación, necesitábamos un color que acompañase al azul anterior y nos permitiese combinarlos fácilmente. Dentro de la aplicación lo usamos en la barra de navegación inferior y para mostrar información.

- 192043 es nuestro color negro personalizado, prácticamente igual al negro que nos proporciona la paleta de colores del icono. Creemos que contar con un color diferente al habitual resalta y contrasta con el resto de colores de la aplicación.

De cara a facilitar el desarrollo, en el proyecto de Expo hemos creado un archivo *Colors.js* donde definimos los colores que usaremos y asignamos su correspondiente valor en hexadecimal. Así lo tenemos todo en un mismo lugar y en caso de querer cambiar, añadir o eliminar colores, sólo tenemos que modificar el archivo *Colors.js* y veremos los nuevos cambios.

Explicación del nombre: GoSykel

Cuando tuvimos que pensar en un nombre para la aplicación queríamos que éste fuera corto, fácil de pronunciar y de recordar. Tras un tiempo pensando nos decantamos por GoSykel.

Inicialmente, pensamos que GoSykkel era un buen nombre para la aplicación ya que es la unión de dos palabras: Go (del verbo ir, en inglés) y Sykkel (bicicleta, en noruego), ir en bicicleta. Sin embargo, tras investigar si había alguna marca registrada con dicho nombre nos dimos cuenta de que sí y por tanto optamos por quitar una K del nombre original derivando finalmente en GoSykel.

5.3.2. Prototipado interfaz de usuario

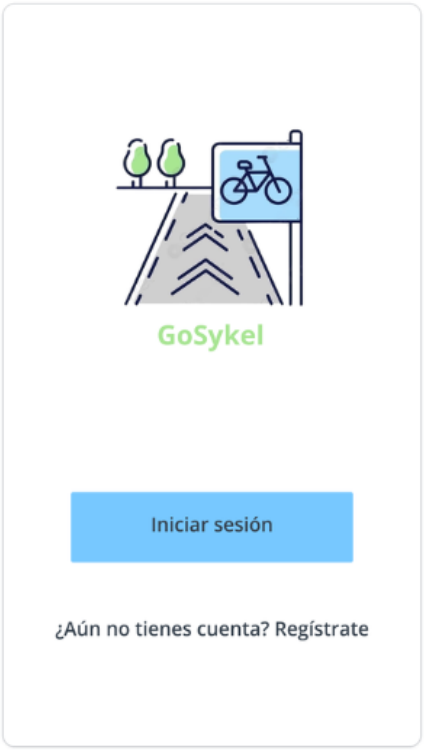
Una vez decididos el nombre y el icono que tendría la aplicación, pasamos a realizar unos bocetos de cómo sería la interfaz de usuario. Mediante estos bocetos pudimos plantear una primera visión general de cómo serían las principales pantallas de la aplicación y la navegación entre las mismas.

Al tratarse de un boceto, dista del resultado obtenido finalmente, sin embargo, nos ayuda a sentar las bases de lo que pretendemos conseguir y nos proporciona una guía de por dónde debemos comenzar en el desarrollo. Como mencionamos en el Capítulo 2, usamos Marvel para realizar dichos bocetos interactivos.

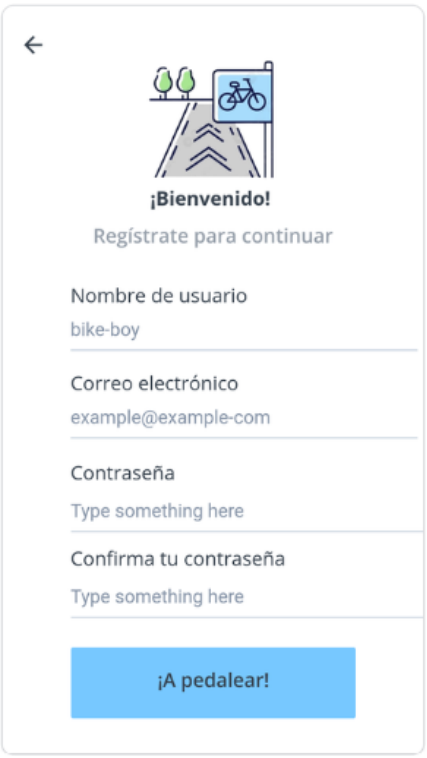
En la Figura 7a podemos ver cómo sería la pantalla de inicio de la aplicación. Es la primera pantalla que ve el usuario nada más iniciar la aplicación. En ella, el usuario identifica claramente la aplicación mediante el uso del icono de la aplicación en grande y claramente observa dos opciones diferentes: registrarse o iniciar sesión, poniendo énfasis en el inicio de sesión que será el uso habitual.

Cuando el usuario pulsa sobre la opción para registrarse visualiza la pantalla de registro (Figura 7b) donde mantenemos el icono de la aplicación adaptado al tamaño de la pantalla. La introducción de información la pedimos en la parte central y hacia abajo y siguiendo el mismo esquema previo mostramos el botón para acabar de introducir datos. De manera más discreta le permitimos volver atrás con un icono en la parte superior izquierda. Seguiremos este esquema en todas las pantallas que requieran introducir información.

En cambio, si el usuario decide iniciar sesión, lo dirigimos a la pantalla de inicio de sesión (Figura 7c) donde, al igual que en la anterior, mantenemos el mismo diseño. Podría ser que el usuario haya olvidado su contraseña, en tal caso, ubicamos antes del botón de iniciar sesión un botón con un color más suave para cambiar la contraseña.



(a) Pantalla inicial



(b) Registro en la aplicación

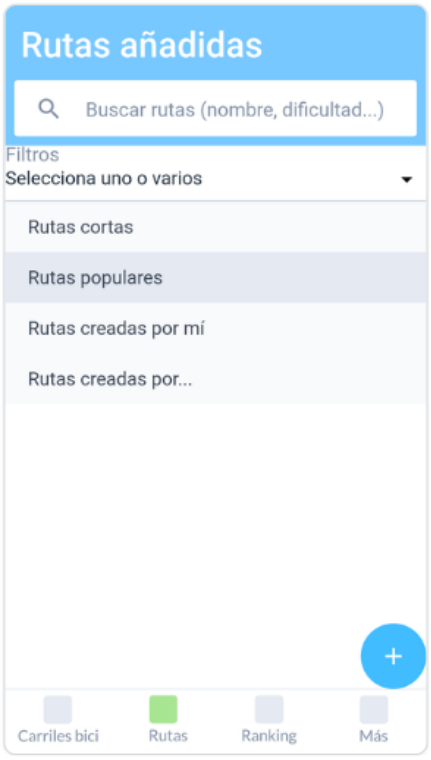
(c) Inicio de sesión en la aplicación

Figura 7: Pantallas inicial, de registro e inicio de sesión

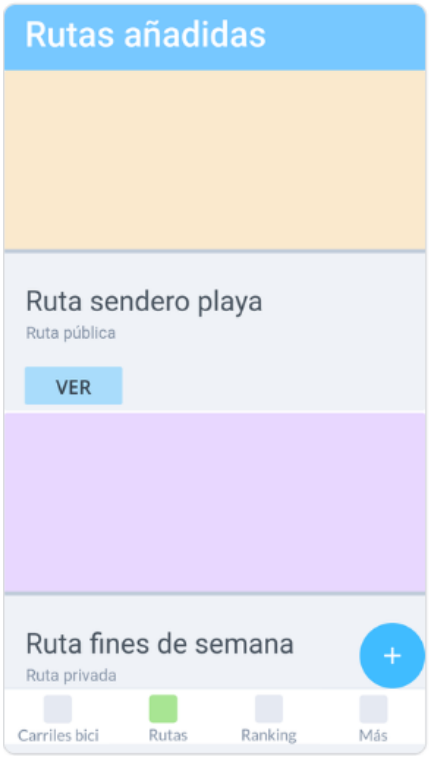
Cuando nuestros usuarios inician sesión o se registran empiezan visualizando los carriles bici disponibles en Málaga capital (Figura 8). El elemento principal de la pantalla es el mapa que muestra los diferentes carriles bici disponibles y en un color llamativo un botón para añadir un nuevo tramo de carril bici. Seguiremos este diseño cuando el usuario añada nuevas rutas o carriles bici. De este modo, el usuario identifica ambos botones como acciones similares. Hemos de destacar la presencia del menú de navegación inferior, que estará presente en toda la aplicación. Compuesto por 4 secciones, cuando el usuario cambia de sección, cambia el color de la sección seleccionada para ayudarlo a distinguir en qué sección se encuentra.

Figura 8: Carriles bici

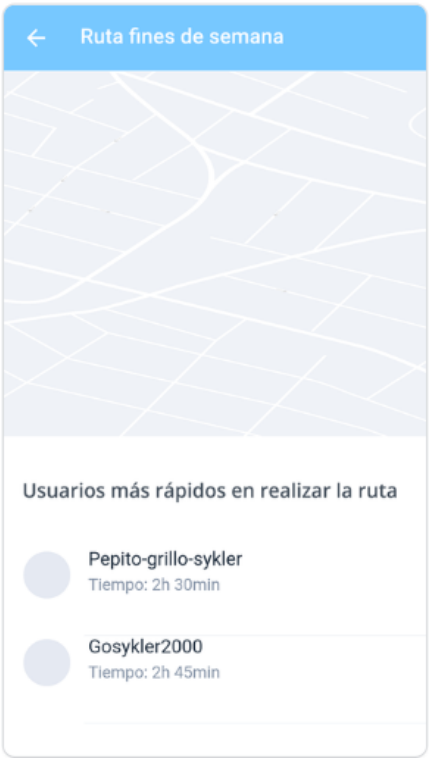
Cuando el usuario pasa a la sección Rutas (Figura 9), ubicamos un buscador en la barra superior de la pantalla (Figura 9a) para centrar la información importante en el resto de la pantalla. Inicialmente el buscador sería un desplegable para así dar protagonismo a otros elementos. Los elementos centrales de la pantalla son las diferentes rutas disponibles (Figura 9b), ocupando el máximo espacio posible para poder visualizarlas con claridad. Desde esta pantalla podemos ver los detalles de la ruta seleccionada (Figura 9c). Inicialmente planteamos que la vista de los detalles de la ruta fuese sencilla, permitiendo volver atrás con un icono discreto en la esquina superior izquierda y eligiendo como protagonista el mapa donde mostramos el recorrido a seguir para completar la ruta. En la parte inferior de la pantalla mostramos información relativa a la ruta, como son los usuarios más rápidos.



(a) Filtros aplicables a las rutas



(b) Rutas disponibles en GoSykel



(c) Detalles de la ruta seleccionada

Figura 9: Pantallas principales de la sección Rutas

Para mostrar el *ranking* de usuarios (Figura 10) y los puntos que llevan así como la posición en que están respecto a otros usuarios de la aplicación, optamos por un diseño llamativo, donde el protagonismo lo tuvieran los 3 primeros usuarios y en menor medida el cuarto y quinto. Le damos bastante prioridad a la posición del usuario que ha iniciado sesión, ubicando dicha información a la vista del usuario. Esta pantalla la alcanza moviéndose a la sección *Ranking*.

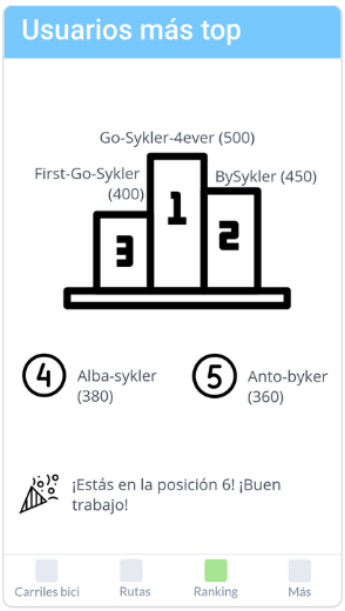


Figura 10: *Ranking* de usuarios

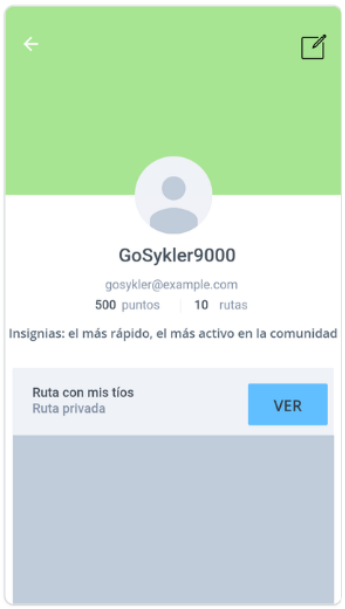


Figura 11: Perfil del usuario

Uno de los requisitos planteados inicialmente es la posibilidad de que los usuarios registrados puedan consultar su perfil (Figura 11), teniendo así una vista general de sus logros y rutas creadas hasta el momento. En esta pantalla enfatizamos el encabezado y la foto de perfil del usuario en la parte central en vertical junto al nombre de usuario y su correo electrónico. En menor medida enfatizamos las rutas creadas por el usuario, mostrándolas en la mitad inferior de la pantalla. De cara a que el usuario pueda editar su perfil le ofrecemos la posibilidad con un discreto icono en la esquina superior derecha ya que no será el uso habitual.

Para que el usuario pueda gastar los puntos conseguidos en elementos para personalizar su perfil, acordamos en poner a disposición de los usuarios una tienda donde poder obtener dichos elementos. En la tienda (Figura 12), en la parte superior de la pantalla clasificamos los elementos disponibles en horizontal, para centrar la atención del usuario en un grupo concreto. Tras esto, los diferentes elementos disponibles del tipo seleccionado cobran importancia ocupando el resto del espacio disponible para su correcta visualización. Ofrecemos al usuario

la posibilidad de comprarlos mediante el uso del icono apropiado.

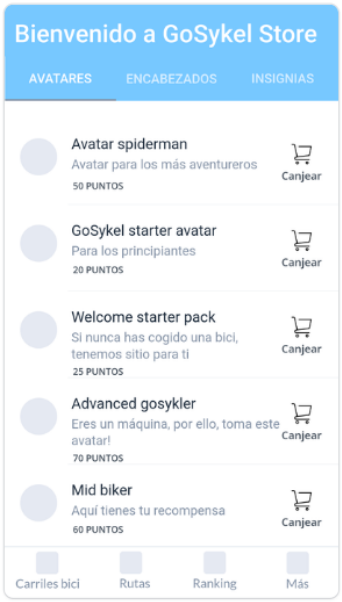


Figura 12: Tienda donde canjear los puntos obtenidos

Tras terminar los diseños mostrados anteriormente contábamos con un prototipo navegable pulsando sobre diferentes botones que nos llevaban a una pantalla u otra. Enviamos el diseño al tutor para validar que lo que pretendíamos conseguir era correcto y sencillo desde el punto de vista del usuario. Finalmente, obtuvimos el visto bueno por su parte.

6

Desarrollo e implementación

A lo largo de este capítulo analizaremos el trabajo realizado en cada una de las iteraciones que han formado parte del desarrollo así como mencionaremos los problemas encontrados durante el desarrollo y cómo planteamos la solución de los mismos.

Luego estudiaremos los diferentes mecanismos de seguridad que hemos empleado para procurar que las comunicaciones entre ambas partes sean seguras. Como hemos mencionado anteriormente, una de las acciones realizadas en materia de seguridad es la implantación de un mecanismo de *tokens* que veremos más tarde.

Finalmente nos centraremos en las pruebas realizadas durante y después del desarrollo de la aplicación. Estas pruebas abarcan desde pruebas de la API REST usando Postman hasta pruebas con usuarios reales mediante un *test* de usabilidad.

6.1. Iteraciones realizadas

En primer lugar comenzaremos repasando las diferentes iteraciones que hemos realizado, los requisitos que se han implementado en cada una de ellas y los problemas encontrados durante la implementación de dichos requisitos.

6.1.1. Iteración 0

Durante la iteración inicial tuvimos un objetivo claro: realizar una pequeña aplicación en React Native con Expo para acercarnos a la tecnología desde un punto de vista totalmente práctico. Ya en esta iteración la aplicación móvil se comunicaba con el servidor para enviar peticiones y gestionar datos. Así aprendimos a enviar información al servidor usando JavaScript usando la API Fetch que proporciona dicho lenguaje de programación.

Esta iteración nos sirvió para conocer cómo sería la estructura de los proyectos que más tarde tendríamos que crear como proyectos definitivos, nos permitió entender que deberíamos tener 2 repositorios: uno para la aplicación móvil y otro para el servidor. Si bien es cierto que podríamos haber incluido ambos proyectos en un único repositorio, optamos por dividirlo en dos diferentes ya que al usar Heroku para los despliegues, pudimos vincular el proyecto de Heroku al repositorio de GitHub de modo que cada vez que hiciésemos *push* al repositorio, los nuevos cambios se desplegaban automáticamente en Heroku, realizando así *Continuous Deployments (CD)*. Otra cosa que aprendimos durante esta iteración fue a crear el proyecto correspondiente en Firebase para habilitar la autenticación de usuarios.

Nuestra aplicación de prueba, pese a ser simple contaba con elementos que más tarde usaríamos y que necesitábamos conocer para estudiar si era viable hacer la implementación usando React Native o por el contrario debíamos elegir otro *framework*.

Además de añadir la autenticación y registro de usuarios a la aplicación, añadimos la visualización de varios mapas, donde en cada uno mostrábamos diferentes elementos típicos de un mapa: una línea que une diferentes coordenadas (*polyline*), un marcador y la ubicación actual del usuario. También estudiamos el funcionamiento de las tareas periódicas en JavaScript ya que más tarde nos haría falta recoger la ubicación en tiempo real del usuario cada cierto intervalo de tiempo.

Mientras que desarrollábamos este pequeño prototipo realizamos una primera captura de requisitos, listando la gran mayoría de ellos, clasificándolos y planificando su implementación en las diferentes iteraciones. No obstante, este conjunto de requisitos inicial sería ampliado más tarde en iteraciones posteriores.

En cuanto a aspectos relacionados con la gestión del proyecto, creamos el tablero de Trello donde más tarde realizaríamos el seguimiento del estado actual del proyecto, también configuramos los repositorios que más tarde contendrán el código y concedimos acceso al tutor en ambos sitios.

Para tener una idea más general de cómo sería la aplicación final, durante esta primera iteración definimos los bocetos mostrados en el Capítulo 5 usando la herramienta Marvel. Estos prototipos sirvieron como apoyo en las primeras iteraciones y fueron supervisados por el tutor durante esta iteración inicial.

Al final de esta iteración definimos un primer acercamiento al esquema de la base de datos

que más tarde usaríamos. Tuvimos que aprender a usar la herramienta MoonModeler y plantear los datos que en principio nos harían falta almacenar para el correcto funcionamiento de la aplicación. En esta primera aproximación ya contábamos con las colecciones *User, Item, Route* y *Bycicle lane*. Tras contar con el visto bueno por parte del tutor pasamos a realizar la Iteración 1 donde comenzamos a implementar funcionalidades propias de GoSykel.

6.1.2. Iteración 1

A lo largo de esta iteración implementamos diferentes requisitos. Hemos de admitir que no completamos todos los requisitos planteados inicialmente para esta iteración ya que experimentamos diversos problemas durante el desarrollo de algunos de ellos. No obstante, aquellos requisitos que quedaron pendientes de completar en la Iteración 1 fueron completados en la siguiente.

En esta iteración implementamos los casos de uso relativos a los siguientes requisitos funcionales:

- RF-USER-1: Un usuario que no estuviera registrado anteriormente podrá registrarse facilitando un *email*, *nickname* y contraseña de 6 caracteres como mínimo.
- RF-USER-2.1: Un usuario podrá iniciar sesión usando su *email* y contraseña.
- RF-USER-2.2: Un usuario podrá cerrar sesión en la aplicación.
- RF-USER-2.5: Un usuario podrá cambiar su contraseña antes de iniciar sesión en caso de que la haya olvidado.
- RF-BICI-1: Un usuario podrá ver los carriles bici disponibles.

Durante el desarrollo encontramos diversos problemas que ralentizaron la velocidad con la que completábamos los requisitos. Algunos problemas surgían debido a la poca experiencia con React Native, especialmente relacionados con los estilos de los elementos de la interfaz gráfica. Otros problemas venían dados por la dificultad de compartir la información entre diferentes pantallas de la aplicación móvil.

El problema que tuvimos fue que no sabíamos cómo guardar el identificador del usuario actual en el *frontend* ya que dicho identificador sería necesario para obtener datos del perfil del

usuario o cerrar la sesión, por ejemplo. Tras investigar diferentes alternativas nos decidimos por el uso de contextos.

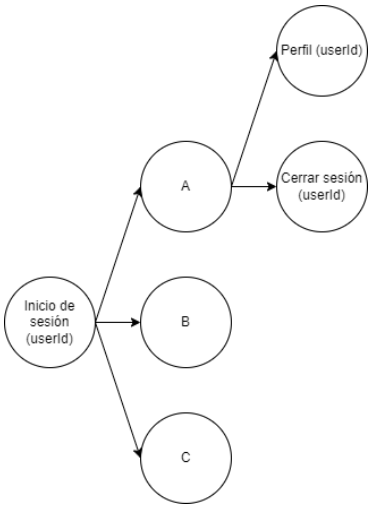


Figura 13: Traspaso de datos entre diferentes pantallas

En el diagrama de navegación de ejemplo de la Figura 13 podemos observar que cuando el usuario inicia sesión el servidor devuelve el identificador del usuario que ha iniciado sesión para enviarlo más tarde. No será hasta las pantallas de Perfil o Cerrar sesión donde dicho identificador sea necesario. Para obtener dicho identificador en esas pantallas, habría que pasarlo como parámetro a la pantalla A que no lo usa por lo que el dato iría pasando por las diferentes pantallas hasta llegar a una pantalla donde sí que es usado. Esto es ineficiente, especialmente cuando tenemos diferentes pantallas, unas dentro de las otras.

Para solventar esto se usan los contextos, cuya función es extraer información que es común para diferentes pantallas o métodos y así evitar el envío de información a través de pantallas que no la necesitan. Cuando se recibe el dato que queremos almacenar, en el ejemplo sería el identificador del usuario, se notifica y se almacena en el contexto y cuando dicho dato necesita ser leído, se obtiene su valor y se utiliza donde haga falta.

A lo largo de la implementación de los requisitos de esta iteración realizamos control de errores, por ejemplo controlamos que cuando un usuario se registra rellene todos los campos, que ambas contraseñas coincidan y que tengan como mínimo 6 caracteres. Gracias a la potencia de Firebase, si un usuario utiliza un correo que ya había sido registrado anteriormente, mostramos el correspondiente mensaje de error al usuario para que facilite los datos correctos. También controlamos en el inicio de sesión que los campos estén rellenos y se muestran

diferentes mensajes de error dependiendo de si el usuario intenta iniciar sesión con un correo electrónico que no ha sido registrado o si indica una contraseña incorrecta.

En esta iteración implementamos la visualización de los carriles bici del conjunto de datos abiertos del Ayuntamiento de Málaga. Tras realizar las llamadas correspondientes para obtener dichos datos, se tratan los datos para mostrar cada carril bici donde corresponde y se muestran en el mapa. Durante la implementación de esta funcionalidad también obtenemos los carriles bici añadidos por los usuarios ya que estos últimos se muestran de un color diferente. Para solventar esto, obtenemos dos conjuntos de coordenadas diferentes, aquellas correspondientes a los datos abiertos del Ayto. de Málaga y las coordenadas añadidas por otros usuarios.

6.1.3. Iteración 2

Tras finalizar la Iteración 1 y validar el trabajo realizado, dio comienzo la Iteración 2. En la Iteración 1, además de revisar todo lo hecho, ajustamos el trabajo a realizar durante la Iteración 2. De este modo definimos los requisitos a implementar durante la nueva iteración teniendo en cuenta que en la Iteración 1 algunos requisitos no pudieron ser completados. Adicionalmente, añadimos nuevas funcionalidades para desarrollar a lo largo de la iteración.

Concretamente durante esta iteración completamos los siguientes requisitos:

- RF-USER-2.3: Un usuario podrá ver la información de su perfil.
- RF-USER-2.4: Un usuario podrá actualizar la información de su perfil.
- RF-USER-2.6: Un usuario podrá eliminar su perfil.
- RF-ROUT-1: Un usuario podrá iniciar una nueva ruta.
- RF-ROUT-1.1: Un usuario podrá descartar una ruta que ha iniciado.
- RF-ROUT-1.2: Un usuario podrá guardar una nueva ruta asignándole nombre, visibilidad (pública o privada) y nivel de dificultad (de 1 a 5).
- RF-BICI-2: Un usuario podrá añadir un nuevo tramo de carril bici.
- RF-BICI-3: Un usuario podrá descartar un nuevo tramo de carril bici.

Concluimos la mayoría de requisitos centrados en la gestión de usuarios. Durante el desarrollo del RF-USER-2.3 tuvimos dificultades a la hora de obtener los datos del servidor ya que en JavaScript la carga de datos se hace de forma asíncrona, mostrando antes de tiempo la capa visual del perfil. Para solventar esto, mientras los datos se están cargando del servidor se muestra una pantalla de carga, usando un *ActivityIndicator* hasta que tenemos los datos listos para mostrar. Por otro lado construir la interfaz para visualizar el perfil del usuario no fue tarea fácil ya que aún teníamos poca práctica con los estilos y era complicado fijar el encabezado y foto del perfil del usuario para poder hacer *scroll* por el resto de la pantalla. Lo conseguimos rediseñando la versión que hicimos al principio. En la Figura 14 podemos ver el resultado final bajando hasta el final de la pantalla mientras que el encabezado y foto de perfil permanecen fijos.

Figura 14: Interfaz definitiva del perfil del usuario

Igualmente tuvimos problemas a la hora de implementar el editar perfil del usuario ya que debíamos mostrar su lista actual de encabezados y avatares para que pudiese cambiarlos como un grupo de *radio buttons*. En este punto del desarrollo aún no conocíamos la biblioteca React Native Paper que sí tiene componentes predefinidos para incluir *radio buttons* en las

aplicaciones. Tras investigar cómo resolver este problema, nos decantamos por diseñar nuestro propio componente que permitiese renderizar un grupo de *radio buttons* de forma que sólo se pueda seleccionar un único elemento del grupo. Al tratarse de un componente aislado que se renderiza en la vista de la edición del perfil del usuario no es posible conocer el identificador del elemento seleccionado por el usuario. Gracias a los contextos, añadimos un nuevo contexto que almacenase los identificadores del avatar y encabezado seleccionados por el usuario.

Respecto al requisito de eliminar perfil, nuestra primera aproximación consistió en eliminar el perfil del usuario en Firebase usando la API que proporciona Firebase para JavaScript y tras eliminar el perfil llamar al servidor para que eliminase los datos del usuario en cuestión. Sin embargo, no funcionó debido a la asincronía de las llamadas en JavaScript.

Tras comentarlo con el tutor e investigar la API de Firebase, vimos que una buena solución sería delegar este proceso al servidor. De este modo, el servidor se encarga de eliminar la cuenta del usuario en Firebase gracias a que cuenta con permisos de administrador y también elimina los datos del usuario que estuvieran almacenados en la base de datos.

Fue en este punto de la iteración donde descubrimos la existencia de la biblioteca React Native Paper. Para la implementación de los requisitos relacionados con las rutas y los carriles bici, desde el punto de vista del diseño de la aplicación móvil incluimos componentes de la biblioteca React Native Paper. Concretamente usamos *Modal, Dialog* y *Switch*. Los primeros son útiles para mostrar ventanas emergentes al usuario y permitirle realizar diferentes acciones mientras que el último permite seleccionar estados exclusivos (o la ruta es privada o es pública). Empleamos estos componentes para pedir confirmación o informar al usuario a la hora de añadir una ruta/carril bici o descartarlos.

Para gestionar y procesar los datos de los requisitos que permiten añadir rutas y carriles bici, en el *frontend* diseñamos una función periódica que cada 5 segundos almacena en una lista la ubicación en tiempo real del usuario. Consideramos que este valor es el adecuado ya que habitualmente en bicicleta los usuarios suelen ir rápido y cada 5 segundos podríamos detectar un cambio significativo de la posición del usuario. Este y otros valores fijos que comentaremos más adelante los modularizamos para que fuesen fácilmente modificables en un futuro.

Cuando el usuario decide terminar la ruta o termina de añadir el carril bici, dicha lista se envía al servidor. Si, por el contrario, el usuario decide descartar el carril bici o la ruta los datos recopilados no son enviados al servidor, de este modo, minimizamos el número de peticiones

realizadas al servidor.

Cuando los datos llegan al servidor se calculan estadísticas (en el caso de las rutas) como el tiempo que ha tardado, la velocidad del usuario, el número de paradas que ha hecho y la distancia recorrida. En el caso de los carriles bici, únicamente se almacenan las coordenadas que son mostradas en el mapa.

El procesamiento de los datos se realiza de la siguiente manera (común para las rutas y los carriles bici): cuando el servidor recibe los datos recibe una lista de objetos que tiene este formato: [{ *latitude*: X, *longitude*: Y, *time*: Z },...] y se realiza un filtrado de los puntos para eliminar puntos cercanos entre sí. Se considera que 2 puntos son cercanos cuando su distancia es menor que 12 metros. Debemos tener presente que el usuario va en bicicleta y que en ocasiones la precisión del GPS puede ser baja y no detectar adecuadamente la posición actual del usuario. Para evitar esto y teniendo en cuenta los 5 segundos de la tarea periódica, creemos que es posible que el usuario haya avanzado 12 metros entre mediciones consecutivas.

Tras filtrar los puntos, si el usuario quería añadir una nueva ruta, con dichos puntos se calculan parámetros como el tiempo que ha tardado (contando las paradas realizadas y el tiempo que ha estado moviéndose), este tiempo es la suma de la diferencia de los sucesivos tiempos recibidos en las coordenadas filtradas. Para determinar el número de paradas que ha realizado el usuario durante la ruta, consideramos que el usuario está parado cuando los puntos son cercanos entre sí durante un periodo de 2 minutos o más ya que el usuario podría parar en un semáforo cuya duración fuese similar o por exceso de tráfico podría avanzar una distancia inferior a 12 metros en 2 minutos.

En el caso donde el usuario quiere añadir un nuevo tramo de carril bici, el procesamiento inicial es el que hemos explicado en párrafos anteriores si bien tenemos que distinguir diferentes casos: qué ocurre cuando el usuario añade un carril bici que forma parte del conjunto de datos abiertos o qué ocurre cuando el usuario añade un carril bici que ya ha sido registrado. Para solventar esta problemática diseñamos la siguiente solución:

– Comprobamos si los puntos filtrados están presentes o no en el conjunto de datos abiertos y en la base de datos.

Si no están presentes en el conjunto de datos abiertos y no están presentes en la base de datos se crea un nuevo carril bici. Si están presentes en el conjunto de datos abiertos, se comprueba cuál es más largo, si el que se quiere añadir o el que ya hay en el conjunto de datos abiertos.

Si el que se quiere añadir es el más largo y no está en la base de datos, se crea un nuevo carril bici.

– Comprobamos si están presentes o no en la base de datos.

Si no están presentes en la base de datos se crea un nuevo carril bici. Si están presentes en la base de datos, se comprueba cuál es más largo, si el que se quiere añadir o el que ya existe. Si el que se quiere añadir es el más largo y está en la base de datos, se añade un nuevo carril bici y se elimina el más corto.

6.1.4. Iteración 3

Tras la validación de la Iteración 2 comenzamos con la Iteración 3. Esta fue una de las iteraciones más largas donde implementamos una cantidad considerable de requisitos, cubriendo la mayoría de las funcionalidades.

En esta iteración implementamos los siguientes requisitos:

- RF-ROUT-2: Un usuario podrá realizar una ruta creada por otro usuario.
- RF-ROUT-3: Un usuario podrá editar el nombre o la visibilidad de una de sus rutas tras crearla.
- RF-ROUT-4: Un usuario podrá eliminar sus rutas creadas y/o completadas.
- RF-VSRT-1: Un usuario podrá ver todas las rutas registradas hasta la fecha, ordenadas de más reciente a más antigua.
- RF-VSRT-2: Un usuario podrá filtrar las rutas registradas según diferentes criterios: creador, (parte del) nombre, visibilidad, rutas más frecuentes y rutas creadas por mí.
- RF-VSRT-3: Un usuario podrá ver las rutas públicas creadas por otro usuario.
- RF-VSRT-4: Un usuario podrá ver información sobre la ruta seleccionada.
- RF-VSRT-4.1: Un usuario podrá ver el recorrido a seguir para completar una ruta.
- RF-VSRT-4.2: Un usuario podrá ver los usuarios más rápidos en realizar una ruta.
- RF-VSRT-5: Un usuario podrá ver desde su perfil las rutas que ha creado y las que ha completado.

- RF-USER-3: Un usuario podrá ver el perfil de otro usuario.
- RF-RANK-1: Un usuario podrá ver un *ranking* de puntos global formado por todos los usuarios.
- RF-RANK-2: Un usuario podrá ver su posición en el *ranking* de puntos global.

Durante el desarrollo del requisito RF-ROUT-2 tuvimos que definir un algoritmo para detectar cuándo consideramos que un usuario ha realizado una ruta creada por otro usuario. Para solventar este problema, diseñamos un algoritmo que sigue los siguientes pasos:

- En el servidor recibimos la misma lista de coordenadas que recibiríamos si estuviésemos añadiendo una nueva ruta.
- Filtramos los puntos de dicha lista eliminando puntos cercanos entre sí (si dos puntos están a menos de 12 metros, consideramos que son cercanos).
- Dado el conjunto de puntos filtrados y las coordenadas de la ruta que el usuario está realizando (ruta objetivo), calculamos el porcentaje de puntos cubiertos de la ruta objetivo. Si el porcentaje de puntos cubiertos es mayor que el 85 %, entonces consideramos que el usuario ha realizado la ruta. Consideramos que un punto está cubierto cuando la distancia entre el punto objetivo y el punto real es menor a 10 metros.
- Tras calcular el porcentaje, calculamos el tiempo que ha tardado en completar la ruta así como actualizamos los correspondientes campos de la base de datos para el usuario que completa la ruta y para la ruta original.

Respecto a la implementación del requisito RF-ROUT-4 hemos de mencionar que cuando el usuario quiere eliminar una ruta creada por él, le informamos de que la ruta pasará a ser propiedad del sistema, cambiando su visibilidad a pública, eliminando cualquier relación existente entre dicho usuario y la ruta. De este modo, el resto de usuarios pueden seguir realizando la ruta.

En la implementación de los requisitos RF-VSRT-1 y RF-VSRT-2 mostramos todas las rutas de la aplicación, ya sean públicas o privadas. Para ahorrar llamadas al servidor, los filtros los hemos implementado dentro de la aplicación móvil. Ya que tenemos una lista de rutas,

cuando el usuario selecciona un filtro, se filtran las rutas de la lista según el campo solicitado por el usuario, mejorando así el rendimiento de la aplicación. Como en todo filtro, hemos considerado el caso donde no hayan rutas que coincidan con los filtros seleccionados, en cuyo caso mostramos el correspondiente mensaje de información al usuario.

Si bien es cierto que los requisitos relacionados con la gestión de los usuarios los acabamos en iteraciones anteriores, en esta iteración concluimos con los requisitos RF-VSRT-3 y RF-USER-3. Dichos requisitos van unidos ya que es al mostrar a un usuario X el perfil del usuario Y cuando queremos mostrar las rutas públicas del usuario Y.

Para la implementación del requisito RF-VSRT-4 tuvimos que tener en cuenta la visibilidad de la ruta y el usuario creador de la misma. De este modo, si la ruta es pública el usuario puede ver la información. Si, por el contrario, es privada, hay que comprobar si el usuario que accede a dicha ruta es el creador de la misma, por lo que podrá verla. Si no es el creador, no tendrá acceso a la visualización de la ruta y mostramos el correspondiente mensaje informativo.

En cuanto a los requisitos relacionados con el *ranking*, en un primer acercamiento a la implementación final, comenzamos esbozando la interfaz gráfica con datos estáticos para poder decidir el diseño que seguiríamos. Una vez concluimos este primer paso, procedimos a crear las funciones necesarias en el servidor para obtener los usuarios del *ranking* ordenando sus puntos en orden descendente.

Para implementar algunas de las funcionalidades acordadas tuvimos que aprender a navegar de una ventana de un *Stack Navigator* a otra ventana ubicada dentro de un *Stack Navigator* diferente. Tras investigar, observamos que la solución es colocarse al principio del árbol de navegación mostrado en la Figura 5 e ir bajando por los diferentes niveles hasta llegar a la ventana objetivo.

Igualmente, a lo largo de esta iteración llevamos a cabo diferentes mejoras en la interfaz gráfica de la aplicación. A continuación, destacamos las más importantes:

- Añadimos encabezados a aquellas pantallas que aún no disponían de uno. Por ejemplo, la pantalla para cambiar la contraseña o la pantalla de editar el perfil.
- Cambiamos el color de los iconos de la barra de navegación inferior y los colores de los correspondientes botones para añadir un nuevo carril bici o una ruta.
- Incorporamos una pantalla de carga compuesta por un *ActivityIndicator* que mostramos

mientras obtenemos los datos del servidor.

6.1.5. Iteración 4

Esta última iteración fue más llevadera y sencilla ya que la funcionalidad pendiente de implementar era similar para muchos aspectos de la aplicación.

Concretamente, completamos las funcionalidades relativas a la gamificación y la tienda y un requisito del *ranking* que surgió en la revisión de la iteración anterior.

Al igual que en la Iteración 3, realizamos diversas mejoras que más tarde repasaremos. En cuanto a la funcionalidad que completamos en esta iteración tenemos:

- RF-STOR-1: Un usuario podrá ver los avatares, insignias y encabezados disponibles.
- RF-STOR-2: Un usuario podrá ver la información de un artículo (nombre, descripción, imagen y precio) desde la tienda.
- RF-STOR-3: Un usuario podrá comprar un avatar, insignia o encabezado con los puntos conseguidos hasta el momento.
- RF-RANK-3: Un usuario podrá ver la posición de otro usuario en el *ranking*.
- RF-GAME-1: Un usuario podrá conseguir puntos por registrarse.
- RF-GAME-2: Un usuario podrá conseguir puntos por iniciar sesión diariamente.
- RF-GAME-3: Un usuario podrá conseguir puntos creando nuevas rutas.
- RF-GAME-4: Un usuario podrá conseguir puntos completando una ruta creada por otro usuario.
- RF-GAME-5: Un usuario podrá conseguir puntos realizando una ruta de otro usuario frecuentemente.
- RF-GAME-6: Un usuario podrá conseguir puntos registrando carriles bici.

Los requisitos relacionados con la tienda los completamos sin problemas aparentes. Sólo destacar que cuando tuvimos que plantear la funcionalidad de comprar un artículo, en un primer acercamiento pensamos en usar dos llamadas al servidor. Una para conocer si el usuario

puede comprar el artículo y en caso afirmativo, otra petición para realizar la compra. Para disminuir el número de llamadas realizadas al servidor optamos por hacer una única llamada donde comprobamos si tiene los suficientes puntos para adquirir el artículo. En caso de tener puntos suficientes, añadimos el artículo a la colección del usuario y en caso contrario informamos al usuario de que sus puntos son insuficientes.

Durante la implementación de este requisito nos dimos cuenta de que si el usuario compra un artículo, los puntos que cuesta dicho artículo serían descontados de sus puntos actuales, provocando su descenso en el *ranking*. Esto no motivaría a los usuarios a comprar artículos por lo que tuvimos que solucionarlo introduciendo un nuevo campo en la colección donde almacenamos los datos del usuario que almacena los puntos disponibles para gastar en la tienda.

En la revisión de la Iteración 3 llegamos a la conclusión de que si un usuario quería ver la posición de otro usuario en el *ranking* fácilmente debería buscarlo en el *ranking* general. En caso de haber muchos usuarios esta tarea se complica por lo que acordamos añadir la posición del usuario en el *ranking* en la pantalla donde se visualiza el perfil del usuario. De este modo, la información se muestra de forma directa.

Respecto a la implementación de los requisitos RF-GAME-X, ésta se realizó sin mayores inconvenientes. Quizás cabe mencionar que durante la implementación del RF-GAME-2 tuvimos que realizar el código teniendo en cuenta diferentes casos:

- El usuario inicia sesión varias veces en un día pero sólo recibe puntos cuando pasan 24 horas desde la última vez que recibió puntos.
- El usuario pasa más de un día sin iniciar sesión, por lo que recibe los mismos puntos que recibiría si iniciase sesión por primera vez.
- El usuario inicia sesión una vez cada 24 horas en cuyo caso obtiene los puntos correspondientes.

Una de las dificultades que experimentamos fue que, al guardar el último acceso del usuario como un número en formato POSIX, dicho número contiene el día, la hora, el minuto y segundo exacto donde el usuario inició sesión. Esto hace que sea difícil darle puntos cada 24 horas exactamente. Supongamos que un usuario inicia sesión el día 01/07/2022 a las 17:30 y le asignamos los puntos correspondientes. Es complicado que el usuario vuelva a iniciar sesión el día

02/07/2022 a las 17:30 (justamente 24 horas después). Para solventar esto, lo que hemos hecho es dar un margen en los inicios de sesión. Esto es, si el usuario inició sesión el día 01/07/2022 a las 17:30, al día siguiente le daremos los puntos correspondientes si inicia sesión entre las 16:30 y las 18:30.

Para dar puntos a los usuarios por completar rutas y motivar a los usuarios a seguir completando rutas seguimos el siguiente patrón. Damos puntos a todos los usuarios sin importar si es o no el usuario más rápido en completar la ruta. Estos puntos los damos según el tiempo que tarda el usuario. Cuando un usuario tarda más que el resto en completar la ruta, recibe menos puntos. Por el contrario, si tarda poco en completar la ruta, recibirá más puntos.

En la Figura 15 podemos ver la cantidad de puntos que damos a los usuarios por realizar diferentes acciones.

Acción	Puntos
Registro	100
Añadir un tramo de carril bici	170
Añadir una nueva ruta	200
Completar una ruta	Entre 150 y 80 puntos
Realizar una ruta frecuentemente	120
Iniciar sesión día 1	5
Iniciar sesión día 2	10
Iniciar sesión día 3	15
Iniciar sesión día 4	20
Iniciar sesión día 5	30
Iniciar sesión día 6	40
Iniciar sesión día 7	50

Figura 15: Puntos que obtienen los usuarios realizando diferentes acciones

Tras implementar los requisitos relacionados con la gamificación asignamos los puntos a los artículos de la tienda. Dicha asignación la hemos realizado teniendo en cuenta un número mínimo de acciones que tendría que hacer el usuario para poder conseguir sus primeros artículos. No obstante, cuando un usuario se registra en GoSykel recibe un encabezado, una insignia y un avatar gratuitos. La asignación de puntos que hemos realizado se muestra en la Figura 16.

Al final de esta iteración realizamos diversas mejoras a la aplicación. Entre ellas, además de generar datos de prueba que más tarde utilizaremos en la grabación del vídeo demostración que también realizamos al final de esta iteración, realizamos una mejora en términos de seguridad,

Encabezados

Nombre	Puntos
Encabezado ciclistas juniors	0
Verticalidad absoluta	300
Amantes del azul	500
Círculos everywhere	700
Burbujas de color	1000

Avatares

Nombre	Puntos
Nivel iniciación	0
Ciclista campestre	250
Bicicleta primaveral	500
Ciclistas al atardecer	600
Avatar cosmopolita	800

Insignias

Nombre	Puntos
Recién llegados	0
Comenzamos a entendernos	200
Nivel intermedio	500
A un paso del cielo	600
Usuarios top	700

Figura 16: Puntos de los artículos de la tienda

incorporando un mecanismo de *tokens* JWT que se intercambian entre la aplicación móvil y el servidor en las sucesivas consultas realizadas.

Mecanismo de *tokens JWT (JSON Web Token)*

Para implementar este mecanismo, en primer lugar nos informamos acerca de él y buscamos información relativa a su implementación. En primer lugar, comenzaremos viendo la estructura de un *token* JWT, que está formado por 3 partes bien diferenciadas: HHHHHH.PPPPPP.SSSSSS donde la primera componente es el *header* que contiene los metadatos, el tipo de *token* (en este caso JWT) y el algoritmo de *hashing* que usará. La segunda parte del *token* es el *payload* que contiene la información que es de nuestro interés, como por ejemplo el identificador del usuario y la fecha y hora en que inició sesión. En el extremo final del *token* tenemos la firma (*signature*) que se compone de 4 partes fundamentalmente: un encabezado en Base64, un *payload* en Base64, un secreto y un algoritmo criptográfico de encriptado. En nuestro caso hemos usado el algoritmo HMAC con SHA-256.

Tras obtener información sobre dicho mecanismo comenzamos a implementar una variante más básica del mecanismo original adaptada a nuestras necesidades. En la Figura 17 podemos ver un diagrama que nos ayuda a tener una idea de cómo es el intercambio de *tokens* entre la aplicación móvil y el servidor.

Como podemos ver, cuando el usuario inicia sesión o se registra, el servidor realiza las validaciones pertinentes y genera un *token* para dicho usuario que almacena y envía a la aplicación móvil donde también será almacenado. Cada vez que el usuario necesite extraer datos del servidor, éste validará el *token* enviado por el usuario en la petición. En caso de que el *token* sea válido, el servidor enviará los datos solicitados. Si no lo es, enviará el correspondiente error al usuario. Cuando el usuario quiere cerrar su sesión o eliminar su perfil, se valida el *token* enviado y si es válido éste se eliminará del servidor así como de la aplicación móvil. En caso de no serlo, se envía el correspondiente mensaje al usuario.

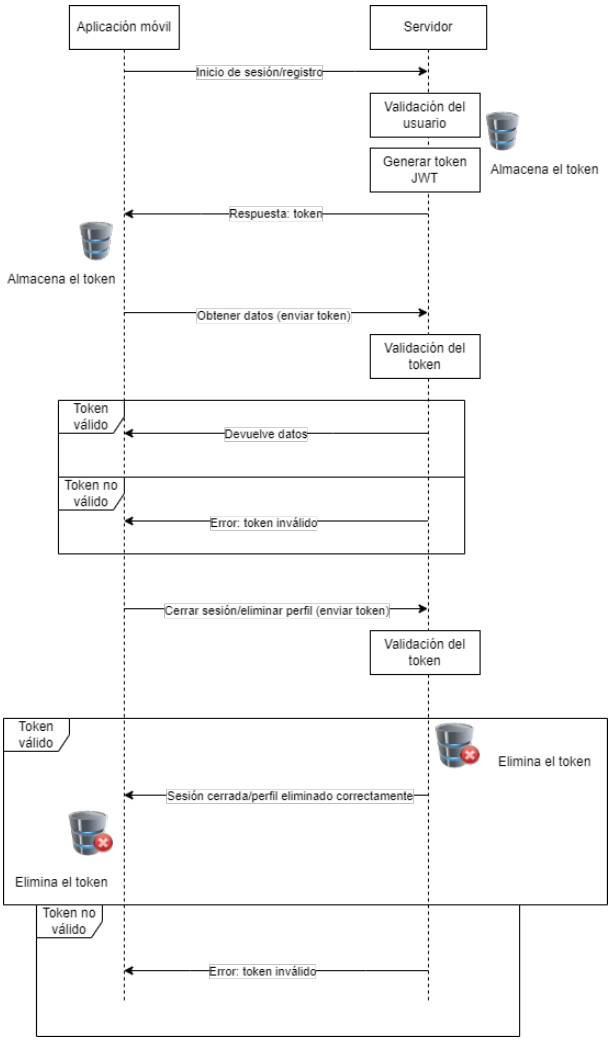


Figura 17: Intercambio de *tokens* entre la aplicación móvil y el servidor

Finalmente, hemos de mencionar que las comunicaciones entre la aplicación móvil y el servidor se hacen utilizando el protocolo HTTPS por lo que podemos garantizar que la información viaja de forma segura a través de Internet.

6.2. Pruebas realizadas

En esta sección comentaremos las pruebas que hemos realizado a lo largo del proceso de desarrollo. Las pruebas abarcan distintos ámbitos, desde pruebas a la API REST usando Postman hasta pruebas en dispositivos reales con usuarios reales mediante un *test* de usabilidad.

En primer lugar, comenzaremos con las pruebas de la API REST que hemos realizado con Postman.

6.2.1. Pruebas de la API REST usando Postman

Esta fase de pruebas de la API REST la hemos realizado durante todo el desarrollo y por tanto no ha sido necesario planificar un periodo de pruebas de la aplicación. A medida que íbamos desarrollando las funcionalidades del servidor, nos íbamos ayudando de Postman para comprobar que las salidas eran correctas cuando ejecutábamos el servidor en local. Cuando nos aseguramos de que la salida es la esperada, actualizamos los nuevos cambios en el servidor de Heroku y en cuanto se despliegan comprobamos nuevamente con Postman que la salida es la esperada.

A medida que íbamos probando las diferentes funcionalidades, íbamos creando diferentes peticiones en Postman y añadiendo una descripción lo suficientemente completa para saber lo que debemos esperar en la salida.

Por ejemplo, en las figuras 18 y 19 podemos ver uno de los casos de prueba que realizamos durante el desarrollo. En él, comprobamos que las rutas se registran adecuadamente. En este caso, el servidor devuelve los puntos que ha obtenido el usuario por registrar una nueva ruta y los puntos que ha acumulado hasta el momento. Por el contrario, cuando queremos registrar una ruta cuyos puntos son cercanos entre sí el servidor devuelve un mensaje de error al usuario.

6.2.2. Pruebas en dispositivos reales

Este tipo de pruebas es el más amplio ya que abarca diferentes pruebas: probar la aplicación móvil en un dispositivo real durante el desarrollo para asegurarnos de que el contenido se ve adecuadamente, probar la aplicación móvil en la calle añadiendo rutas reales una vez finalizado el desarrollo de la misma o probar la aplicación en otros dispositivos diferentes al dispositivo donde se ha desarrollado para asegurarnos de que los contenidos se ven bien.

A lo largo del desarrollo y al final del mismo hemos realizado las pruebas mencionadas anteriormente. Las pruebas realizadas en un teléfono real durante el desarrollo nos han servido para comprobar que, por ejemplo, los mensajes en caso de no haber rutas se muestran correctamente o que se muestran los mensajes de error en caso de haberlos.

En algunos puntos clave del desarrollo probamos en la calle el algoritmo que diseñamos para registrar nuevas rutas, adaptando los parámetros del mismo (cercanía entre puntos) al rit-

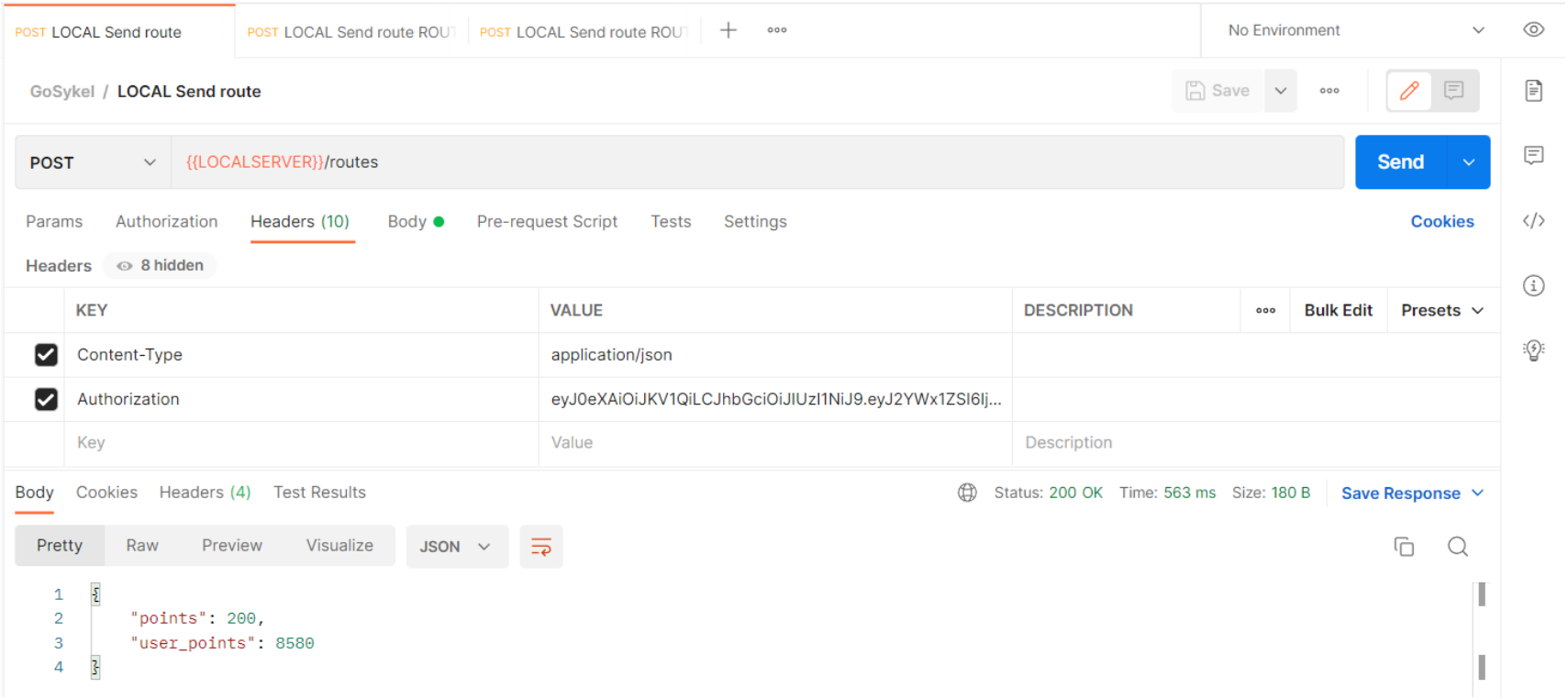


Figura 18: Caso de éxito, la ruta se añade correctamente

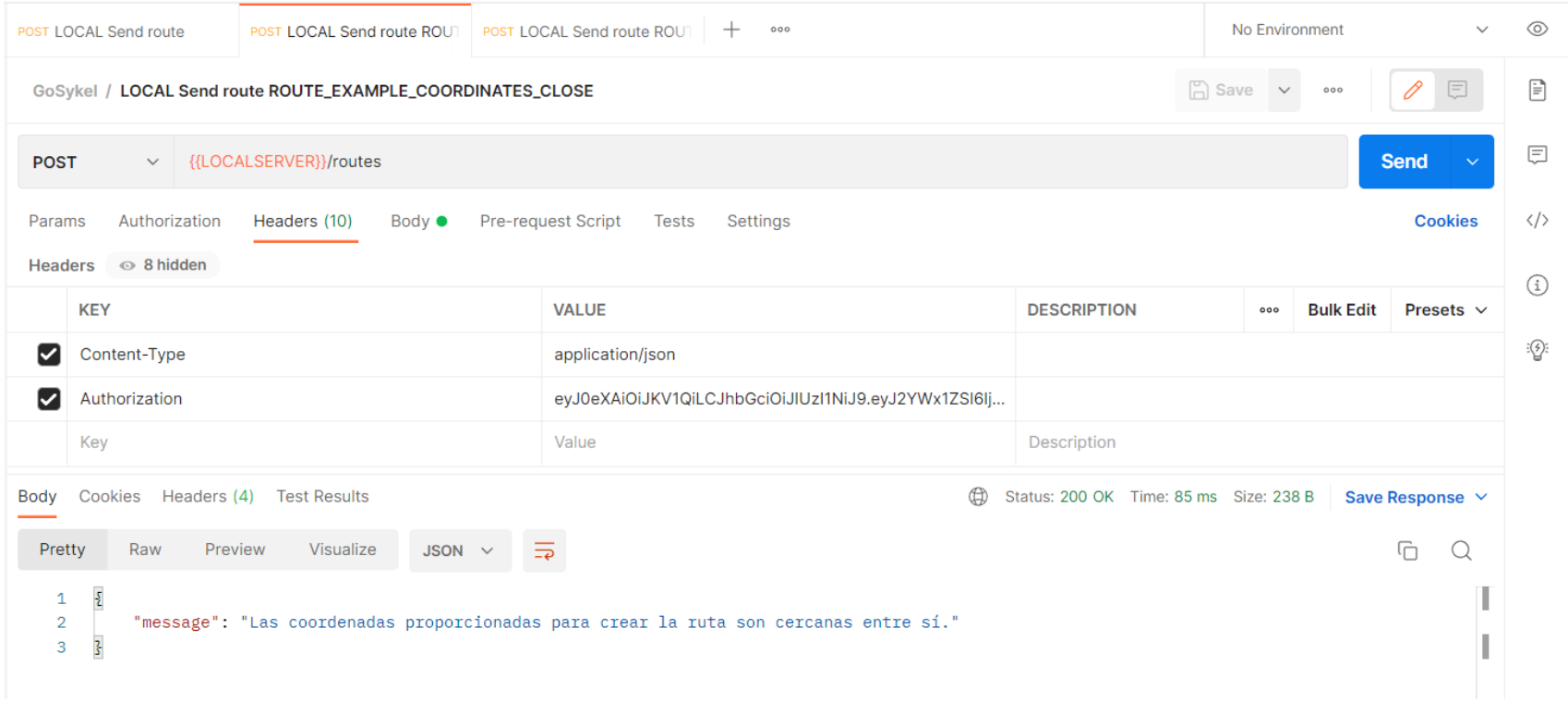


Figura 19: Caso de error: las coordenadas son cercanas entre sí

mo que llevaría una persona andando. De este modo comprobamos que el algoritmo calculaba las estadísticas como se esperaba.

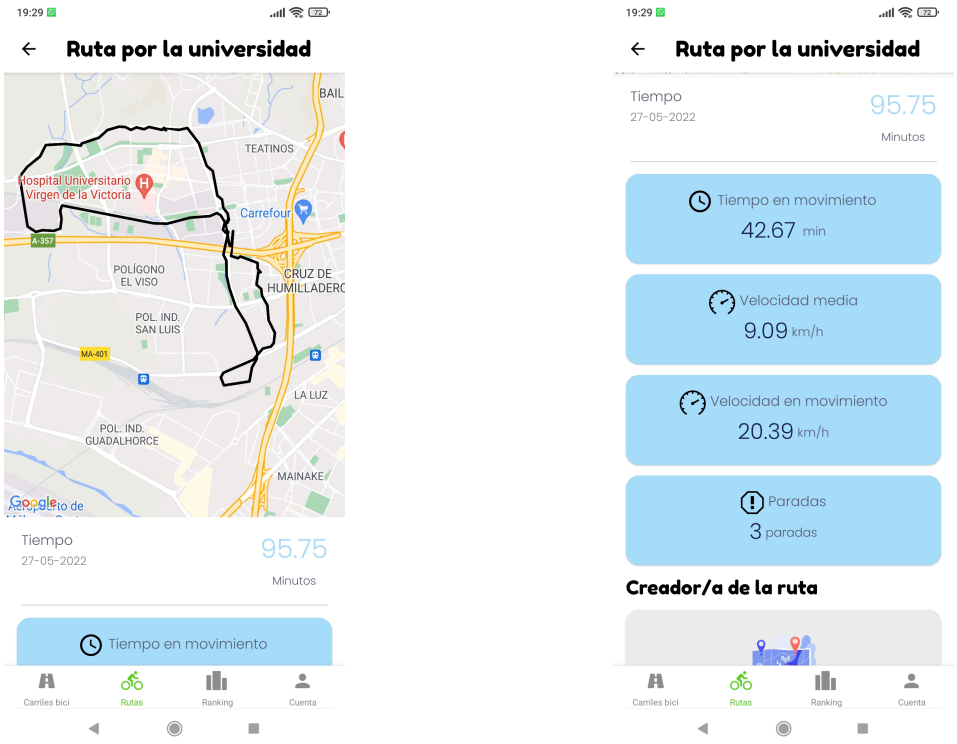
Una vez concluido el desarrollo, en la última iteración, realizamos una ruta en bicicleta en exterior para comprobar que se registraban correctamente y que la información era correcta. Los resultados obtenidos tras añadir una nueva ruta realizada por los alrededores del Campus de Teatinos los podemos ver en la Figura 20.

Al principio planteamos esta ruta de prueba como una ruta que contendría 2 paradas y que no sería muy larga. En el resultado muestra que contiene 3 paradas y ello se debe a que tardamos un poco más de 2 minutos en ponernos en marcha. Adicionalmente, durante la ruta hicimos las 2 paradas que habían previstas, obteniendo un resultado favorable en este aspecto.

En cuanto a la distancia, aunque inicialmente habíamos previsto una ruta más corta, durante el desarrollo de la misma decidimos continuar hasta realizar una ruta de 14 kilómetros aproximadamente. Tras almacenar la ruta y comprobar los datos recogidos, esta distancia tiene sentido ya que el recorrido es de una distancia similar.

Si nos fijamos en los puntos recogidos y cómo se muestran en el mapa vemos que el algoritmo de filtrado funciona generalmente bien, mostrando la mayoría de la ruta sin incidentes.

Por otra parte, el tiempo total que tardamos en completar la ruta es correcto ya que paralelamente al registro en la aplicación, registramos la actividad en un reloj inteligente de cara a conocer cuánto tiempo hemos estado en movimiento realmente.



(a) Visualización del recorrido realizado

(b) Visualización del tiempo, velocidad y paradas

Figura 20: Estadísticas de la ruta de prueba en el Campus de Teatinos

Para concluir las pruebas, pedimos a diferentes personas que probasen nuestra aplicación en sus teléfonos y en todos ellos la visualización de los datos era correcta. Podían ver todas las pantallas y la información mostrada en cada una de ellas sin dificultades. Las pantallas de la mayoría de los dispositivos donde realizamos estas pruebas tenían una dimensión aproximada de 6 pulgadas.

6.2.3. *Test* de usabilidad

Cuando tuvimos la aplicación final, decidimos ponerla en manos de usuarios reales ajenos al desarrollo para conocer sus opiniones de cómo era el diseño y la navegación en la aplicación. Para ello, realizamos un *test* de usabilidad a 11 usuarios.

Procedimiento del *test* de usabilidad

Para realizar esta prueba con usuarios reales seguimos diferentes fases. La primera de ellas es el diseño de la prueba a realizar a los usuarios. Tenemos que decidir qué queremos probar dentro de la aplicación y cómo lo haremos.

En nuestro caso queríamos comprobar que la navegación, el diseño y el lenguaje de la aplicación son correctos. Para conocer la opinión de los usuarios, posteriormente, diseñamos un formulario de Google que usamos durante el *test*.

El desarrollo de la prueba consistió en sentar al usuario final frente al evaluador. El evaluador comienza poniendo en contexto al usuario acerca del origen de GoSykel y el propósito de la prueba. El evaluador proporciona unas instrucciones básicas que debe seguir el usuario durante la prueba: realizar las acciones solicitadas por el evaluador, parar cuando el evaluador se lo solicite y expresar en voz alta sus pensamientos a medida que navega por la aplicación.

Tras investigar en Internet las fases de un *test* de usabilidad acordamos dividirlo en 3 fases bien diferenciadas: *pre-test*, *test* y *post-test*.

El objetivo de la fase *pre-test* es conocer las impresiones del usuario sobre el uso de la bicicleta y aplicaciones relacionadas con el deporte durante el uso de la misma. Para cumplir este objetivo, realizamos las siguientes preguntas al usuario:

- En una escala del 1 al 5 (1 - poco seguro, 5 - muy seguro), ¿cómo de seguro te sentirías al usar tu teléfono móvil para hacer rutas en bicicleta por la ciudad suponiendo que cumples con las medidas de seguridad oportunas (por ejemplo, el teléfono móvil está colocado sobre un soporte para bicicletas)?
- ¿Tienes bicicleta?
- En caso de tener bicicleta, en una escala del 1 al 5 (1 - nunca, 5 - siempre), ¿con cuánta frecuencia usas tu bicicleta?
- ¿Cuándo fue la última vez que usaste tu bicicleta?
- ¿Usas alguna aplicación cuando coges la bicicleta? Algunas aplicaciones como Strava miden datos del ejercicio que has hecho o los kilómetros que has recorrido.
- Si seleccionaste “Sí” en la pregunta anterior, ¿Podrías describir cómo ha sido tu experiencia con las aplicaciones que hayas usado?

La siguiente fase (*test*) del formulario se cumplimentaba tras realizar 3 tareas diferentes dentro de la aplicación solicitadas por el evaluador del *test*. Las tareas solicitadas por el evaluador fueron: ver la información de una ruta y el perfil de otro usuario; ver las rutas creadas por el

usuario que ha iniciado sesión; editar la información (nombre y contraseña) del usuario que ha iniciado sesión. También preguntamos a los usuarios dónde esperarían encontrar la posición de un usuario en el *ranking*.

Cuando realizaron las tareas y respondieron a la pregunta, rellenaron la fase *test* del formulario que contiene las siguientes preguntas:

- En una escala del 1 al 5 (1 - muy mala, 5 - excelente), ¿cómo calificas la experiencia de usar GoSykel para encontrar la posición de un usuario en el *ranking*?
- ¿Crees que el lenguaje utilizado en la aplicación es bueno y es comprendido por cualquier usuario?
- En una escala del 1 al 5 (1 - muy difícil, 5 - muy fácil), ¿cómo ha sido la navegación por la aplicación?
- En una escala del 1 al 5 (1 - muy malo, 5 - muy bueno), ¿cuál es tu opinión sobre el diseño?

Finalmente se realiza la fase *post-test* para conocer las impresiones finales del usuario acerca del producto. En esta fase realizamos las siguientes preguntas:

- En una escala del 1 al 5 (1 - muy mala, 5 - excelente), ¿cómo describirías la experiencia general con el producto?
- En una escala del 1 al 5 (1 - poco probable, 5 - muy probable), ¿cómo de probable es que recomiendes GoSykel a un amigo/a?
- ¿Con cuanta frecuencia usarías GoSykel si tuvieras bicicleta?

Análisis y conclusiones de los resultados obtenidos

Una vez concluida la fase de evaluación junto a los usuarios, es momento de analizar los resultados obtenidos en la encuesta. Hemos de tener en cuenta que la población a la que hemos realizado la prueba es joven (entre 18 y 22 años), por lo que no todos están igual de experimentados en el uso de la bicicleta.

Tras poner a los usuarios en contexto, la gran mayoría se sentiría seguro usando el teléfono móvil para realizar rutas en bicicleta siempre que éste cuente con las medidas de seguridad

oportunas. Del mismo modo, 8 de los 11 usuarios encuestados disponen de bicicleta y la mayoría usa su bicicleta ocasionalmente, tanto es así que la última vez que la usaron fue hace más de dos semanas. Analizando el uso de aplicaciones como Strava para registrar la actividad deportiva, observamos que la gran mayoría de encuestados no usan este tipo de aplicaciones.

Una vez concluidas las tareas solicitadas durante la prueba, casi todos los usuarios coinciden en que la experiencia de encontrar la posición de un usuario en el *ranking* ha sido fácil, la gran mayoría está de acuerdo en que la experiencia de usuario en este aspecto ha sido excelente así como todos concuerdan en que el lenguaje de la aplicación es correcto. Respecto a la navegación por la aplicación la mayoría está de acuerdo en que es fácil por lo que es posible encontrar la información sin dificultades. En cuanto al diseño, todos coinciden en que es muy bueno; esto nos da a entender que la interfaz de usuario está bien diseñada y que los diferentes elementos interactivos son intuitivos.

Finalmente, la experiencia general con el producto ha sido bastante buena, más de un 80 % de los usuarios recomendarían GoSykel a un amigo y generalmente usarían la aplicación ocasionalmente. Esto último está alineado con el objetivo de la aplicación: hacer un producto que ocasionalmente puedan usar los usuarios cuando van en bicicleta.

Tras analizar los resultados de las pruebas realizadas concluimos que el propósito general de la aplicación se cumple adecuadamente: gracias al diseño, es un producto sencillo de utilizar por los usuarios, la información la hemos expresado de forma comprensible y la frecuencia de uso de la aplicación está en línea con lo que habíamos planeado al principio de este trabajo.

7

Conclusiones y trabajo futuro

Una vez finalizado el proyecto debemos pensar en las diferentes fases realizadas a lo largo del mismo, extraer conclusiones de la experiencia y determinar posibles acciones a realizar en el futuro.

7.1. Conclusiones

A modo de conclusión general podemos destacar que este trabajo cumple con los requisitos definidos inicialmente.

Si bien es cierto que al principio del desarrollo teníamos cierta inquietud acerca de cómo sería la implementación del tratamiento de los datos tras recoger las coordenadas y la obtención de estadísticas, consideramos que la solución que proponemos es adecuada para el propósito de la aplicación. Igualmente, hemos procurado que en este sentido el almacenamiento y tratamiento de los datos sea lo más eficiente posible, minimizando las llamadas al servidor para reducir los tiempos de espera.

Un reto a nivel personal que nos planteamos al principio era aprender nuevas tecnologías de *frontend* que sirviesen para desarrollar aplicaciones móviles multiplataforma. A pesar de que al principio no contábamos con la experiencia suficiente en JavaScript, consideramos que aprender estas tecnologías ha sido beneficioso ya que muchas empresas las utilizan y dan un valor añadido al currículum.

A lo largo del trabajo podemos ver cómo se han empleado los conocimientos adquiridos durante estos 4 años. Desde la gestión de proyectos hasta las pruebas pasando por el modelado y diseño de la estructura de la aplicación o la seguridad de la misma. La estructura de la aplicación está bien diferenciada en dos partes: cliente y servidor; el uso de componentes modulares

facilita el desarrollo y hace que el código sea legible por otras personas. Igualmente, el uso de Scrum adaptado al proyecto ha sido todo un éxito ya que nos ha permitido concluir dentro de los plazos acordados y ha evitado que nos quedemos estancados.

Si bien es cierto que durante el desarrollo hemos encontrado dificultades de diferentes tipos debido al desconocimiento de algunas de las tecnologías, las búsquedas en Internet, las consultas con el tutor y con otros compañeros han contribuido a que en ningún momento nos hayamos bloqueado más de 3 o 4 días.

Para finalizar, este trabajo nos ha ayudado a crecer como profesionales, a conocer cómo plantear un proyecto desde cero y a seguir aprendiendo acerca del proceso de desarrollo de software. Una de las competencias fundamentales dentro de los proyectos software es la de aprender a resolver problemas y ser creativos. Creemos que durante este trabajo hemos adquirido estas competencias así como hemos aprendido a redactar la documentación necesaria para que otras personas puedan ejecutar nuestro trabajo.

7.2. Trabajo futuro

Generalmente la aplicación es completa y podría ser puesta a disposición de los usuarios sin ningún inconveniente. No obstante, siempre podemos mejorar y aumentar el valor del producto. A continuación, proponemos varios aspectos en los que podríamos mejorar la aplicación:

- **Pausar la toma de datos durante la creación de nuevas rutas o carriles bici:** cuando los usuarios van en bicicleta hay algunos momentos donde se tienen que parar debido a diferentes motivos: tráfico o semáforos en rojo para las bicicletas, por ejemplo. Para evitar tomar datos en estas circunstancias, una posible mejora sería añadir un botón de pausar que permita detener la toma de datos actual y retomarla más tarde cuando el usuario esté listo para continuar.

- **Añadir un buscador de usuarios en el *ranking*:** cuando el número de usuarios de la aplicación incremente, buscar un usuario en el *ranking* puede ser una tarea complicada. Una buena solución es añadir un buscador que permita encontrar a los usuarios y visualizar el número de puntos que tienen y la posición en que se encuentran.

- **Refinamiento en la gestión de los datos de GPS para mejorar su visualización y en el cálculo de estadísticas:** si bien es cierto que ahora mismo contamos con un algoritmo de filtrado que cumple el propósito de la aplicación, una posible mejora sería incluir mecanismos más precisos para comprobar las rutas que se están añadiendo. Una posible solución sería utilizar la API de Google que permite conocer la calle donde se encuentra el usuario a través de las coordenadas.

Ya que la aplicación depende de diferentes servicios como son Firebase, Expo, React Native, Python y diferentes librerías, hemos de considerar otra línea adicional relacionada con el mantenimiento de la aplicación. Estas tecnologías están en constante mejora y en algún momento podríamos detectar incompatibilidades entre versiones que deberemos solventar adaptando la aplicación para que sea compatible.

Referencias

[1] *¿Qué es la gamificación y cuáles son sus objetivos? | EDUCACIÓN 3.0*. es. Section: Noticias. Ago. de 2019. URL: https://www.educaciontrespuntocero.com/noticias/gamificacion-que-es-objetivos/ (visitado 23-06-2022).

[2] *¿Qué es Scrum?* en. URL: https://www.scrum.org/resources/blog/que-es-scrum (visitado 04-06-2022).

[3] *AppLoading*. en. URL: https://docs.expo.dev/versions/latest/sdk/app-loading (visitado 03-06-2022).

[4] *Build software better, together*. en. URL: https://github.com (visitado 03-06-2022).

[5] *Conjuntos de datos - Datos abiertos Ayto. Málaga*. URL: https://datosabiertos.malaga.eu/dataset (visitado 18-06-2022).

[6] Datensen. *Moon Modeler - Draw ER Diagrams for your Data Models*. en. URL: https://www.datensen.com/data-modeling/moon-modeler-for-databases.html (visitado 03-06-2022).

[7] *Expo Documentation*. en. URL: https://docs.expo.dev/ (visitado 03-06-2022).

[8] *expo-google-fonts*. original-date: 2020-05-02T13:26:31Z. Mayo de 2022. URL: https://github.com/expo/google-fonts (visitado 03-06-2022).

[9] *Firebase*. es-419. URL: https://firebase.google.com/?hl=es-419 (visitado 03-06-2022).

[10] *Firebase Admin Python SDK*. URL: https://firebaseopensource.com/projects/firebase/firebase-admin-python/ (visitado 03-06-2022).

[11] *Git*. URL: https://git-scm.com/ (visitado 03-06-2022).

[12] *Google Docs: editor de documentos online gratuito | Google Workspace*. es. URL: https://www.facebook.com/GoogleDocs/ (visitado 03-06-2022).

[13] *Google Forms: solución gratuita para crear formularios online | Google Workspace*. es. URL: https://www.facebook.com/GoogleDocs/ (visitado 03-06-2022).

[14] *Home · React Native Paper*. en. URL: https://callstack.github.io/react-native-paper/ (visitado 03-06-2022).

[15] *Image and Video Upload, Storage, Optimization and CDN.* URL: https://cloudinary.com/ (visitado 03-06-2022).

[16] *JavaScript | MDN.* es. URL: https://developer.mozilla.org/es/docs/Web/JavaScript (visitado 03-06-2022).

[17] *La mejor alternativa al transporte público.* es. URL: https://www.fundacionaquae.org/wiki/cinco-ventajas-utilizar-la-bicicleta-medio-transporte/ (visitado 23-06-2022).

[18] *Location.* en. URL: https://docs.expo.dev/versions/latest/sdk/location (visitado 03-06-2022).

[19] *Marvel - The design platform for digital products. Get started for free.* URL: https://marvelapp.com/ (visitado 14-06-2022).

[20] *MongoDB Compass.* es. URL: https://www.mongodb.com/es/products/compass (visitado 03-06-2022).

[21] *MongoDB: The Application Data Platform.* en-us. URL: https://www.mongodb.com (visitado 02-06-2022).

[22] United Nations. *Datos y cifras | Naciones Unidas.* es. Publisher: United Nations. URL: https://www.un.org/es/actnow/facts-and-figures (visitado 23-06-2022).

[23] *Overleaf, Editor de LaTeX online.* es. URL: https://www.overleaf.com (visitado 03-06-2022).

[24] *Plataforma de almacenamiento personal en la nube y uso compartido de archivos - Google.* es. URL: https://www.google.com/intl/es/drive/ (visitado 03-06-2022).

[25] *Postman API Platform | Sign Up for Free.* URL: https://www.postman.com/ (visitado 03-06-2022).

[26] *PyMongo 4.1.1 Documentation — PyMongo 4.1.1 documentation.* URL: https://pymongo.readthedocs.io/en/stable/ (visitado 03-06-2022).

[27] *React Native · Learn once, write anywhere.* en. URL: https://reactnative.dev/ (visitado 03-06-2022).

[28] *React Navigation | React Navigation*. en. URL: https://reactnavigation.org// (visitado 03-06-2022).

[29] *react-native-maps*. original-date: 2015-12-29T19:54:20Z. Jun. de 2022. URL: https://github.com/react-native-maps/react-native-maps (visitado 03-06-2022).

[30] *Telegram – una nueva era de mensajería*. URL: https://telegram.org/?setln=es (visitado 03-06-2022).

[31] *Trello*. URL: https://trello.com (visitado 03-06-2022).

[32] *Visual Studio Code - Code Editing. Redefined*. en. URL: https://code.visualstudio.com/ (visitado 03-06-2022).

[33] *Welcome to Flask — Flask Documentation (2.1.x)*. URL: https://flask.palletsprojects.com/en/2.1.x/ (visitado 02-06-2022).

[34] *Welcome to GeoPy's documentation! — GeoPy 2.2.0 documentation*. URL: https://geopy.readthedocs.io/en/stable/ (visitado 03-06-2022).

[35] *Welcome to PyJWT — PyJWT 2.4.0 documentation*. URL: https://pyjwt.readthedocs.io/en/stable/ (visitado 03-06-2022).

[36] *Welcome to Python.org*. en. URL: https://www.python.org/about/ (visitado 02-06-2022).

[37] *What is Heroku | Heroku*. en. URL: https://www.heroku.com/what (visitado 03-06-2022).

Apéndice A
Manual de usuario

A.1. Introducción

El objetivo de este apéndice es facilitar el uso de la aplicación a los nuevos usuarios, indicando los pasos a seguir para realizar cada tarea y resolviendo las dudas que puedan surgir.

Suponemos que previamente ha instalado la aplicación en su teléfono móvil, concretamente en un dispositivo Android.

En caso de tener dudas o sugerencias, por favor, envíenos un correo electrónico a esta dirección.

A.2. ¿Cómo acceder a GoSykel?

Cuando inicia la aplicación verá la pantalla de bienvenida (Figura 21) donde aparecerán dos opciones, iniciar sesión o registrarse. Ya que no tiene cuenta, si pulsa sobre la opción "¿Aún no tienes cuenta? Regístrate" le llevará a la pantalla de registro (Figura 22) donde debe rellenar sus datos. En cuanto a los datos que debe proporcionar se encuentran un nombre de usuario, un correo electrónico válido y que no estuviera registrado anteriormente y una contraseña que debe tener como mínimo 6 caracteres. Tras rellenar la información, pulse el botón "¡A pedalear!" y habrá completado su registro con éxito.

Si, por el contrario, ya está registrado, en la pantalla de bienvenida debe pulsar "Iniciar sesión" y le llevará a la pantalla de inicio de sesión (Figura 23) donde deberá cumplimentar sus datos de inicio de sesión. Cuando los rellene, pulse "Iniciar sesión" y accederá a GoSykel.

En caso de que haya olvidado su contraseña antes de acceder a GoSykel y ya estaba registrado, diríjase desde la pantalla de bienvenida a la pantalla de inicio de sesión y pulse sobre "¿Has olvidado tu contraseña?". A continuación verá una pantalla donde introducir su correo electrónico para cambiar la contraseña (Figura 24). Cuando pulse sobre "Cambiar contraseña" se le enviará un correo electrónico con los pasos a seguir para cambiar su contraseña. Una vez la actualice, deberá iniciar sesión con la nueva contraseña. Cualquier fallo durante los pasos

anteriores será notificado con el correspondiente mensaje informativo.

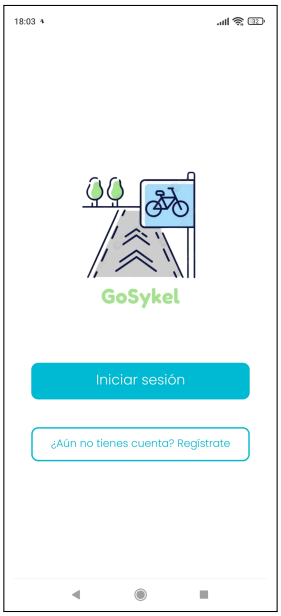


Figura 21: Pantalla de bienvenida

Figura 22: Pantalla de registro de usuarios

Figura 23: Pantalla de inicio de sesión

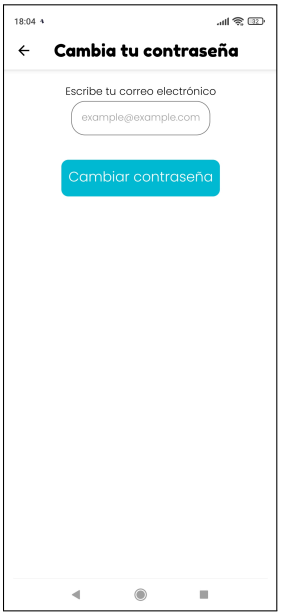


Figura 24: Cambiar la contraseña

A.3. Principales secciones de GoSykel

Cuando inicia sesión o se registra lo primero que verá es los carriles bici disponibles en Málaga capital (Figura 25). Si en la barra de navegación inferior pulsa sobre el icono “Rutas” se

moverá a otra pantalla (Figura 26) donde puede ver las rutas disponibles en la aplicación. Otra sección que puede consultar desde la barra de navegación inferior es la sección del *ranking* de usuarios (Figura 27). Puede llegar a ella pulsando sobre *"Ranking"*. Para gestionar su cuenta, sólo debe pulsar el icono "Cuenta" ubicado en la barra de navegación inferior y verá las diferentes opciones para gestionar su cuenta (Figura 28). Cualquier error que ocurra en estas secciones será mostrado con el correspondiente mensaje informativo.

Figura 25: Carriles bici en Málaga capital

Figura 26: Rutas registradas por los usuarios

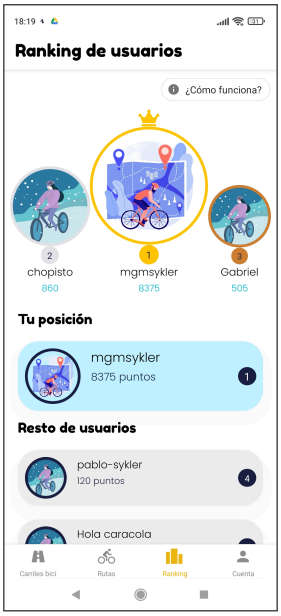


Figura 27: *Ranking* de usuarios de GoSykel

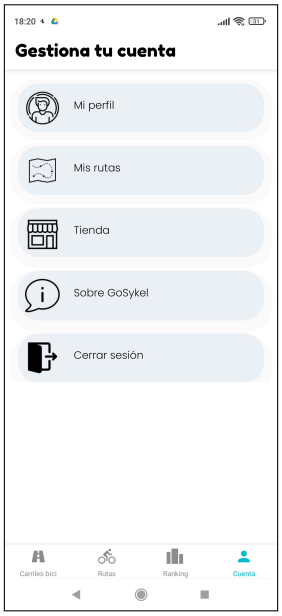


Figura 28: Gestionar su cuenta

A.4. Sección: Carriles bici

Figura 29: Añadir un nuevo carril bici

En esta sección, además de visualizar los carriles bici, puede añadir un nuevo tramo de carril bici. Para ello, debe pulsar el botón "+ CARRIL BICI" que puede ver en la Figura 25. Tras pulsar dicho botón, verá una pantalla con un mapa que mostrará su ubicación en tiempo real (Figura 29). Usted empezará a avanzar con su bicicleta por el tramo de carril bici y cuando haya llegado el final del mismo pulsará el botón "Guardar" donde le pedirán su confirmación (Figura 30). En cambio, si pulsa sobre la flecha situada en la esquina superior izquierda, descartará el tramo de carril bici (Figura 31) y tendrá que dar su confirmación. En ambos casos, tanto si decide añadirlo como si lo descarta, será dirigido al mapa de carriles bici.

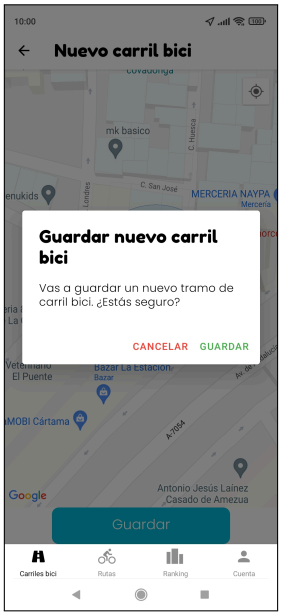


Figura 30: Guardar el carril bici

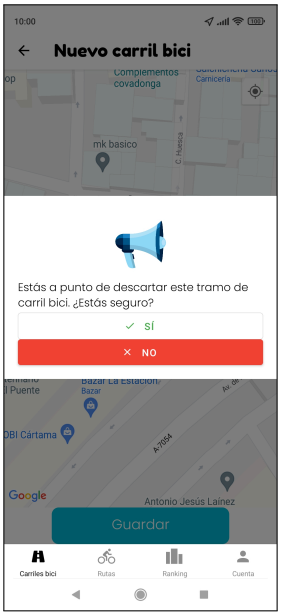


Figura 31: Descartar el carril bici

A.5. Sección: Rutas

Figura 32: Resultado del filtro de rutas más frecuentes

En esta sección puede visualizar la lista de rutas disponibles en la aplicación como se muestra en la Figura 26. Además, puede filtrarlas atendiendo a diferentes criterios. Si pulsa sobre “Creador” y en el campo de búsqueda superior introduce parte del nombre del creador, mostrará las rutas de dicho creador. En cambio, si pulsa sobre “Nombre de la ruta” y en el campo de búsqueda introduce parte del nombre de la ruta, filtrará aquellas rutas que contengan la cadena especificada. Cuando pulsa sobre “Mis rutas”, verá las rutas que ha creado. Pulsando sobre “Rutas privadas” o “Rutas públicas” verá las rutas privadas o públicas. Pinchando sobre “Rutas más frecuentes” ordenará la lista de rutas mostrando en primer lugar aquellas que sean más frecuentes. Es importante tener en cuenta que sólo puede aplicar un filtro cada vez. En la Figura 32 puede ver las rutas más frecuentes de la aplicación.

Para añadir una nueva ruta, pulse el botón "+ NUEVA RUTA" que puede ver en la Figura 26. Tras pulsar dicho botón, verá una pantalla con un mapa que mostrará su ubicación en tiempo real (Figura 33). Usted empezará a realizar la ruta y cuando haya llegado el final de la misma pulsará el botón "Guardar" donde le pedirán que rellene el nombre de la ruta, la visibilidad (pública o privada) y la calificación de la ruta (Figura 34). En cambio, si pulsa sobre la flecha situada en la esquina superior izquierda, descartará la ruta (Figura 35) y tendrá que dar su confirmación. En ambos casos, tanto si decide crear la ruta como si la descarta, será dirigido a la lista de rutas.

Figura 33: Pantalla para añadir una nueva ruta

Figura 34: Guardar la nueva ruta

Figura 35: Descartar la nueva ruta

Si desea ver la información de la ruta, pulse sobre el botón “Ver ruta” y le dirigirá al detalle de la ruta seleccionada como puede ver en la Figura 36. Si desliza en dicha pantalla hacia abajo, podrá ver las estadísticas de la ruta y verá también un botón para ver el perfil de los usuarios más rápidos en hacer la ruta (Figura 37). Pulsando en dicho botón, le mostrará el perfil del usuario (Figura 38) y las rutas públicas del mismo, siendo posible verlas pulsando sobre “Ver ruta”. Si pulsa sobre el botón “Realizar ruta” que puede ver en la Figura 37, le llevará a otra pantalla (Figura 39) donde verá un mapa con su ubicación actual y el recorrido a seguir para completar la ruta. Si pulsa sobre el icono de la esquina superior izquierda tendrá que dar su confirmación para abandonar la ruta que está realizando (Figura 40) y pasará a ver el detalle de la ruta. En cambio, si pulsa sobre “Finalizar recorrido” estará dando por finalizada la ruta y pasará a ver la lista de rutas.

Figura 36: Detalles de la ruta, recorrido

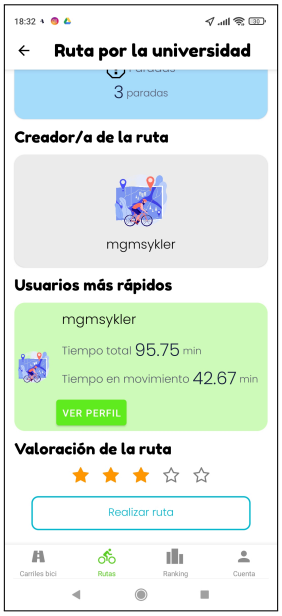


Figura 37: Usuarios más rápidos de la ruta

Figura 38: Ver el perfil de otro usuario

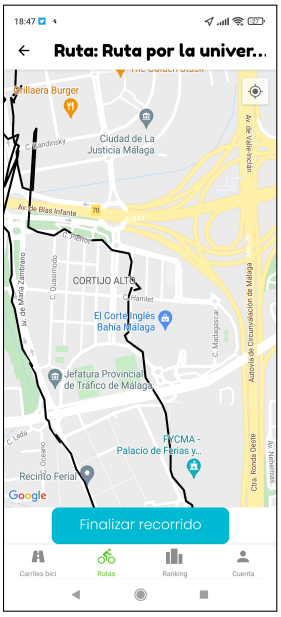


Figura 39: Recorrido para completar la ruta

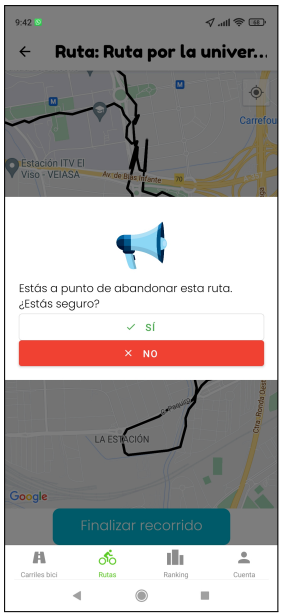


Figura 40: Abandonar ruta en curso

A.6. Sección: *Ranking*

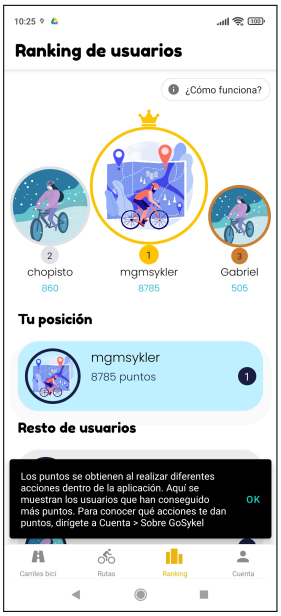


Figura 41: Funcionamiento del *ranking*

En esta sección puede visualizar el *ranking* de usuarios (Figura 27) donde puede consultar los 3 usuarios con más puntos de la aplicación, su posición y la posición del resto de los usuarios.

Si pulsa sobre "¿Cómo funciona?" (Figura 41) verá en la parte inferior de la pantalla un breve mensaje explicando el funcionamiento de esta sección y le indicará dónde puede conocer más información. Dicho mensaje se mostrará durante 10 segundos como máximo y si pulsa sobre "OK", desaparecerá.

A.7. Sección: Cuenta

En la Figura 28 puede observar las acciones que puede realizar para gestionar su cuenta: ver su perfil, sus rutas, acceder a la tienda, conocer un poco más sobre GoSykel o cerrar sesión.

A.7.1. Subsección: Mi perfil

Concretamente, si pulsa sobre “Mi perfil”, le mostrará los datos de su perfil como puede ver en la Figura 42. Desde esta pantalla es posible editar la información de su perfil.

Para editar su perfil, deslice hasta el final de la pantalla y verá el botón “Editar perfil”. Si pulsa sobre el botón, le mostrará la pantalla de edición del perfil de la Figura 43. En ella, debe rellenar el nombre de usuario con uno nuevo si desea cambiarlo, seleccionar el avatar y encabezado para su perfil y rellenar el campo contraseña actual si no quiere cambiar la contraseña. Si, por el contrario quiere cambiar su contraseña, cumplimente el resto de campos y además proporcione la nueva contraseña dentro del campo nueva contraseña. Tras esto, pulse “Guardar cambios” y será redirigido a la información de su perfil.

Por otro lado, si desea eliminar su cuenta, pulse “Borrar perfil”. Le pedirá su confirmación para eliminar el perfil como se muestra en la Figura 46. Si selecciona la opción “Sí”, eliminará su cuenta y le redirigirá a la pantalla de bienvenida.

Figura 42: Información del perfil del usuario

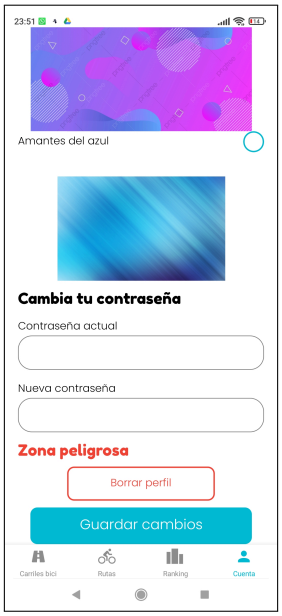


Figura 43: Editar el perfil del usuario

A.7.2. Subsección: Mis rutas

En caso de que quiera consultar sus rutas, en la pantalla de gestión de su cuenta, debe pulsar en “Mis rutas” y podrá ver las rutas que ha creado y las rutas que ha realizado y que

son propiedad de otro usuario (Figura 44). En ambas listas de rutas, si pulsa sobre “Ver ruta” será dirigido a la pantalla de detalles de la ruta.

De las rutas creadas por usted, si pulsa en “Editar ruta” podrá editar el nombre y la visibilidad de la ruta (Figura 45). Tras editar los campos necesarios y pulsar “Guardar cambios” será redirigido a las listas de rutas. Si quiere eliminar la ruta, pulse el botón “Eliminar ruta” que aparece en la Figura 45 y que solicitará su confirmación como le ha sido solicitada en acciones anteriores. Tras dar su confirmación, será redirigido a las listas de rutas. Para retroceder y navegar a la pantalla donde estaba anteriormente, use el icono situado en la parte superior izquierda.

En cuanto a las rutas completadas por usted, si desea eliminar dicha ruta de su registro de rutas completadas, debe pulsar “Eliminar ruta” y se le solicitará su confirmación como se muestra en la Figura 47. Cuando dé su confirmación, dicha ruta será eliminada de las rutas completadas.

Figura 44: Rutas del usuario

Figura 45: Editar la información de la ruta

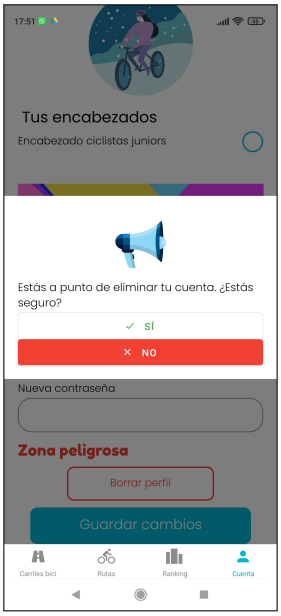


Figura 46: Eliminar el perfil del usuario

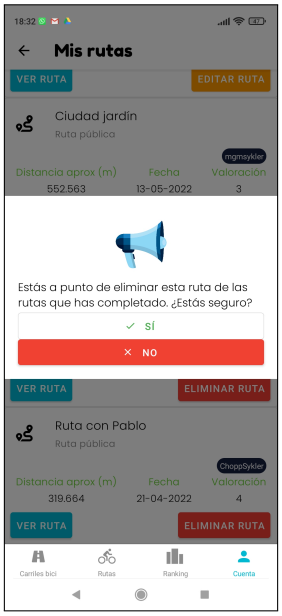


Figura 47: Eliminar ruta completada

A.7.3. Subsección: Tienda

Desde la pantalla de gestión de su cuenta, si pulsa sobre el botón “Tienda” entrará a la tienda de objetos de GoSykel (Figura 48). En dicha tienda podrá ver los diferentes avatares, encabezados e insignias disponibles así como los puntos que puede gastar en la tienda. Para ver los diferentes productos de cada tipo, deslice de derecha a izquierda. Si quiere conocer más información acerca de algún artículo pulse sobre el artículo que desea ver y se le mostrará los detalles del mismo como puede ver en la Figura 49.

Figura 48: Tienda donde canjear puntos

Figura 49: Detalles del artículo seleccionado

Para comprar el artículo seleccionado, pulse sobre el carrito de la compra y le pedirá la confirmación que se muestra en la Figura 50 para canjear los puntos. Si su respuesta es afirmativa comprobará si dispone de puntos suficientes y en caso de tenerlos, dicho artículo pasará a su colección. En caso de que el artículo ya sea parte de su colección, en lugar del carrito de la compra, aparecerá "Comprado" como en la Figura 51.

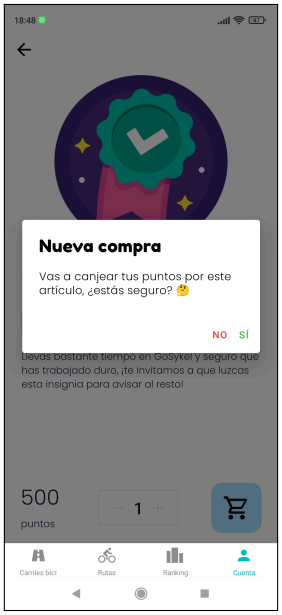


Figura 50: Confirmación de compra

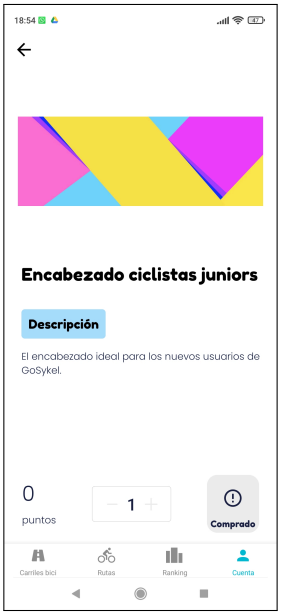


Figura 51: Artículo comprado

A.7.4. Subsección: Sobre GoSykel

Desde la pantalla de gestión de su cuenta, si pulsa en el botón “Sobre GoSykel” entrará a la pantalla de información general de GoSykel que puede ver en la Figura 52. En esta pantalla, puede encontrar información relacionada con el origen de GoSykel, las acciones que puede realizar dentro de la aplicación, las funcionalidades disponibles en la versión actual y la forma de conseguir puntos. Puede volver atrás usando el icono situado en la esquina superior izquierda.

Figura 52: Información general sobre GoSykel

A.7.5. Subsección: Cerrar sesión

Desde la pantalla de gestión de su cuenta, si pulsa el botón “Cerrar sesión” cerrará su sesión en la aplicación, mostrándole la pantalla de bienvenida.

Apéndice B
Manual de despliegue del *backend*

B.1. Introducción

Este documento describe el proceso a seguir para instalar el servidor de la aplicación en su ordenador.

En este manual se detallan de forma clara y ordenada los pasos a seguir para instalar el servidor web que emplea la aplicación en su ordenador.

Para completar los pasos que se describen posteriormente, es necesario que el sistema operativo de su ordenador sea Windows o Linux.

Si le surge cualquier duda o comentario, por favor, no dude en enviar un email a esta dirección.

B.2. Pre-requisitos

Para ejecutar los pasos de este manual correctamente, se recomienda que tenga el siguiente software instalado en su ordenador:

- Python: durante el desarrollo se ha usado la versión 3.10.2
- Postman

B.3. Instalación de dependencias

En primer lugar, descargue los archivos adjuntos en la entrega. Entre estos archivos, localice uno llamado GoSykel.zip, y descomprímalo. Como podrá ver, tiene dos directorios: GoSykel-Backend-Public y GoSykel-Frontend-Public. Muévase al directorio GoSykel-Backend-Public.

Dentro del directorio, abra una terminal y ejecute el comando

```
python -m venv <nombre-venv>
```

para crear un nuevo entorno virtual donde instalar todas las dependencias sin interferir con otras instalaciones que pudiera tener en su ordenador.

Active el entorno virtual, para activarlo en Windows ejecute el comando

```
<nombre-venv>\Scripts\activate
```

o para activarlo en Linux ejecute el comando

```
source <nombre-venv>/bin/activate
```

El entorno virtual está activo cuando en la línea de comandos le aparece lo siguiente:

```
(nombre-venv) path\hacia\la\carpeta\GoSykel-Backend-Public>
```

Use el paquete pip para instalar las dependencias. Para ello ejecute:

```
pip install -r requirements.txt
```

B.4. Configuración de Firebase

De cara a ejecutar el proyecto con sus credenciales de Firebase, por favor, siga los pasos de este tutorial para crear un proyecto de Firebase y habilitar la autenticación con correo electrónico y contraseña. Es importante seleccionar la opción web a la hora de crear el proyecto en Firebase.

Además, si planea usar el servidor en combinación con la aplicación móvil desarrollada en React Native, le recomiendo que guarde la información de la configuración del proyecto de Firebase que acaba de crear. Por favor, diríjase al manual que contiene las instrucciones de ejecución de la aplicación móvil donde se le explicará como proceder.

Para habilitar Firebase Admin SDK diríjase al proyecto que acaba de crear >Configuración del proyecto >Cuentas de servicio >Seleccionar Python >Generar nueva clave privada.

Esto descargará un archivo JSON (serviceAccountKey.json) cuyo contenido debe copiar al archivo llamado firebase.py ubicado dentro de la carpeta firebase. Una vez abra el archivo firebase.py, reemplace los siguientes campos por la información disponible en el archivo JSON que acaba de guardar.

```
  CERTIFICATE = {
  "type": "complete",
  "project_id": "complete",
  "private_key_id": "complete",
  "private_key": "complete",
  "client_email": "complete",
  "client_id": "complete",
  "auth_uri": "complete",
  "token_uri": "complete",
  "auth_provider_x509_cert_url": "complete",
  "client_x509_cert_url": "complete"
}
```

B.5. Uso

Tenga en cuenta que el servidor se ejecutará en el puerto 5000, así que si tiene algún servicio ejecutándose en ese puerto, deténgalo.

Tras instalar las dependencias, para poner en marcha el servidor con el entorno virtual activo ejecute

```
  python app.py
```

Para comprobar que el servidor funciona como se espera, abra Postman y cree una nueva petición POST cuyo *endpoint* sea http://127.0.0.1:5000/users. En el cuerpo de la petición POST complete la siguiente información

```
  {
    "email": "completar",
    "nickname": "completar"
 }
```

Esto creará un nuevo usuario, si todo ha ido bien le devolverá el ID del usuario que acaba de crear y un *token* que deberá incluir en el *header Authorization* en todas las peticiones que haga al servidor.

Si por el contrario desea obtener un *token* iniciando sesión tras haber creado un nuevo usuario previamente, realice una petición POST al *endpoint* http://127.0.0.1:5000/login. En el cuerpo de la petición complete la siguiente información

```
{
  "email": "completar"
}
```

Tras esta petición el servidor le devolverá el ID del usuario que acaba de iniciar sesión y el *token* correspondiente a dicho usuario y que deberá incluir en peticiones posteriores.

De cara a ser rigurosos con el ciclo de vida de los *tokens*, tras comprobar el funcionamiento del servidor en local debe hacer una petición GET al *endpoint* http://127.0.0.1:5000/logout indicando en el *header Authorization* el *token* devuelto anteriormente.

Cuando termine de probar el servidor en local ejecute el comando

```
Ctr + C
```

para detener la ejecución del servidor. Si ha realizado cambios y los quiere probar en local, ejecute el comando

```
Ctr + C
```

y acto seguido inicie de nuevo el servidor con el comando

```
python app.py
```

Para desactivar el entorno virtual donde se estaba ejecutando el servidor anteriormente ejecute

```
deactivate
```

Este comando sirve tanto para Windows como para Linux.

B.6. Despliegue en la nube

Para desplegar los cambios que se realicen sobre el servidor se le ofrecen varias alternativas:

- Firebase

- Heroku

Tras desplegar la aplicación en uno de esos dos proveedores se le asignará una URL para acceder a la aplicación. Guarde esta URL ya que será la que usará para probar los *endpoints* con Postman. Si quiere usar su propia versión del servidor en la aplicación móvil conserve la URL y diríjase al manual que contiene las instrucciones de ejecución de la aplicación móvil donde se le explicará como proceder.

Apéndice C
Manual de instalación y ejecución de la aplicación móvil en un dispositivo físico

C.1. Introducción

Este documento describe el proceso a seguir para instalar y ejecutar la aplicación móvil en su dispositivo Android.

En este manual se detallan de forma clara y ordenada los pasos a seguir para instalar la aplicación móvil en su teléfono.

Para completar los pasos que se describen posteriormente, es necesario que el sistema operativo de su teléfono sea Android.

Si le surge cualquier duda o comentario, por favor no dude en enviar un email a esta dirección.

C.2. Pre-requisitos

Para ejecutar los pasos de este manual correctamente, se recomienda que tenga el siguiente software instalado en su ordenador:

- Expo: durante el desarrollo se ha usado la versión 44 del SDK. En el tutorial puede encontrar información sobre cómo instalar Expo Go, la aplicación móvil que le permite probar el código en un dispositivo real a medida que desarrolla la aplicación.

C.3. Instalación de dependencias

En primer lugar, descargue los archivos adjuntos en la entrega. Entre estos archivos, localice uno llamado GoSykel.zip, y descomprímalo. Como podrá ver, tiene dos directorios: GoSykel-Backend-Public y GoSykel-Frontend-Public. Muévase al directorio GoSykel-Frontend-Public y abra la carpeta go-sykel.

Dentro del directorio, abra una terminal y ejecute el comando

```
npm install
```

que instalará las dependencias incluidas dentro del archivo package.json.

C.4. Configuración de Firebase

De cara a ejecutar el proyecto con sus credenciales de Firebase, por favor, siga los pasos de este tutorial para crear un proyecto de Firebase y habilitar la autenticación con correo electrónico y contraseña. Es importante seleccionar la opción web a la hora de crear el proyecto en Firebase.

Además, guarde la información de la configuración del proyecto de Firebase que acaba de crear.

Tras crear el proyecto o si ya tiene un proyecto creado y tiene dicha información, diríjase a la carpeta firebase y dentro del archivo Firebase.js complete la sección *firebaseConfig* con los datos obtenidos durante la creación de la aplicación en la consola de Firebase.

```
const firebaseConfig = {
    apiKey: "complete",
    authDomain: "complete",
    projectId: "complete",
    storageBucket: "complete",
    messagingSenderId: "complete",
    appId: "complete"
};
```

Tenga en cuenta que para usar esta configuración de Firebase también debe configurar el servidor y desplegarlo con la nueva configuración.

Una vez tenga desplegado el servidor con la configuración de Firebase, obtenga la URL donde está desplegado y diríjase a la carpeta *constants*, abra el archivo *Constants.js* y asigne a la variable *SERVER_URL* la URL del servidor.

C.5. Uso de la aplicación en local

Tras instalar las dependencias y configurar Firebase es momento de poner en marcha la aplicación móvil. Para ello, asegúrese de que su teléfono móvil y su ordenador están conectados a la misma red WiFi. Posteriormente, abra una consola y ejecute el comando

```
npm run start
```

Este comando empezará a mostrar información en consola y le proporcionará un código QR.

Para leer dicho código QR en su dispositivo Android, abra la aplicación Expo Go y seleccione la opción *'Scan QR code'* que le mostrará la cámara de su dispositivo y podrá escanear el código QR que pondrá en marcha la aplicación en su teléfono.

A medida que vaya haciendo cambios en el código y los vaya guardando se irá actualizando la aplicación en su teléfono y podrá probar los cambios al instante.

Cuando termine de probar la aplicación móvil ejecute el comando

```
Ctr + C
```

para detener la ejecución de la aplicación.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA