



Bikethe Ride



Rubén Jesús Ruiz Cantero
Desarrollo de aplicaciones multiplataformas

Contenido

1.	PRESENTACIÓN DEL PROYECTO	2
1.1.	Explicación resumida.....	2
1.2.	Estudio de mercado.....	2
1.3.	Valor del producto	6
2.	Análisis de la solución.....	6
2.1.	Análisis de la solución	6
2.2.	Análisis de escenarios	9
3.	PLANIFICACIÓN DE TAREAS Y ESTIMACIÓN DE COSTES.....	12
3.1.	Planificación y organización de tareas.	12
3.2.	Estimación de costes y recursos: hardware, software y humanos.	15
3.3.	Herramientas usadas.	16
3.4.	Gestión de riesgos.	16
4.	DISEÑO DE LA SOLUCIÓN	19
4.1.	Diseño de la interfaz de usuario y prototipos.	19
4.2.	Diagrama de clases	29
4.3.	Diseño de la persistencia de la información	32
4.4.	Arquitectura del sistema	34
5.	IMPLEMENTACIÓN DE LA SOLUCIÓN	37
5.1.	Ánáisis tecnológico.....	37
5.2.	Elementos a implementar.....	43
6.	Testeo y pruebas.....	57
6.1.	Plan de pruebas	57
7.	Repositorio	73
8.	Bibliografía.....	74

1. PRESENTACIÓN DEL PROYECTO

1.1. Explicación resumida.

En estos tiempos en los que la eficiencia energética y el cambio climático están a la orden del día, toda iniciativa para cuidar del medio ambiente y reducir nuestra huella de carbono es bienvenida.

Voy a implementar una aplicación de alquiler de bicicletas, en la que una persona decide ofrecer su bicicleta a otra persona.

Para acceder a la aplicación podremos hacerlo mediante registro de un usuario con correo electrónico y contraseña, o mediante autenticación con una cuenta de Google.

Tendremos a nuestra disposición el catálogo de bicicletas disponibles para el alquiler en un día determinado, con información como la foto de la bicicleta, el nombre del dueño o una pequeña descripción de la bicicleta. Al hacer la reserva se le notificará al dueño mediante mensaje. Podremos ver las reservas que hemos realizado, y tendremos la opción de eliminar una reserva, en cuyo caso también se le notificará al dueño mediante mensaje.

Además, dispondremos de un mapa en el que aparecerá la ubicación de las bicicletas disponibles.

Tendremos la opción de añadir una nueva bicicleta para ser alquilada, la cual se agregará con nuestra ubicación actual. Así mismo, podremos ver cuales son las bicicletas que tenemos en alquiler, y la posibilidad de eliminarla del catálogo.

Podremos acceder a nuestro perfil y modificar nuestro nombre de usuario.

1.2. Estudio de mercado

Un estudio de mercado es una investigación que se realiza para conocer mejor las perspectivas comerciales ante el lanzamiento de una marca, un producto o un servicio, con el objetivo de intentar reconocer las potenciales respuestas de los clientes y de la competencia hacia ellos.

El estudio de mercado servirá para tomar decisiones de negocio orientadas a satisfacer las necesidades de los clientes. Para eso, es importante conocer su opinión y recordar que, a veces se te puede escapar alguna creencia o percepción que podría determinar el éxito de tu emprendimiento.

Los pasos que se seguirán para elaborar el estudio de mercado son:

- Definir las fuentes de información
- Definir el público objetivo
- Conocer la competencia
- Análisis DAFO

- **Definir las fuentes de información**

Llevaremos a cabo una investigación primaria, con la que buscar la manera de entrar en contacto directo con los consumidores finales, con el objetivo de que sean ellos mismos quienes manifiesten su opinión sobre el producto.

Es una información de primera mano sobre tu mercado y tus clientes.

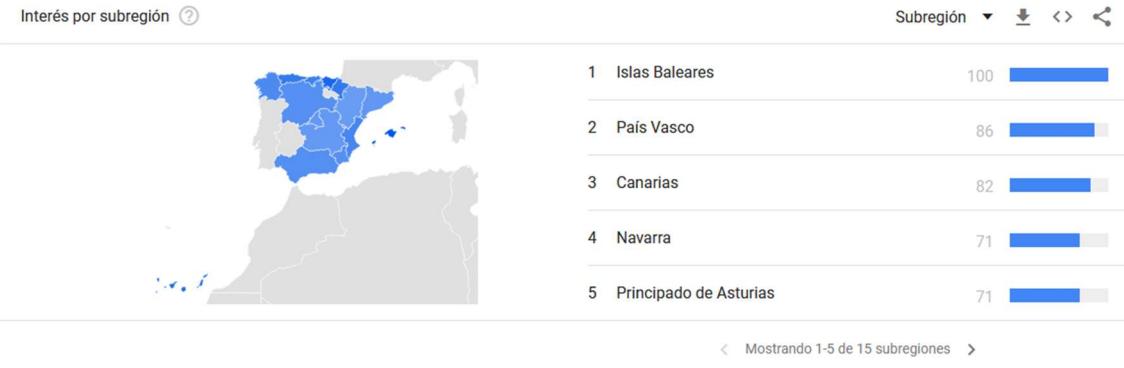
Dado que tenemos acceso a un perfil de Strava (red social enfocada a deportistas) con un amplio abanico de conocidos, usaremos estos datos para realizar el contacto con los posibles consumidores.

Se llevará a cabo mediante encuestas online, con Google Forms como herramienta de encuestas online. Una vez diseñada la encuesta, se creará una dirección URL para acceder al cuestionario, la cual se compartirá por correo electrónico.

Unas posibles preguntas para la encuesta serían:

- ¿Consideras que este servicio satisface tus necesidades? (Si/No)
- ¿Usas la bicicleta en tus viajes a otras ciudades? (Si/No)
- ¿Qué tipo de bicicleta de gustaría que hubiese para alquilar? (Montaña/Carretera/Urbana/Eléctrica/Otras)
- ¿Te gustaría alquilar tu bicicleta a otra persona? (Si/No)
- ¿Con qué frecuencia usas la bicicleta como medio de transporte? (Todos los días/Varias veces a la semana/Ocasional)

Otra forma de recopilar información sería usar Google Trends, como un instrumento muy potente para comprobar cuál son las tendencias en uno de los buscadores más importantes en el mundo. Haciendo una búsqueda de “alquiler de bicicletas” en España, podemos hacernos una idea de en qué zonas están más interesados por el alquiler de bicicletas, y así hacer más énfasis en la difusión y marketing del servicio.



- **Definir el público objetivo**

Debemos averiguar los distintos tipos de clientes que existen y conocer en profundidad las personas que adquieren nuestro producto. Algunos de los elementos variables que debemos definir son:

- Demográfico: años, género, ocupación, ingresos, etc.
- Psicográfica: hábitos de consumo
- Geográfico: la región o ubicación del producto o servicio.
- Actitudinal: el por qué consumen determinados productos y no otros.

Por tanto, podríamos decir que en nuestro público objetivo el género es indiferente, pues va dirigida a ambos sexos, la edad la podríamos delimitar a personas mayores de 14-16 años de cualquier clase social, que tengan un hábito de vida saludable y le guste el deporte, de cualquier ubicación, y que ya sea por viaje o por ocio quieran dar un agradable paseo con una bicicleta.

- **Conocer la competencia**

Es primordial analizar el estado del sector en el que nos encontramos para conocer a los competidores, ya que ayudará a mejorar el servicio.

Como es una aplicación móvil para alquiler de bicicletas, tan solo debemos echar un vistazo a que aplicaciones tenemos ya disponibles.

Algunas de las aplicaciones son:

- Bicis Barcelona, Sevici Sevilla, Chicago Bikes, NYC Citi Bike...: Son aplicaciones para consultar toda la información sobre las estaciones del servicio de alquiler de bicicletas, la gran mayoría de ciudades grandes dispone de aplicación.

- Lime - #RideGreen: Alquiler de bicicletas eléctricas en una amplia gama de ciudades repartidas por todo el mundo.
- Donkey Republic Bicicletas: Alquiler de bicicletas en una amplia gama de ciudades repartidas por todo el mundo.

Los rivales más directos serían Lime y Donkey Republic, ya que las otras aplicaciones tienen un ámbito geográfico muy restringido (solo a dicha ciudad).

- **Análisis DAFO**

El DAFO (iniciales de Debilidades, Amenazas, Fortalezas y Oportunidades) es una herramienta que permite analizar la realidad de su empresa, marca o producto para poder tomar decisiones de futuro.



En la fase de análisis interno valoramos:

- Debilidades: Se recogen todo aquello que limita la posición predominante del producto respecto a la competencia.
- Fortalezas: Los factores internos que proporcionan una ventaja comparativa dentro de nuestros competidores.

En la fase de análisis externo valoramos:

- Amenazas: Recogen cualquier factor que frene nuestra posición en el mercado.
- Oportunidades: Aquellos aspectos positivos que te brinda el entorno y que puedes aprovechar para crecer.

Nuestro diagrama de análisis DAFO quedaría:

Debilidades <ul style="list-style-type: none"> – Presupuesto limitado – Falta de experiencia en el sector 	Amenazas <ul style="list-style-type: none"> – Desconfianza de los usuarios por la novedad del servicio
Fortalezas <ul style="list-style-type: none"> – Atención personalizada, centrándonos en lo mas importante que es el usuario 	Oportunidades <ul style="list-style-type: none"> – No hay mucha competencia, lo que favorece el servicio

1.3. Valor del producto

Después de realizar el análisis de mercado, y analizando los resultados, podemos decir que el servicio que vamos a ofrecer tiene un alto potencial de negocio, ya que sería un servicio mas personal, pues serían los propios usuarios los que prestarían el producto. También podría tener una amplia disposición geográfica que no se consigue con otras aplicaciones, si podemos disponer de una gran base de datos de usuarios.

2. Análisis de la solución

2.1. Análisis de la solución

Requisitos funcionales

Un requisito funcional es una declaración de cómo debe comportarse un sistema. Define lo que el sistema debe hacer para satisfacer las necesidades o expectativas del usuario. Los requisitos funcionales se pueden considerar como características que el usuario detecta.

Requisitos funcionales de la aplicación:

1. El sistema debe permitir el registro de usuario.
2. El sistema debe permitir el registro de usuario mediante cuenta de Google.
3. El sistema debe permitir el inicio de sesión de usuario.
4. El sistema debe tener una sección en la que aparezcan las bicicletas disponibles.
5. El sistema debe tener una interfaz fácil de usar para los usuarios.
6. El sistema debe tener una sección de mapa.
7. El usuario podrá ver la fotografía de una bicicleta, así como el nombre del dueño y la ubicación.
8. El sistema debe proporcionar un mecanismo de comunicación entre usuarios.
9. El sistema debe tener una sección en la que permita añadir una nueva bicicleta.
10. El usuario debe poder enviar un mensaje a otro usuario.
11. El sistema debe permitir la selección de un día en concreto al usuario.
12. El usuario debe poder ver en un mapa la ubicación de las bicicletas disponibles.
13. El usuario podrá registrar una bicicleta con la fotografía, sus datos personales y la ubicación en la que se encuentra.
14. El sistema debe permitir cerrar sesión al usuario.
15. El sistema debe permitir ver las reservas realizadas.
16. El sistema debe permitir ver las bicicletas en alquiler de un usuario.
17. El usuario debe poder eliminar una bicicleta en alquiler.
18. El usuario debe poder modificar su perfil.
19. El usuario debe poder eliminar una reserva realizada.

Requisitos no funcionales

Un requisito no funcional especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos. Se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento. Son las restricciones o condiciones que impone el cliente al programa que necesita.

Requisitos no funcionales de la aplicación:

1. El sistema deberá estar alojado en un servidor eficiente que pueda manejar gran concurrencia de usuario en ciertos periodos de tiempo.
2. El acceso a los datos debe ser de forma segura.
3. El sistema deberá ser fácilmente escalable, con el fin de poder hacer crecer la aplicación al incorporar a futuro nuevas funcionalidades.
4. El sistema estará restringido bajo usuarios definidos.

Requisitos de información

Los requisitos de información describen la información que debe almacenar y gestionar el sistema para dar soporte a los procesos de negocio.

1. De un usuario se almacenará:

- Id
- Nombre
- Email

2. De una bicicleta se almacenará:

- Dueño
- Id usuario
- Dirección
- Ciudad
- País
- Fotografía
- Descripción
- Latitud
- Longitud
- Email

3. De una reserva se almacenará:

- Ciudad
- Fecha
- Email dueño bicicleta
- Email usuario
- Id usuario

2.2. Análisis de escenarios

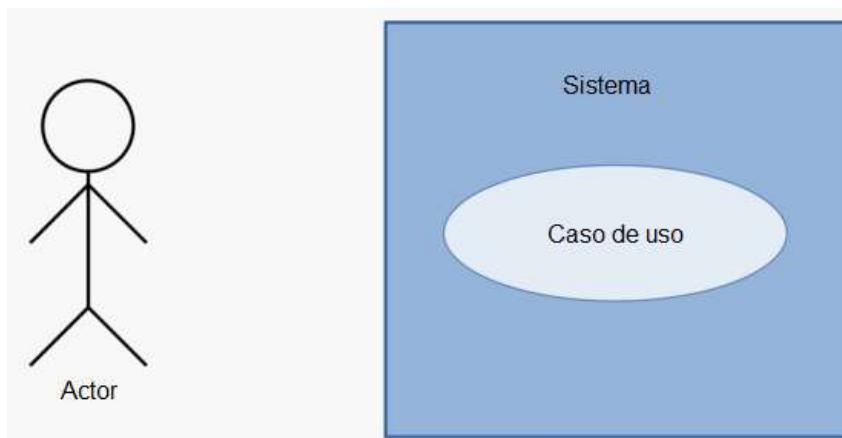
Un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto. El conjunto de casos de uso forma el "comportamiento requerido" de un sistema.

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto, los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar.

Un diagrama de casos de uso es una visualización gráfica de los requisitos funcionales del sistema, que está formado por casos de uso (se representan como elipses) y los actores que interactúan con ellos (se representan como monigotes).

Los elementos de un diagrama de casos de uso son:

- Actor: Representa un tipo de usuario del sistema.
- Casos de uso: Función o una acción dentro del sistema.
- Sistema: El sistema se utiliza para definir el alcance del caso de uso y se dibuja como un rectángulo.



La relación entre los elementos del diagrama se representa con unas líneas de conexión llamadas asociaciones. Las relaciones entre un actor y un caso de uso se representan mediante una línea continua entre ellos. Las relaciones entre casos de uso se representan con una flecha discontinua con el nombre del tipo de relación como etiqueta, que pueden ser de inclusión (la ejecución de un caso de uso implica necesariamente la ejecución del segundo) o de extensión (la ejecución de un caso de uso puede provocar la ejecución del segundo).

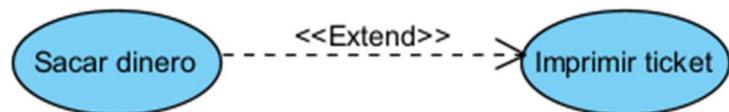
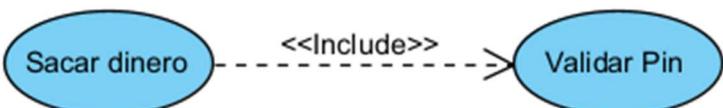
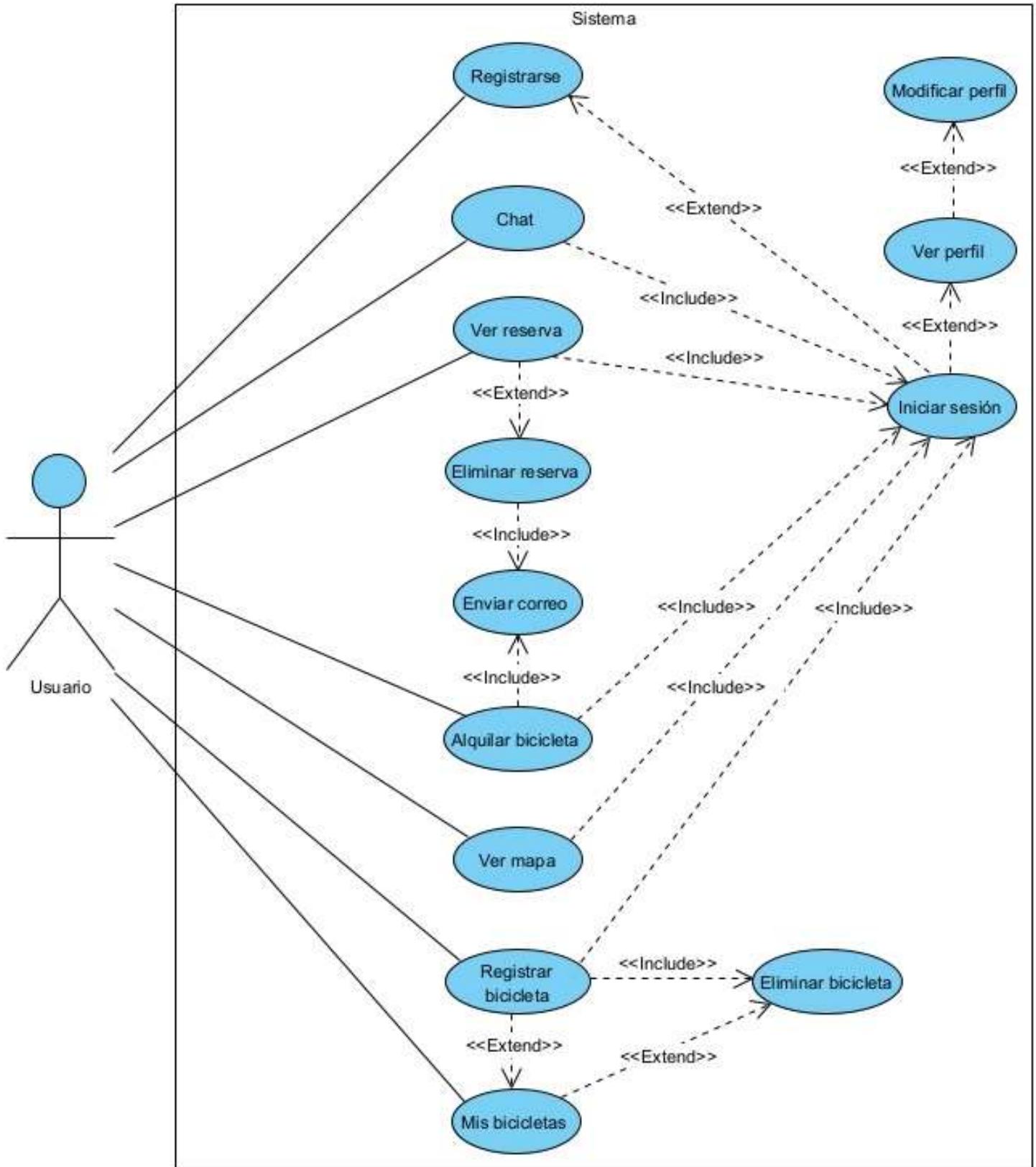


Diagrama de casos de uso



3. PLANIFICACIÓN DE TAREAS Y ESTIMACIÓN DE COSTES.

3.1. Planificación y organización de tareas.

Para ayudarnos a organizar y priorizar el trabajo, usaremos la herramienta proyectos de github. Crearemos diferentes issues con las tareas a realizar, las cuales podremos ir añadiendo a nuestro proyecto, indicándole una fecha determinada.

Una issue en detalle:

La planificación de tareas se ha gestionado de la siguiente forma:

- 5 de octubre

Explicación resumida del proyecto.

- 10 de octubre

Análisis de la solución, en la que se identificarán los requisitos funcionales, no funcionales y de información.

- 13 de octubre

Presentación del proyecto, aportando un estudio de mercado y realizando una valoración del producto.

- 17 de octubre

Análisis de escenarios.

- 24 de octubre

Planificación y organización de tareas.

- 27 de octubre

Planificación y costes, en la que se hará una estimación de costes y recursos humanos, salarios, hardware, software, licencias, financiación y ayudas. Además, describiremos las herramientas a usar, y la gestión de riesgos.

- 31 de octubre

Diseño de la interfaz de usuario.

- 3 de noviembre

Diseño de la solución, en la que se diseñara el diagrama de clases y el diagrama de persistencia de la información.

- 7 de noviembre

Arquitectura del sistema.

- 13 de noviembre

Análisis tecnológico.

- 27 de noviembre

Implementar login y registro.

Implementar registro bici.

Implementar mapa.

Implementar chat.

Implementar acceso a bicis.

Implementar acceso y modificar perfil.

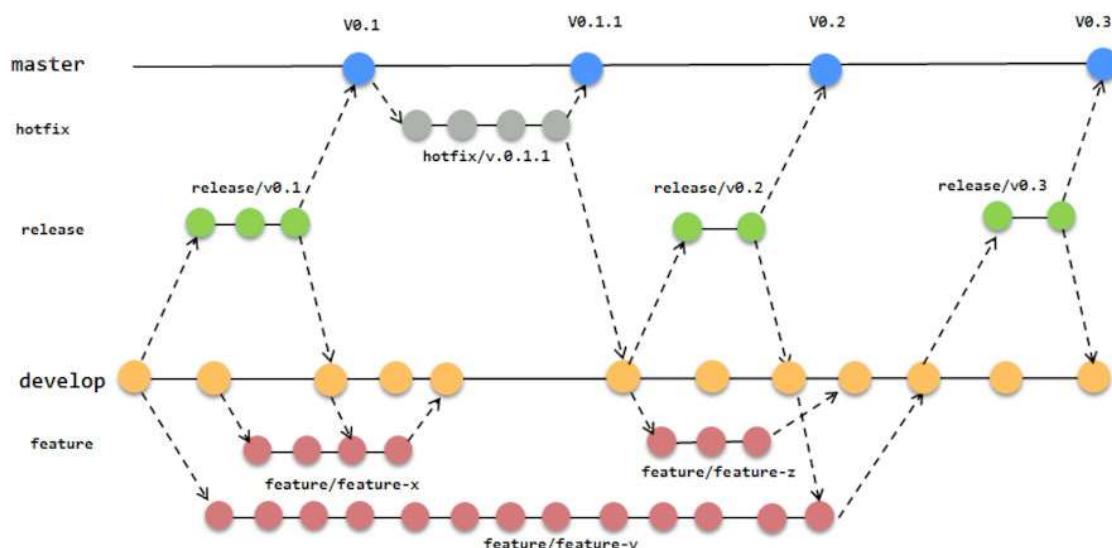
Implementar ver reservas.

Implementar mis bicis.

- 11 de diciembre

Entrega de proyecto, documentación y repositorio.

Para la gestión del repositorio se usará el flujo de trabajo Gitflow, que define un modelo de ramificación para gestionar el proyecto de desarrollo de software.



Tendremos dos ramas principales, master y develop, y varias ramas de soporte, como feature, release y hotfix.

Master contiene la versión que se usará en producción, y develop es una rama de integración para nuevas características en desarrollo. Se usará una rama feature a partir de una rama develop para desarrollar nuevas funcionalidades, y al terminar se fusiona con develop. Release será una rama de lanzamiento y se fusionará con master y con develop. En caso de que haya que corregir el lanzamiento ya en producción, se usará una rama hotfix, la cual saldrá de master y finalizará fusionándose con master y develop.

Para documentar el proyecto se usará la herramienta Wiki de GitHub, donde se alojará toda la documentación.

3.2. Estimación de costes y recursos: hardware, software y humanos.

La estimación de costes es el proceso de prever los recursos financieros y otros necesarios para completar un proyecto dentro de un alcance definido. La estimación de costes tiene en cuenta cada elemento requerido para el proyecto, desde los materiales hasta la mano de obra, y calcula una cantidad total que determina el presupuesto de un proyecto.

Una estimación de costes inicial puede determinar si se da luz verde al proyecto, y si el proyecto avanza, la estimación puede ser un factor en la definición del alcance del proyecto. Si la estimación de costes es demasiado alta, se puede decidir reducir el proyecto, o empezar a obtener financiación para el proyecto.

Los gastos derivados del hardware serán únicamente la publicación de la aplicación en Google Play, que tiene un coste de 25\$ (al cambio actual son 25€). La base de datos se implantará en un servidor remoto, Firebase, en el cual tenemos la opción de crear un “Plan Spark” que es gratuito, el cual nos valdrá dado que en un primer momento el volumen de usuarios y transacciones no será muy grande. Este plan nos ofrece 1 GB de almacenamiento total, 20K escrituras/día, 50K lecturas/día y 20K eliminaciones/día.

En cuanto al coste de software, todas las aplicaciones que se usarán son gratuitas, las cuales se detallaran en el siguiente apartado.

El coste por hora de un programador autónomo/freelance parte desde 30€/h, por lo tanto, nos ajustaremos a este precio base.

Con una estimación de 190h, el coste del desarrollador será de 5700€.

Como para el backend vamos a utilizar Firebase, el coste del servidor será de 0€ con el plan gratuito que se nos ofrece.

El coste total quedaría desglosado de la siguiente forma:

Coste humano	5700€
Publicación aplicación	25€
Gasto servidor	0€
Coste total	5725€

Como forma de financiación, actualmente está disponible el “Plan de Impulso al Trabajo Autónomo”, dotado con 3.000 euros que pueden llegar hasta los 4.200 euros en función de si la actividad se desarrolla en zonas en riesgo de despoblación. La solicitud de esta subvención se podrá presentar de forma telemática, a través de un formulario incluido en la sede electrónica de la Administración de la Junta de Comunidades de Castilla-La Mancha.

3.3. Herramientas usadas.

Se usará LibreOffice, que es un paquete de software de oficina libre y de código abierto.

Para el desarrollo del código se usará Android Studio, que es el entorno de desarrollo integrado oficial para la plataforma Android, con licencia gratuita.

Para el diseño del diagrama de casos de uso y diagrama de clases nos apoyaremos en Visual Paradigm, que es una herramienta UML CASE, que en su versión Community Edition es gratuita.

Como sistema de control de versiones se usará Git y la plataforma GitHub para el alojamiento del proyecto.

3.4. Gestión de riesgos.

El análisis y la gestión del riesgo son una serie de pasos que ayudan al equipo del software a comprender y a gestionar la incertidumbre.

Los principales riesgos laborales y como minimizarlos son:

- Fatiga visual o muscular

Hay que tener una colocación ergonómica de la pantalla. También es importante que el espacio cuente con la luz adecuada. Evitar posturas forzadas o inadecuadas, así como estar mucho tiempo sentado.

- Golpes o caídas

El lugar de trabajo debe mantenerse limpio y ordenado, dejando libre de obstáculos las zonas de paso.

- Contacto eléctrico

Revisar el estado de cada equipo antes de su uso.

- Carga mental

Realizar paradas periódicas para prevenir la fatiga.

Los tipos de riesgo relacionados con el software son:

- Riesgos del proyecto

Pueden amenazar al plan del proyecto, porque puede retrasar el proyecto y aumentar los costos.

Identifican problemas de:

- Presupuesto
 - Personal
 - Recursos
 - Cliente
 - Requisitos
- Riesgos técnicos

Amenazan la calidad del software y la planificación temporal del proyecto. La implementación puede llegar a ser difícil o imposible.

Identifican problemas de:

- Diseño
 - Implementación
 - De interfaz
 - Verificación
 - Mantenimiento
- Riesgos del negocio

Amenazan la viabilidad del software a construir. Se pone en riesgo el proyecto y el producto.

Identifican problemas de:

- Mercado
- Estrategia
- Ventas
- Gestión
- Presupuesto

Riesgo	Acción
<ul style="list-style-type: none"> – Tamaño de la base de datos – Número de usuarios – Se hace uso de una base de datos centralizada 	Se dispondrá de un servicio centralizado y escalable a las necesidades actuales (Firebase)
<ul style="list-style-type: none"> – Están todas las herramientas de desarrollo integradas – Hay herramientas de prueba apropiadas 	Dispondremos de un IDE de calidad
<ul style="list-style-type: none"> – Fecha límite de entrega 	Se propondrá una fecha límite de entrega razonable
<ul style="list-style-type: none"> – Cantidad y calidad de la documentación – Se hace uso de repositorio centralizado 	Tendremos a disposición las herramientas que GitHub nos ofrece
<ul style="list-style-type: none"> – Costos asociados al retraso en la entrega 	Se deberá seguir la planificación y cumplir con los plazos
<ul style="list-style-type: none"> – Costos asociados a errores en el producto 	El producto deberá pasar correctamente una serie de pruebas
<ul style="list-style-type: none"> – Tienen los miembros las técnicas apropiadas 	El personal del equipo deberá estar capacitado correctamente

<ul style="list-style-type: none"> - Está el personal comprometido en toda la duración del proyecto - Tiene el personal la necesaria formación - Es requerida una interface de usuario especializada - El producto final no es de calidad 	
<ul style="list-style-type: none"> - Hay suficiente gente disponible 	Ajustar el proyecto al personal disponible

4. DISEÑO DE LA SOLUCIÓN.

4.1. Diseño de la interfaz de usuario y prototipos.

La interfaz de usuario es el medio visual que combina una serie de controles y elementos que permiten al usuario comunicarse e interactuar con el dispositivo electrónico.

En el diseño de la aplicación, la interfaz que planteemos será fundamental para que el usuario se sienta cómodo usándola. Un mal diseño puede provocar que los clientes abandonen nuestra aplicación.

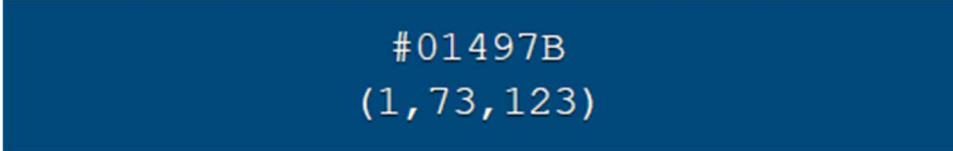
Para diseñar una interfaz usable, es decir, que los usuarios puedan interactuar con ella de la forma más fácil, cómoda e intuitiva posible, las características más importantes son:

- Útil: que sea capaz de cumplir las tareas específicas para la cual se ha diseñada.
- Fácil de usar: que sea eficiente, veloz y con la menor cantidad de errores posibles.
- Fácil de aprender: que no se necesite excesivo tiempo en aprender a trabajar con la aplicación y que sea sencillo recordar su funcionamiento.
- Elegante en su diseño: para favorecer la percepción del usuario y sus emociones.
- Simplicidad de diseño de la interfaz: evitar sobrecargarla será esencial para un buen uso de la misma.

El color es un elemento de gran importancia en la determinación del éxito de la interfaz. Utilizado adecuadamente, constituye una potente herramienta de comunicación. Su uso inadecuado puede provocar molestias visuales o asociaciones erróneas que impidan la

correcta percepción del mensaje. Hay que tener precaución en no abusar del color, pues puede producir confusión al usuario.

Para la pantalla de inicio de sesión y registro se ha usado principalmente el azul, con un fondo en diferentes tonalidades, y el color #01497B para el texto del logo y los botones. Además, un dorado #D4AF37 en el ícono de la bicicleta para resaltar y no ser tan monótono, y el blanco para que sea mas legible el texto de los botones.



#01497B
(1, 73, 123)



#D4AF37
(212, 175, 55)

En el resto de la aplicación se ha usado el blanco, negro, azul #01497B para los botones de interacción y rojo #E2202C en algunos elementos para dar la sensación de advertencia.



#E2202C
(226, 32, 44)

En cuanto a las fuentes se han usado dos tipos:

- Para el logo y los títulos de las diferentes secciones, Krona One.



- Para los demás textos, Roboto.

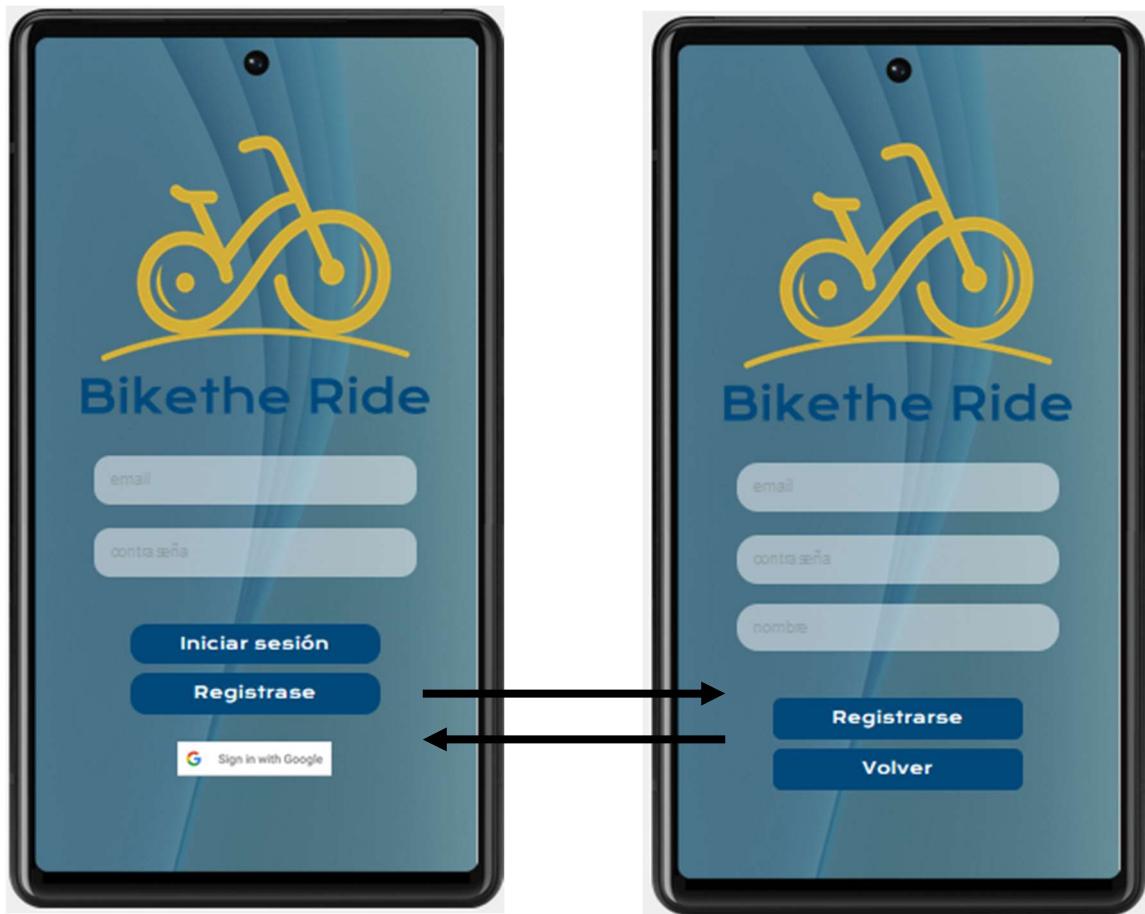


A continuación, pasamos a ver el prototipo de la interfaz de usuario:

- Inicio de sesión y registro

La primera pantalla que aparece al abrir la aplicación es la de inicio de sesión, en la que nos encontramos dos campos de texto, uno para introducir el email y otro para la contraseña. Si las credenciales son correctas podremos acceder a la aplicación pulsando el botón de iniciar sesión. También hay un botón disponible para iniciar sesión con cuenta de Google.

En caso de no disponer de cuenta de usuario, tendremos el botón de registrarse para darnos de alta en el sistema. Se podrá ingresar un correo electrónico, contraseña y nombre de usuario y pulsar el botón de registrar, o volver a la pantalla de inicio de sesión.

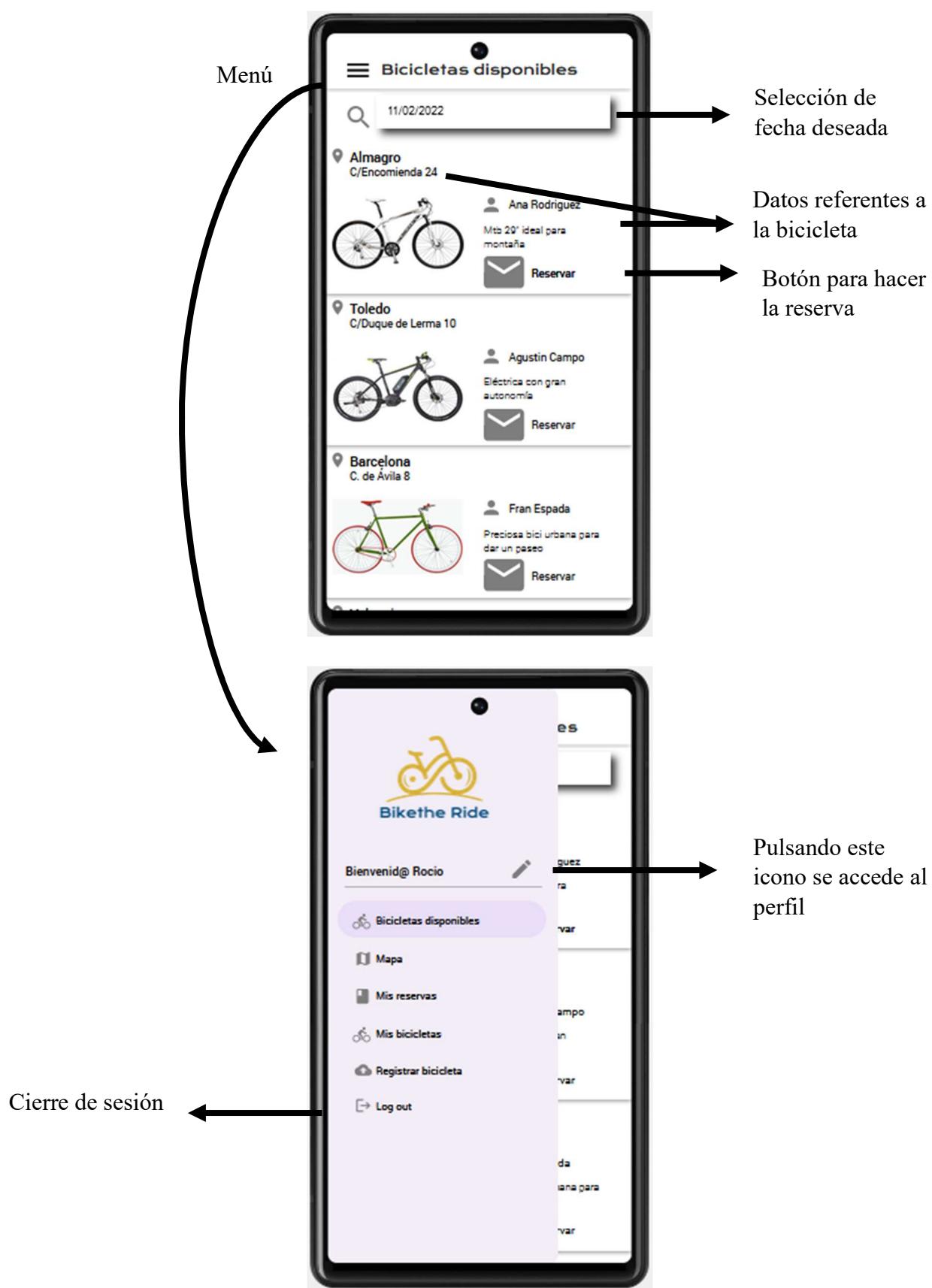


- Bicicletas disponibles

Una vez hecho el inicio de sesión, aparecerá la pantalla de bicicletas disponibles con el título como cabecera. Nos aparecerán las bicicletas disponibles para una fecha determinada, la cual se deberá seleccionar para el filtrado.

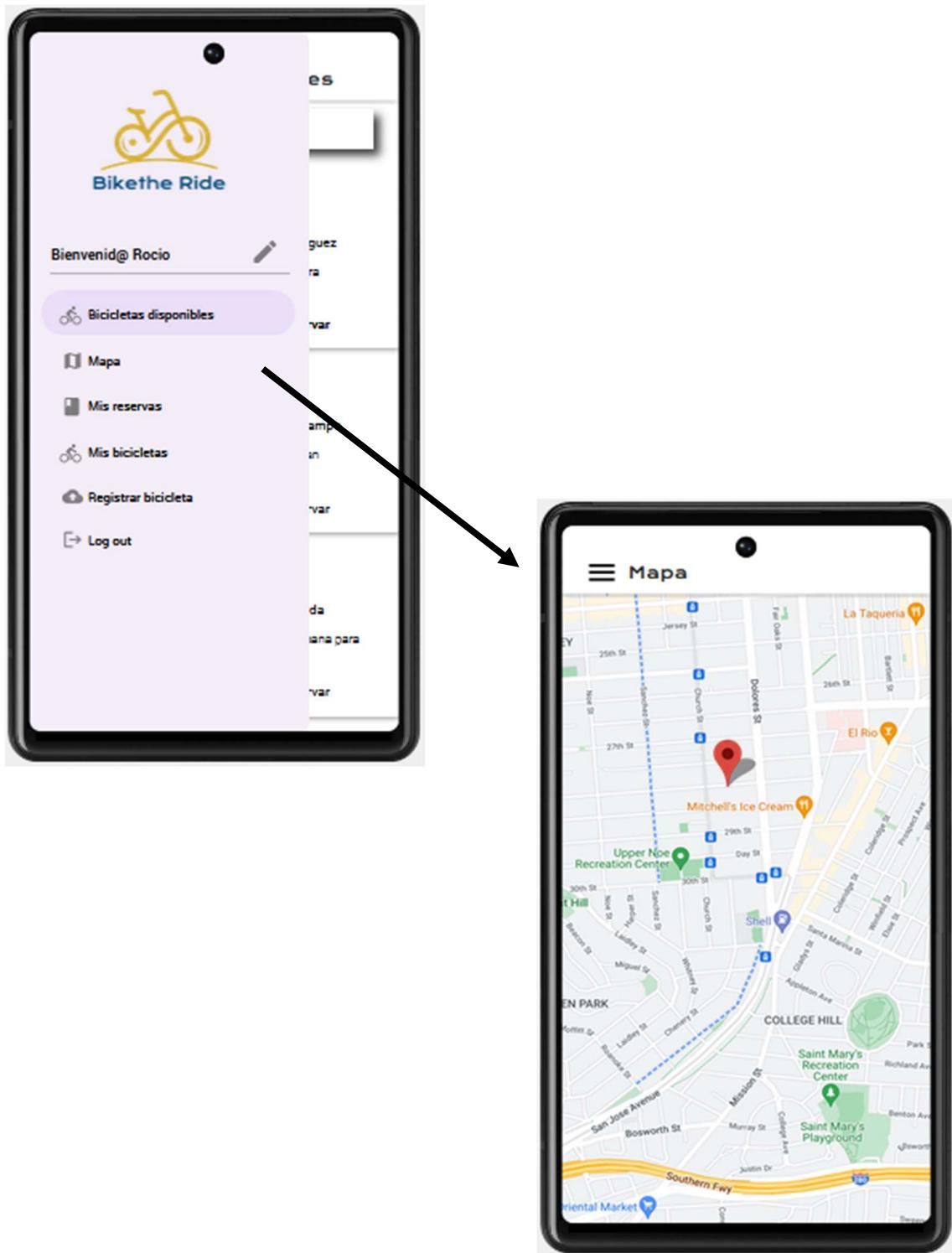
Antes de entrar en detalles, hay que mencionar el ícono dedicado a abrir el menú, en la parte superior izquierda, en el que aparecerán las diferentes secciones además del cierre de sesión, así como una cabecera con el logo de la aplicación y un mensaje de bienvenida con nuestro nombre y el ícono para modificar el perfil.

Siguiendo hablando de la pantalla de bicicletas disponibles, aparecerá todo el listado con su foto, la localización, el nombre del dueño, una pequeña descripción y, un botón para reservar la bicicleta y enviar un email al dueño con la fecha seleccionada y el nombre del arrendatario.



- Mapa

En esta pantalla se abrirá un mapa en el que de forma visual nos aparecerán la posición de las bicicletas.

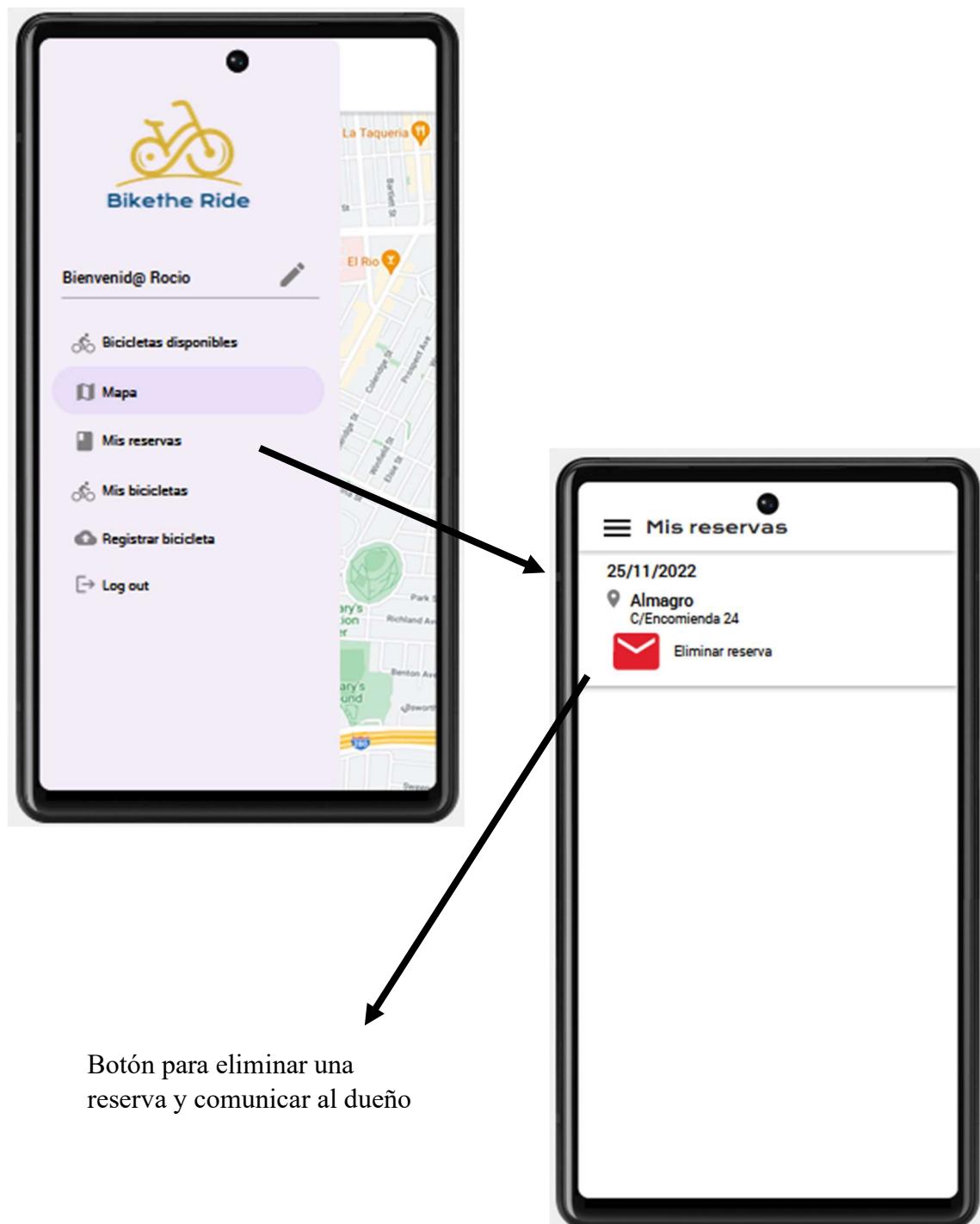


- Mis reservas

Seleccionando la opción de mis reservas, aparecerá una nueva pantalla en la que, si tenemos alguna reserva realizada, se mostrarán aquí.

De una reserva se podrá ver la fecha y la localización.

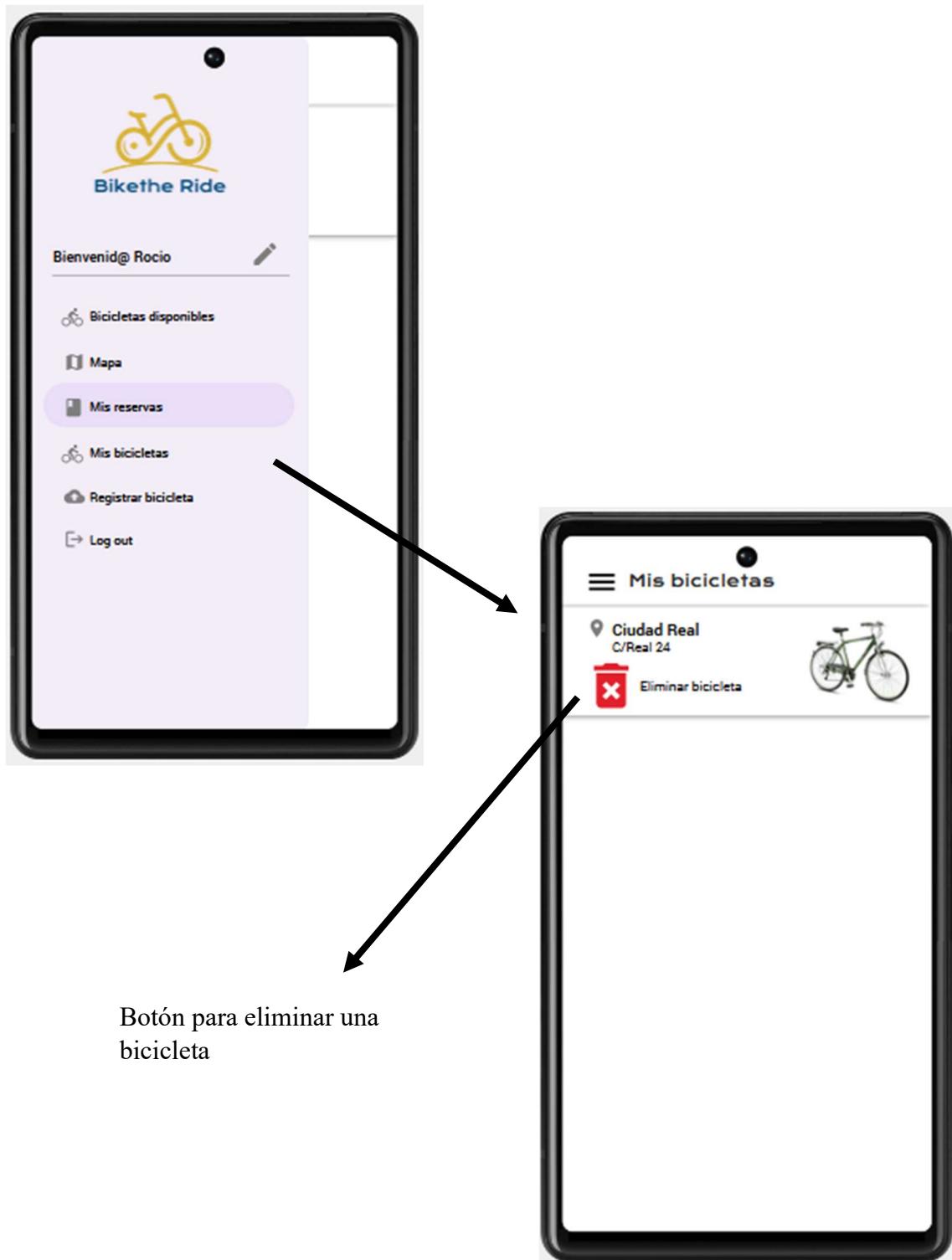
También habrá un botón para eliminar la reserva, el cual además mandará un correo electrónico al dueño de la bicicleta comunicándole el desistimiento.



- Mis bicicletas

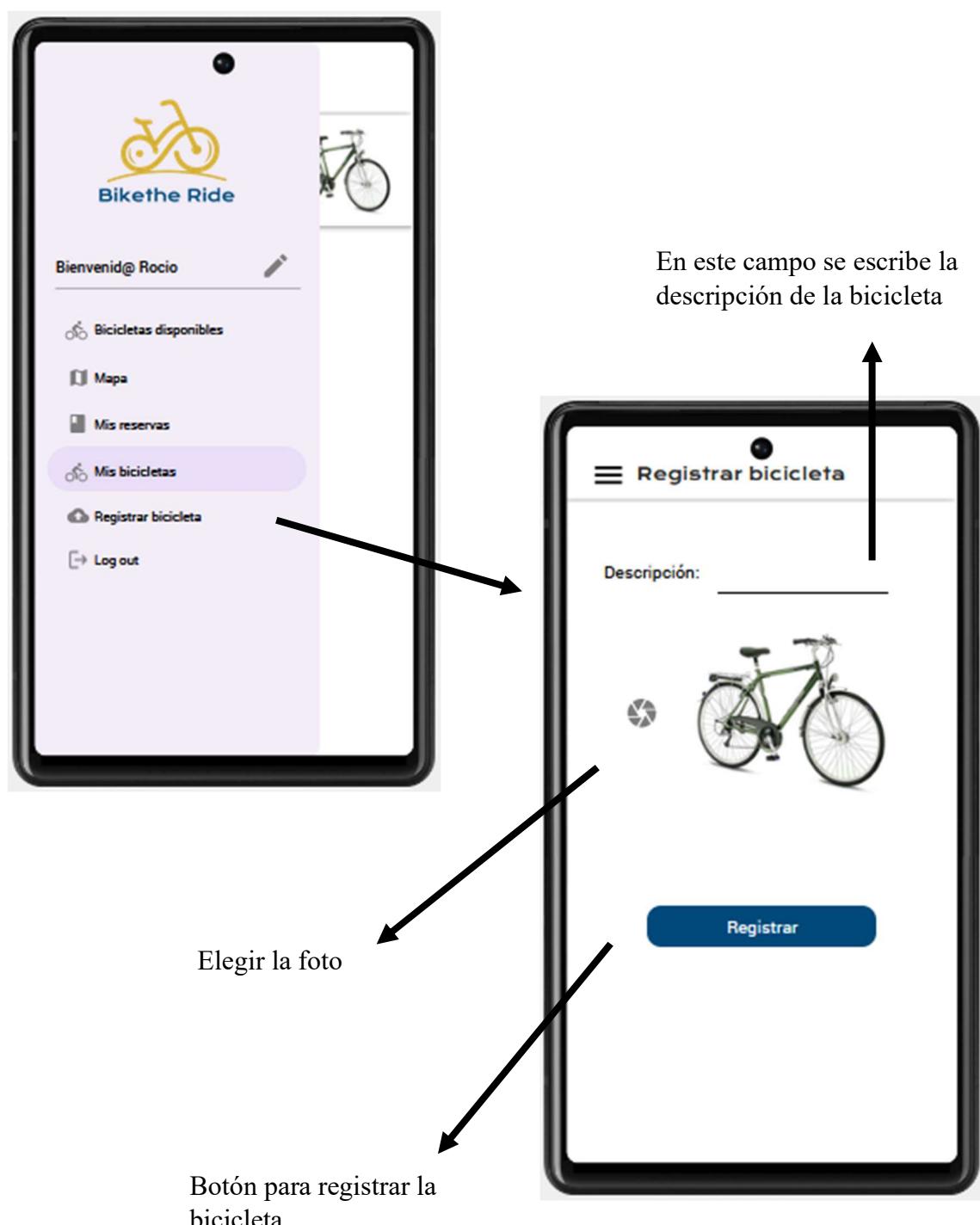
En esta sección se podrán ver las bicicletas que tenemos registradas para alquilar. Se podrá ver la foto de la bicicleta y la localización.

Además, se dispondrá de un botón para eliminar la bicicleta en caso de que no se quiera seguir alquilándola.



- Registrar bicicleta

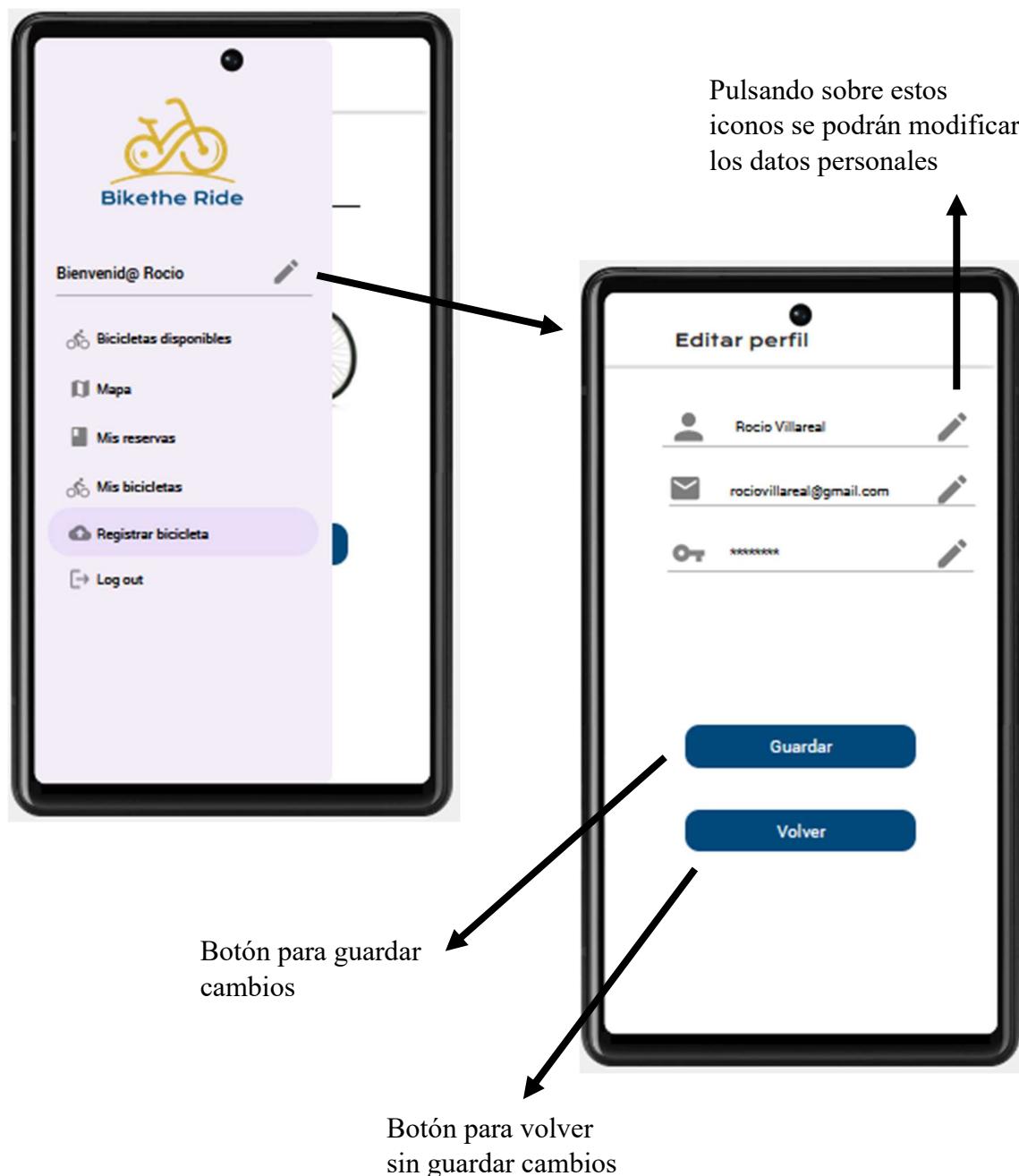
Aquí es donde se podrá registrar una bicicleta para poder ser alquilada. Habrá un campo para escribir una breve descripción, adjuntar la foto, y un botón para registrarla y que esté disponible para su alquiler. La localización se agregará automáticamente a través del GPS del dispositivo.



- Editar perfil

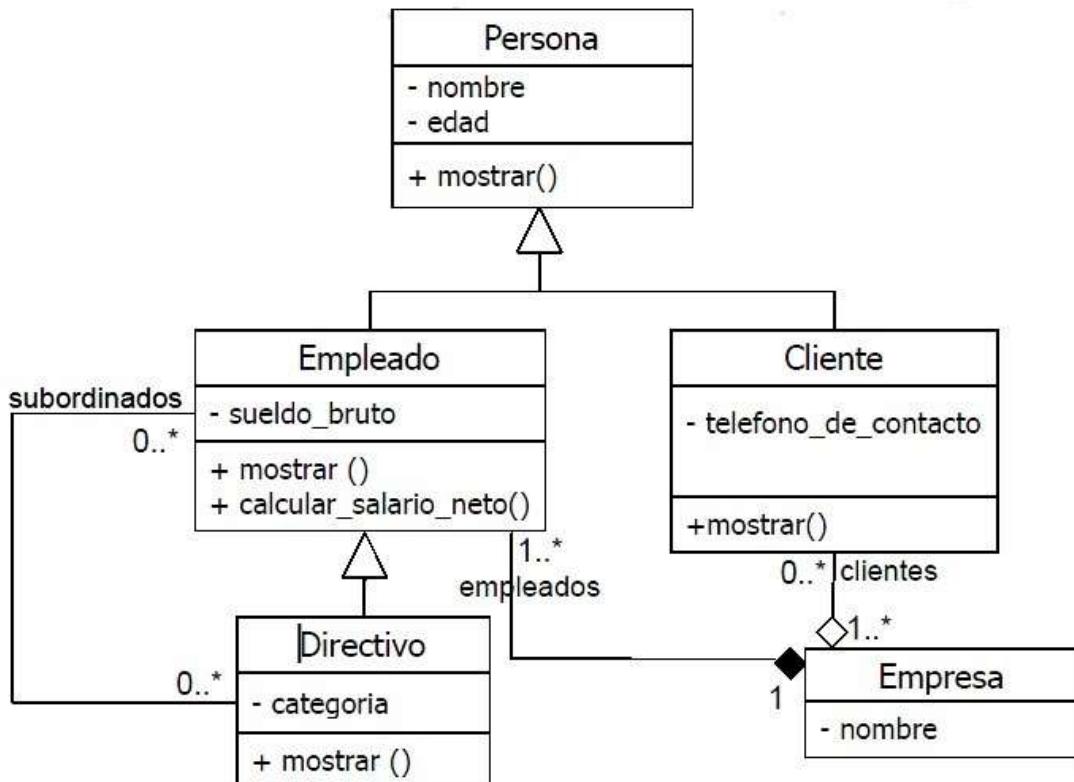
Para editar el perfil accedemos desde el ícono que está en el Navigation Drawer. Se abrirá una pantalla en la que se podrá modificar el nombre de usuario, email o contraseña.

Habrá un botón disponible para guardar los cambios, y otro para volver sin guardar.



4.2. Diagrama de clases

Un diagrama de clase es un tipo de diagrama UML que describe un sistema visualizando los diferentes tipos de objetos dentro de un sistema y los tipos de relaciones estáticas que existen entre ellos. También ilustra las operaciones y atributos de las clases.



En un diagrama de clases nos podemos encontrar:

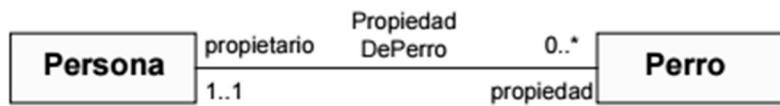
- Clases

Agrupan conjuntos de objetos con características comunes, que serán los atributos, y su comportamiento, que serán métodos.

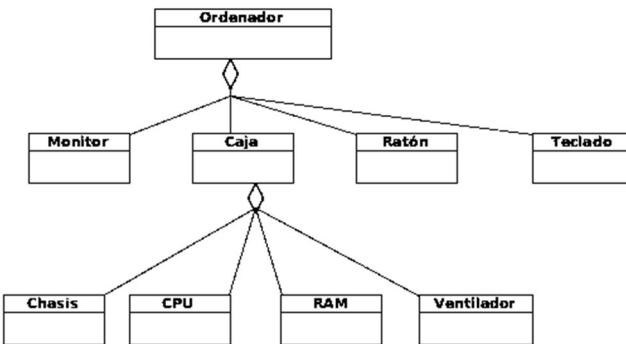
- Relaciones

Una relación es una unión entre dos clases. Pueden ser de:

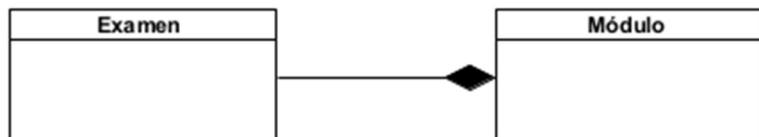
- Asociación: Va a describir un conjunto de vínculos entre las instancias de las clases.



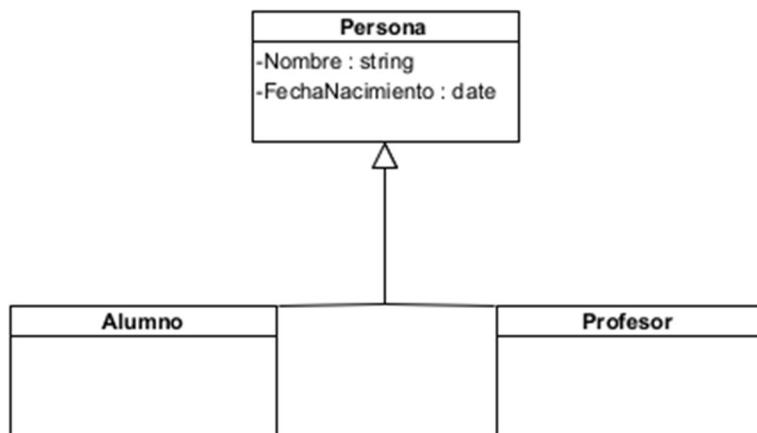
- Agregación: Representa relaciones en las que un objeto es parte de otro, pero aun así debe tener existencia en sí mismo.



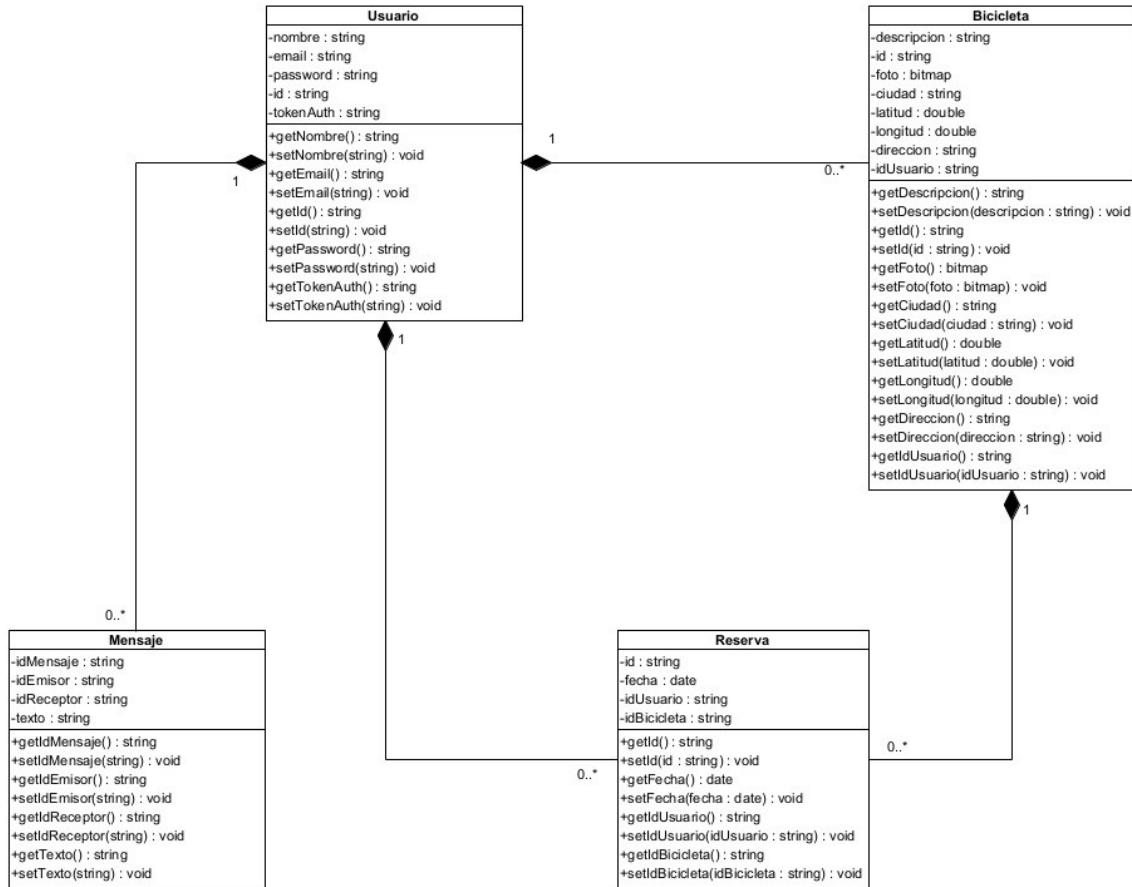
- Composición: Representa una relación jerárquica entre un objeto y las partes que lo componen. En este caso, los elementos que forman parte no tienen sentido de existencia cuando el primero no existe.



- Generalización o herencia: Una clase (clase hija o subclase) recibe los atributos y métodos de otra clase (clase padre o superclase).



El diagrama de clases quedaría de la siguiente forma:



El sistema gira en torno a la clase Usuario, en el que todas las relaciones son de composición, ya que, en caso de desaparición de dicha clase, la existencia de las demás no tiene sentido.

Las relaciones en detalle son:

- Usuario ♦— Bicicleta

La clase usuario y la clase bicicleta están relacionadas, ya que un usuario puede registrar una bicicleta.

El tipo de relación es de composición, ya que, si un usuario desaparece, la bicicleta desaparece con él.

La cardinalidad es de 1–0..*: Un usuario puede tener ninguna o muchas bicicletas, pero una bicicleta solo pertenece a un usuario.

- Usuario ♦— Reserva —♦ Bicicleta

Los usuarios reservan las bicicletas.

El tipo de relación es de composición, con lo que, si el usuario o la bicicleta desaparece, desaparece la reserva.

La cardinalidad es de 1–0..*: Un usuario puede tener ninguna o muchas reservas, pero una reserva solo pertenece a un usuario. Una bicicleta puede tener ninguna o muchas reservas, pero una reserva solo pertenece a una bicicleta.

- Usuario ♦— Mensaje

La clase usuario y la clase mensaje están relacionadas, ya que un usuario puede enviar o recibir mensajes.

El tipo de relación es de composición, ya que, si un usuario desaparece, sus mensajes desaparecen con él.

La cardinalidad es de 1–0..*: Un usuario puede enviar o recibir ningún o muchos mensajes, pero un mensaje es enviado o recibido por un usuario.

4.3. Diseño de la persistencia de la información

La persistencia es la acción de mantener la información del objeto de una forma permanente (guardarla), pero también debe de poder recuperarse dicha información para que pueda ser utilizada nuevamente.

La pregunta es, ¿estructura Relacional o NoSQL?

El modelo de base de datos SQL se distingue por su eficiencia a la hora de organizar la información. Funciona con un lenguaje estructurado de consulta.

El modelo NoSQL no necesitan un esquema fijo y son fácilmente modulares.

Ambas tecnologías tienen el mismo objetivo que es almacenar datos, pero lo hacen de maneras muy diferentes. Debido a la flexibilidad y escalabilidad de las bases de datos NoSQL y que están especialmente recomendadas para aplicaciones móviles, nos decantaremos por la base de datos NoSQL de Google Firebase.

Las bases de datos NoSQL (Not Only SQL) aparecen debido a la necesidad de flexibilidad para almacenar distintos tipos de información no estructurada como documentos (PDF,

Word, Excel...), emails, SMS, localizaciones geográficas, audio, vídeo, publicaciones de RRSS... Cuando este volumen de datos es muy grande, es lo que se conoce como Big Data.

El Big Data, al tratarse de unas cantidades de información muy elevadas, necesita de arquitecturas o esquemas de BBDD con capacidades de alta escalabilidad, rápidas, etc. que las bases de datos SQL no pueden proporcionar.

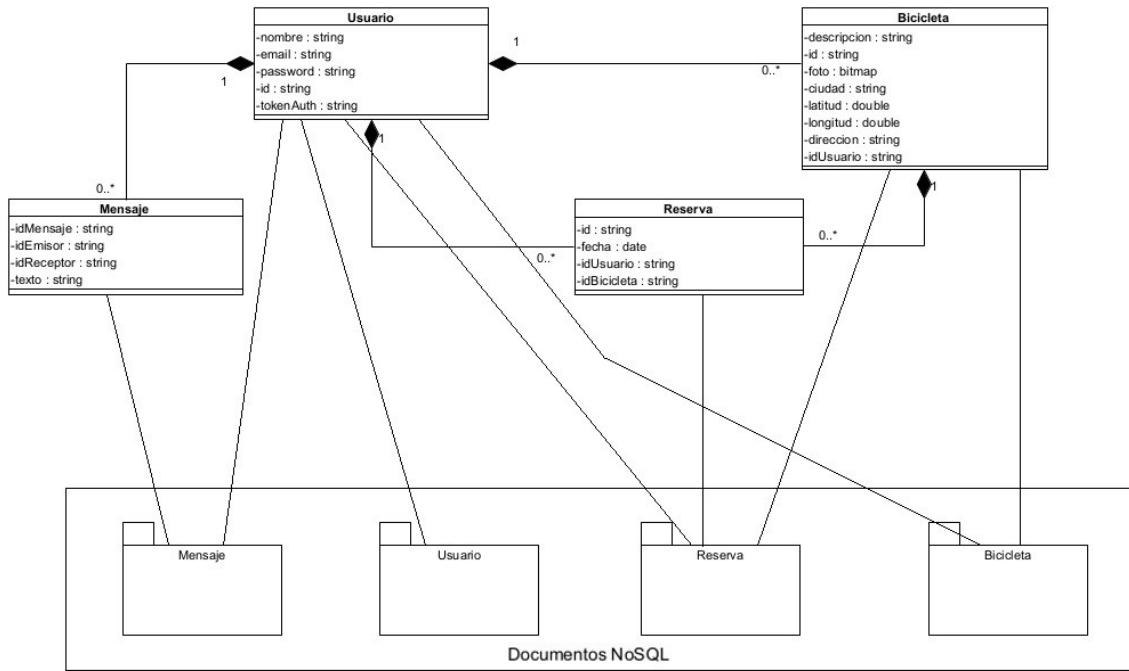
Las ventajas de las bases de datos NoSQL son:

- Aplicaciones de big data: grandes volúmenes son manejados fácilmente.
- Administración de la base de datos: Requieren menos administración práctica, cuenta con capacidades de distribución de datos y reparación automática, modelos de datos simplificados y menos requisitos de ajuste y administración.
- Versatilidad: Las posibilidades de crecimiento en el volumen de datos o la posibilidad de incluir cambios sobre la forma en la que ingresan los datos sin necesidad de alterar la estructura, permite adaptarse de forma rápida a un entorno de alto dinamismo.
- Crecimiento Horizontal: Son altamente escalables, si se requiere instalar mayor cantidad de nodos para ampliar la capacidad, se puede hacer sin problemas. Esto no interrumpe la usabilidad o consultas dentro de la base de datos.
- Economía: No se necesitan servidores con gran cantidad de recursos para operar. La adaptabilidad y flexibilidad permiten empezar con bajos niveles de inversión en equipos e ir ampliando la capacidad a medida de las necesidades.

Conclusión

Una vez visto las ventajas, podemos corroborar nuestra elección con varios aspectos como la escalabilidad, ya que si la aplicación va creciendo con el tiempo es fácil agregar más servidores y los datos se pueden distribuir entre estos. Además, como estamos en los principios del desarrollo, tenemos la flexibilidad de poder cambiar los tipos de datos. También la velocidad puede ser mayor y la latencia menor, al estar los datos distribuidos por muchos nodos.

A continuación, se mostrará el diagrama de persistencia de la información, en el que se describen las clases y relaciones en la parte superior, y los documentos que se crearan en la base de datos en la parte inferior.



4.4. Arquitectura del sistema

En primer lugar, diremos que Firebase es un BaaS (Backend as a Service). Para comprender esto, vamos empezar definiendo el concepto de serverless.

Una arquitectura serverless (sin servidor), significa que no hay una administración de los servidores por los desarrolladores, sino que los servidores son administrados por proveedores como Amazon, Google, IBM, etc.

La arquitectura Serverless depende significativamente de servicios de terceros (conocido como Backend como servicio o “BaaS”) o de código personalizado que se ejecuta en contenedores efímeros (función como servicio o “FaaS”).

El modelo BaaS proporciona a los desarrolladores una forma de vincular las aplicaciones al almacenamiento en nube (cloud storage), servicios analíticos y/o otras características tales como la gestión de usuarios, la posibilidad de enviar notificaciones push y la integración con servicios de redes sociales. Estos servicios se prestan a través de la utilización de kits personalizados de desarrollo de software (SDK) y las interfaces de programación de aplicaciones (API).

Los proveedores BaaS forman un puente entre el frontend de una aplicación y varios servicios backend en la nube a través de una API y SDK unificada.

Firebase es un claro ejemplo de BaaS.

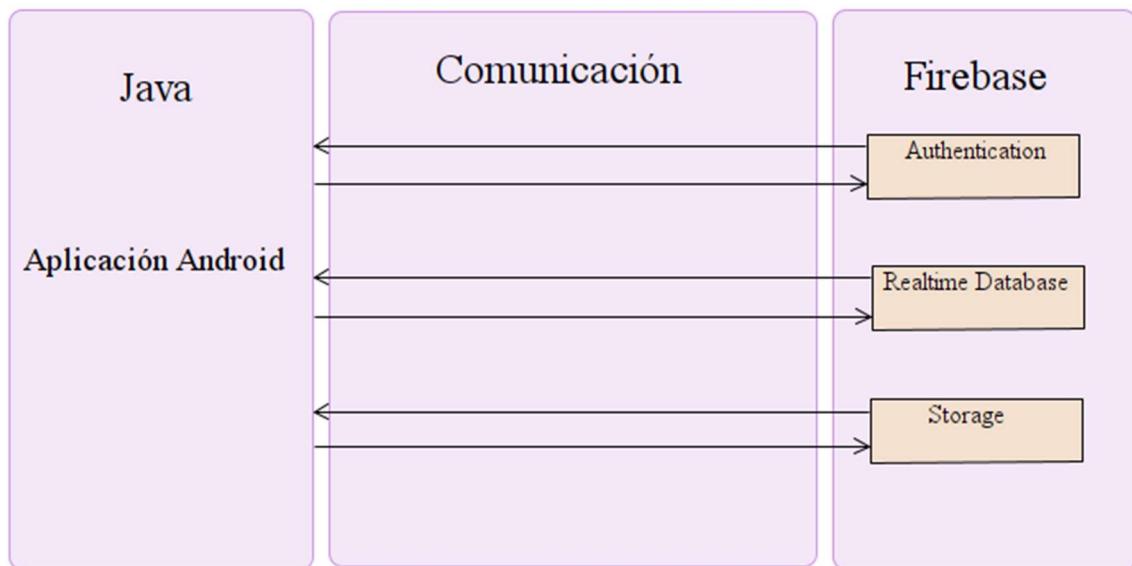


Como la finalidad de un Baas es facilitar el desarrollo de aplicaciones, algunas herramientas que debe ofrecer el proveedor son:

- SDK: Librería de fácil integración y uso que nos dé acceso a las funciones proporcionadas por el BaaS.
- Analíticas: Monitorización de propiedades del usuario y eventos de uso.
- Crash reporting: Necesitamos estar informados de cuándo y dónde surgen los problemas.
- Servicios de autenticación: Nos debe permitir la autenticación basada en usuario (correo electrónico) / password, OAuth integrado con Google, Facebook, Twitter, etc. O integrar nuestro propio sistema de autenticación.
- Almacenamiento: Espacio donde guardar y compartir ficheros, vídeo, imágenes, documentos.
- Base de datos: Almacenamiento NoSQL de fácil uso y que permita la sincronización bidireccional.
- Push notifications: Nos permite enviar notificaciones a la app cuando necesitamos comunicar algo al usuario o sincronizar algún dato.
- Funciones en la nube: Automatización de tareas. Disparo de notificaciones en función de triggers, batch jobs periódicos.

- Dashboard: Control de todo el BaaS, desde la creación de usuarios o los modelos de base de datos a la monitorización de las estadísticas.

El diagrama de la aplicación quedaría:



El servidor Firebase a través de diferentes servicios hace uso de sus métodos para la comunicación. Los servicios que se utilizarán son:

- Firebase Auth: Incluye la autenticación mediante proveedores de inicio de sesión, así como mediante correo electrónico y contraseña.
- Realtime Database: Proporciona una base de datos en tiempo real, back-end y organizada en forma de árbol JSON.
- Firebase Storage: Proporciona cargas y descargas seguras de archivos. Se puede utilizar para almacenar imágenes, audio, vídeo, o cualquier otro contenido generado por el usuario. Firebase Storage se basa en el almacenamiento de Google Cloud Storage.

La parte del cliente será desarrollada con Java.

5. IMPLEMENTACIÓN DE LA SOLUCIÓN

5.1. Análisis tecnológico

A continuación, se van a mostrar una serie de tecnologías con las que poder desarrollar la aplicación, empezando por la parte cliente y continuando por la parte servidor. Finalmente se expondrán las soluciones elegidas.

- **Tecnología cliente**

Xamarin

Xamarin es un kit de herramientas de desarrollo de aplicaciones multiplataforma, con la particularidad de poder desarrollar aplicaciones que funcionen en cualquier tipo de dispositivo móvil (iOS, Android o Windows) con el mismo código de programación, escrito en el lenguaje C# con el framework .NET.

El mayor beneficio es la capacidad que tiene la plataforma para que el desarrollador escriba la aplicación móvil en lenguaje C#, y el mismo código se pueda ejecutar en diferentes sistemas, manteniendo todas las capacidades de una aplicación nativa.

Como desventajas podemos destacar problemas de compatibilidad con librerías y herramientas de terceros, y tamaño de aplicación más grande.

Flutter

Flutter es un SDK de código fuente abierto de desarrollo de aplicaciones móviles creado por Google, que está construido en C y C++ para programar en lenguaje Dart.

Está pensado para agilizar el proceso de desarrollo de aplicaciones móviles para Android e iOS pero a través de una misma base de código, además de reducir los costes y permitir crear apps de forma más rápida.

El desarrollo de aplicaciones móviles con Flutter permite su personalización completa gracias a la arquitectura en capas. Además, ofrece una alta calidad en la experiencia de usuario respetando a su vez las características de diseño de los dispositivos Android e iOS.

Ha conseguido generar confianza y seguridad en los desarrolladores, ya que el hecho de que Google esté detrás de Flutter permite un cierto grado de confianza al conocer la experiencia que tienen en tecnología y desarrollo de software. Además, con el auge que

está experimentando Flutter en los últimos meses, las actualizaciones de las librerías y los widgets, así como el mantenimiento es algo constante.

Flutter hace que el rendimiento de las apps que se desarrollan con este lenguaje sea prácticamente nativo.

Como desventajas podemos citar que al ser un framework bastante reciente, implica que el desconocimiento y la experiencia que se tiene en su uso es mucho menor que con otros sistemas de desarrollo. Además del mayor peso de la aplicación, y la dificultad para integrarse con librerías de terceros nativas.

Java

Java es uno de los lenguajes de programación más populares del mundo. Es un lenguaje orientado a objetos, potente, versátil y multiplataforma. Además, se puede obtener Java y gran cantidad de herramientas para trabajar con él de forma gratuita, siendo la mayor parte de su código libre y abierto.

Existe una comunidad extensa, muchas bibliotecas, soluciones y módulos listos para usar. En cuanto al código, comparado con otros lenguajes, puede llegar a ser más extenso y repetitivo.

Ionic

Ionic Framework es un SDK de frontend de código abierto para desarrollar aplicaciones híbridas basado en tecnologías web (HTML, CSS y JS). Es decir, un framework que nos permite desarrollar aplicaciones para iOS nativo, Android y la web, desde una única base de código. Se integra con los principales frameworks de frontend, como Angular, React y Vue, aunque también se puede usar Vanilla JavaScript. Ha sido una de las tecnologías líderes para el desarrollo de aplicaciones móviles híbridas hasta la llegada de React Native.

Como ventajas podemos destacar la facilidad de aprender y utilizar, buena documentación y respaldo de la comunidad, y diseño de interfaces sencillo.

Las principales desventajas son su peor rendimiento que las aplicaciones nativas, y aplicaciones más pesadas.

React Native

React Native, es un framework de código abierto creado por Meta Platforms y basado en JavaScript y ReactJS, para desarrollar aplicaciones multiplataforma.

Como punto positivo para el crecimiento, desarrollo y mantenimiento es que hay una gran comunidad de desarrolladores usándolo.

El rendimiento es muy similar a si se desarrolla en nativo, aunque esta algo por debajo.

Como contra podemos destacar que hay pocos componentes, aunque como tiene una gran comunidad detrás, tiene fácil solución.

Kotlin

Kotlin es un lenguaje de programación de tipado estático que corre sobre la máquina virtual de Java y que también puede ser compilado a código fuente de JavaScript.

Aunque no tiene una sintaxis compatible con Java, está diseñado para interoperar con código Java y es dependiente del código Java de su biblioteca de clases.

Una de las principales ventajas es que el código es simple y se necesita menos para realizar las funciones. Además, tiene una comunidad muy grande.

Como desventajas podemos destacar el binding de las vistas, que, para obtener una referencia a una vista en Android, tienes que realizar manualmente el binding de los componentes, generando mucho más código.

- **Tecnología servidor**

Back4App

Back4app es una plataforma que proporciona servicios fáciles de usar para crear aplicaciones de Android. La mejor parte aquí es que se basa en tecnología de código abierto. Por lo tanto, ayuda a los desarrolladores a crear aplicaciones donde el backend de las aplicaciones está completamente alojado.

Características principales:

- Escalabilidad
- Backend flexible según los requisitos.
- De código abierto

Firebase

Firebase es una de las plataformas de backend mejor administradas para aplicaciones web y especialmente para dispositivos móviles. Esta plataforma es proporcionada por Google, por lo que la seguridad y una mejor administración son uno de los aspectos más destacados de Firebase.

Características principales:

- Sincronización automática de datos
- Análisis
- Servicios de autenticación

Parse

Parse es una de las plataformas BaaS líderes con una gran comunidad de desarrolladores trabajando en ella. Es una plataforma de código abierto para que todos los desarrolladores puedan usarla para sus necesidades de desarrollo de aplicaciones.

Características principales:

- Modelado de datos
- Notificaciones push y en tiempo real
- Inicio de sesión social

Kumulos

Kumulos es una plataforma que facilita las cosas a los desarrolladores al proporcionar todas las funciones necesarias en una plataforma. Viene con algunas de las características más interesantes y únicas como un servicio de backend.

Características principales:

- Análisis de campaña
- Seguimiento de la opinión de los usuarios
- Mensajería en la aplicación
- Enlaces profundos diferidos

AWS Amplify

AWS Amplify es un backend administrado para el proceso de desarrollo de aplicaciones proporcionado por Amazon. Lo mejor de esta plataforma es que puede ponerla en marcha en cuestión de minutos y todo el proceso lleva unos segundos.

Características principales:

- Gestión sencilla
- Increíbles CLI y bibliotecas para características
- Integración de inteligencia artificial y aprendizaje automático

Conclusión

Una vez vistas y comentadas a grandes rasgos las tecnologías de lado cliente y servidor, se ha decidido desarrollar Bikethe Ride con Java y Firebase.



Java es uno de los lenguajes más utilizados para el desarrollo de aplicaciones (el tercero según el índice TIOBE). Es un lenguaje que se adapta al crecimiento de la aplicación, ya que es de código abierto, lo que permite crear código reutilizable y proyectos modulares.

Tiene una gran biblioteca de código abierto, con lo que nos ayudan a desarrollar la aplicación de una forma más rápida. Es un lenguaje seguro y robusto.

Por no hablar de la gran comunidad que tiene este lenguaje, la cantidad de desarrolladores que hay, y los años de experiencia.

Utilizando Firebase para el backend nos facilita mucho, ya que de forma sencilla podemos usar diferentes servicios que nos ofrece esta plataforma. Podemos hacer uso de Realtime que es una base de datos en tiempo real. Un servicio de autenticación fácil de integrar como Firebase Authentication, con el que no solo usando el correo electrónico y la contraseña podemos verificar nuestra identidad, sino que también con cuenta de Google o Twitter entre muchos otros. Además de un almacén de datos fácil y rápido como Firebase Storage.

Como todo comienzo de proyecto es difícil, con Firebase tenemos la mayoría de las funciones de forma gratuita. En el caso de que nuestra aplicación fuese un éxito, es Firebase el que se encargará del escalamiento.

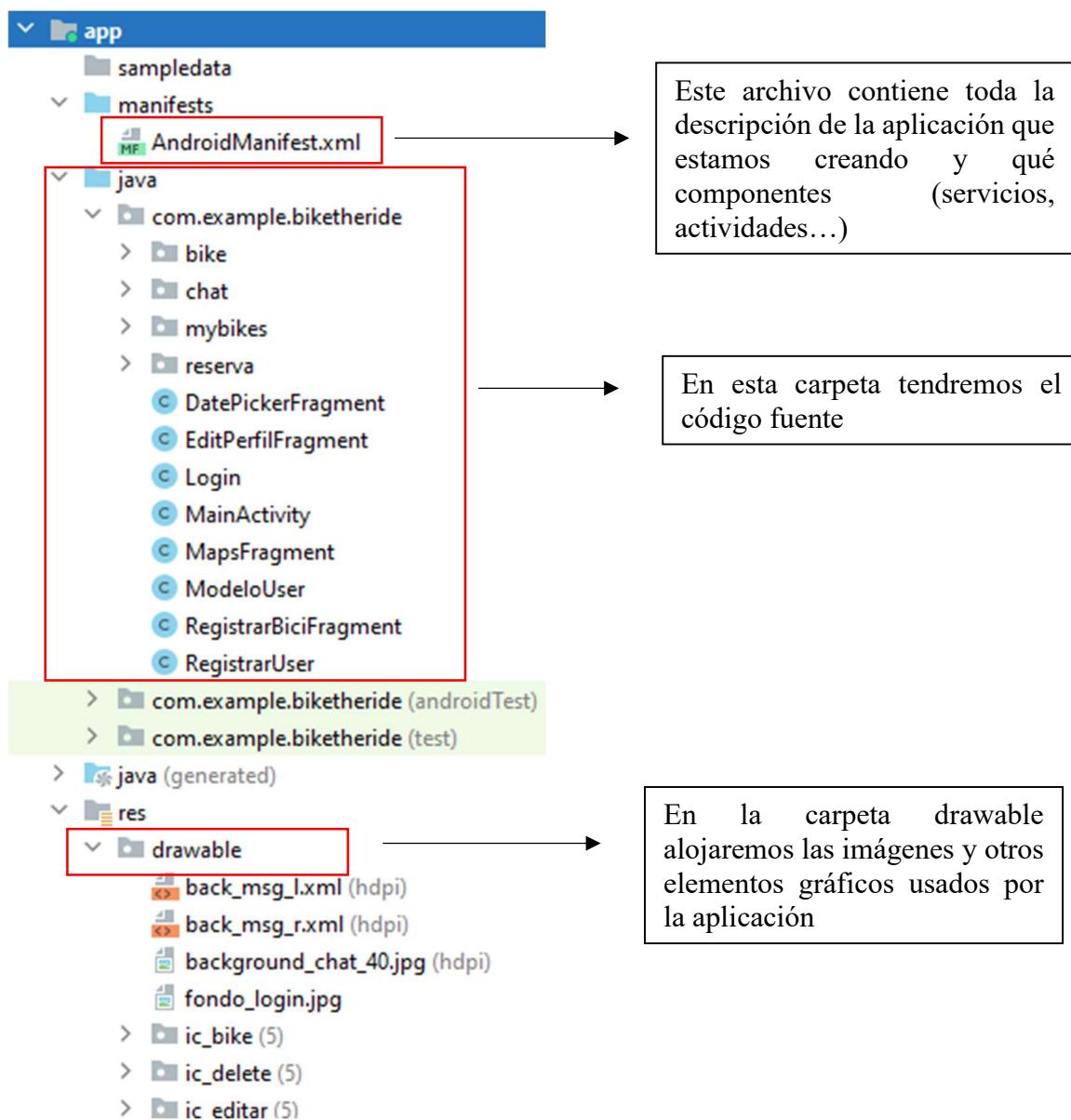
Con la experiencia de Java y la integración de Firebase con Android, no hay duda de que el resultado sea todo un éxito.

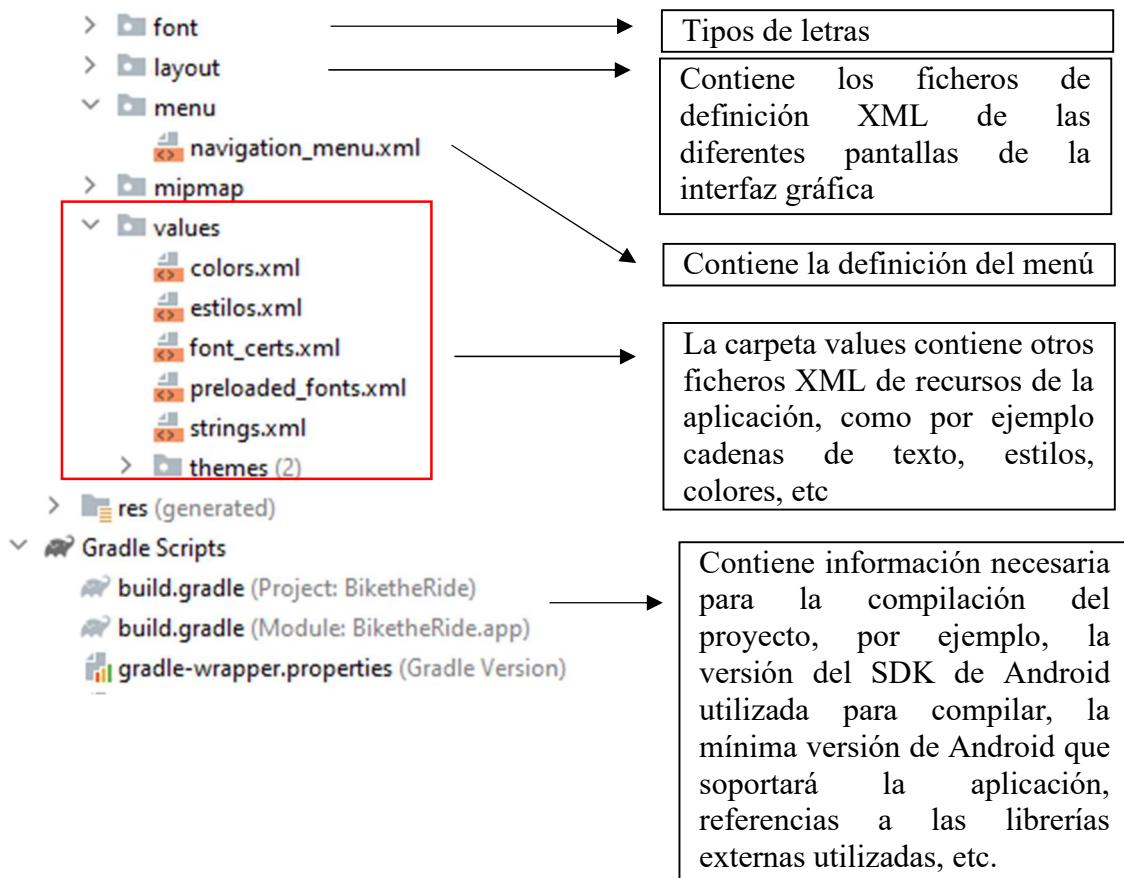
5.2.Elementos a implementar.

Las funcionalidades a implementar en el proyecto serán:

- Login y registro
- Modificar nombre, contraseña y dirección de email
- Registrar bicicleta con localización actual, eliminar bicicleta
- Listar una serie de bicicletas a partir de una fecha seleccionada, mapa con la localización, reservar y eliminar reserva
- Contacto entre usuarios mediante chat

A continuación, se muestra el esquema del proyecto:





- Login y registro

La primera toma de contacto del usuario con la aplicación será con la pantalla de login, desde la que se podrá acceder al registro en el caso que no se posea autenticación.

Se podrá acceder mediante cuenta de Google o con usuario y contraseña.

En caso de acceder mediante cuenta de Google, se construirá un objeto del tipo `GoogleSignInOptions` para proporcionar las características del registro de usuarios y utilizar el método `getClient` para obtener el cliente de autentificación.

```

GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestIdToken(getString(R.string.default_web_client_id))
    .requestEmail()
    .build();
mGoogleSignInClient = GoogleSignIn.getClient(activity, gso);
mGoogleSignInClient.signOut();

Intent signInIntent = mGoogleSignInClient.getSignInIntent();
startActivityForResult(signInIntent, RC_SIGN_IN);

```

Seguidamente se arrancará la actividad de autentificación y registro a través de Google. Si el usuario existe en Google, se permitirá el acceso a la aplicación. De lo contrario, se denegará. Esto se gestiona al examinar el resultado de la actividad de login, en onActivityResult.

```

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    /* Resultado devuelto al iniciar el Intent desde GoogleSignInApi.getSignInIntent(...); */
    if (requestCode == RC_SIGN_IN) {
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        try {
            /* Google Sign In fue exitoso, autenticar con Firebase */
            GoogleSignInAccount account = task.getResult(ApiException.class);
            Log.d(TAG, msg: "firebaseAuthWithGoogle:" + account.getId());
            firebaseAuthWithGoogle(account.getIdToken());

        } catch (ApiException e) {
            /* Falló el inicio de sesión de Google */
            Log.w(TAG, msg: "Google sign in failed", e);
        }
    }
}

```

Si el login se hizo con éxito, se procede a lanzar la siguiente actividad, de lo contrario, no se permite el acceso a esa segunda actividad. También se añadirá el usuario a la base de datos, en caso de que no exista.

```

private void firebaseAuthWithGoogle(String idToken) {
    AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);
    mAuth.signInWithCredential(credential).addOnCompleteListener(activity: this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                Inicio de sesión correcto
                Log.d(TAG, msg: "signInWithCredential:success");
                DatabaseReference mDatabase = FirebaseDatabase.getInstance("https://biketheride-d83a4-default-rtdb.firebaseio.com");

                mDatabase.child("user").child(mAuth.getCurrentUser().getUid()).addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        if (!snapshot.exists()){
                            ModeloUser user=new ModeloUser(mAuth.getCurrentUser().getDisplayName(),mAuth.getCurrentUser().getEmail(),mAuth.getCurrentUser().getUid());
                            user.addToDatabase(mAuth.getCurrentUser().getUid());
                        }
                        irMain(task.getResult().getUser().getEmail());
                    }
                });
            }
        }
    });
}

```

Usando el método `getReference` del objeto `FirebaseDatabase` podemos obtener una referencia a la colección. Una referencia es un objeto de la clase `DatabaseReference` que apunta a un nodo de la base de datos. A partir de esa referencia podemos hacer la operación de escritura.

Una vez creado el modelo de usuario, se accede al método `addToDatabase()` de dicha clase, y se utiliza el método `setValue` de la referencia.

```

public void addToDatabase(String uid){
    DatabaseReference database= FirebaseDatabase.getInstance()
    database.child("user/"+uid).setValue(this);
}

```

En el caso del registro mediante correo electrónico y contraseña, el usuario se añadirá a la base de datos de la misma forma.

Veamos a continuación el registro y login mediante correo electrónico y contraseña.

Cuando un nuevo usuario se registre mediante el formulario de registro de la aplicación, y complete todos los pasos de validación, se crea una nueva cuenta pasando la dirección de correo electrónico y la contraseña del nuevo usuario al método `createUserWithEmailAndPassword()` de la instancia del objeto `FirebaseAuth`.

```

binding.btRegistro.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String emailUser = binding.etEmail.getText().toString().trim();
        String passUser = binding.etPassword.getText().toString().trim();
        String nombreUser = binding.etNombre.getText().toString().trim();

        if (emailUser.isEmpty() || passUser.isEmpty() || nombreUser.isEmpty()) {
            Toast.makeText(context: RegistrarUser.this, text: "Complete los datos", Toast.LENGTH_SHORT).show();
        } else {
            registrarUser(nombreUser, emailUser, passUser);
        }
    }
}

private void registrarUser(String nombreUser, String emailUser, String passUser) {
    mAuth.createUserWithEmailAndPassword(emailUser, passUser).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {

                ModeloUser user=new ModeloUser(nombreUser,mAuth.getCurrentUser().getEmail(),mAuth.getCurrentUser()
                user.addToDatabase(mAuth.getCurrentUser().getUid());
                Toast.makeText(context: RegistrarUser.this, text: "Usuario registrado correctamente", Toast.LENGTH_LONG).show();
                finish();
            } else {
                AlertDialog.Builder builder = new AlertDialog.Builder(context: RegistrarUser.this);

                builder.setTitle("Error de registro");
                builder.setMessage("Comprueba que el correo electrónico sea válido y la contraseña tenga como mínimo 6 dígitos");
                builder.setPositiveButton(text: "Aceptar", listener: null);

                AlertDialog dialog = builder.create();
                dialog.setCanceledOnTouchOutside(false); // para hacerlo modal
                dialog.show();
            }
        }
    });
}

```

En caso de que la tarea no se complete satisfactoriamente, mostrara un mensaje de error con una serie de acciones para solventarlo.

Para iniciar sesión los pasos son parecidos que para crear la nueva cuenta. Cuando el usuario inicie sesión en la aplicación, se pasa la dirección de correo electrónico y la contraseña del usuario al método signInWithEmailAndPassword() de la instancia del objeto FirebaseAuth.

```

mAuth.signInWithEmailAndPassword(binding.etEmail.getText().toString(), binding.etPassword.getText().toString());
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
            e sesión correcta
                Log.d(TAG, msg: "signInWithCredential:success");

                irMain(task.getResult().getUser().getEmail());

            } else {
                inicio de sesión
                    mostrarAlert();
                    Log.w(TAG, msg: "signInWithCredential:failure", task.getException());
            }
        }
    });
}

```

Con el inicio de sesión correcto, se iniciará la “actividad principal” MainActivity. Como forma de navegación por la aplicación se usará es un menú deslizante Navigation Drawer. Dicha actividad cargará el menú de navegación y el layout en el que se cargarán los fragments, que serán las diferentes secciones del menú.

- Modificar nombre, contraseña y dirección de email

Desde la cabecera del menú, habrá disponible un ícono para editar el nombre, email y contraseña del usuario.

Una vez obtenida la referencia al nodo de la base de datos (DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference();), se puede usar el método child(), que devuelve una referencia relativa a un nodo dato.

```

public void obtenerDatos() {

    mDatabase.child("user").child(mauth.getUid()).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            String tvnombre = snapshot.child("name").getValue().toString();
            String tvemail = snapshot.child("email").getValue().toString();
            binding.etEditName.setHint(tvnombre);
            binding.etEditEmail.setHint(tvemail);
        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {
        }
    });
}

```

Para leer datos de la base de datos, debemos buscar la referencia que contiene esa información y poner un listener para detectar cualquier cambio de valor. Para añadir un listener de este tipo se utilizar el método `addValueEventListener()` que añade un objeto de la clase `ValueEventListener` con la implementación de dos métodos:

- `onDataChange`: Se ejecutará cuando haya algún cambio en la colección que cuelga de la referencia.
- `onDataCancelled`: Se ejecutará cuando el acceso a la base de datos falle por algún motivo relacionado con la seguridad de la base de datos o las reglas.

El objeto `dataSnapshot` contiene los datos de la ubicación especificada en la base de datos. Con el método `getValue()` del “hijo” que se deseé, obtenemos el dato.

Para modificar el correo electrónico y la contraseña del usuario, se llamará a los métodos `updateEmail()` y `updatePassword()` del objeto `FirebaseAuth` (usuario autenticado).

```
private void firebaseEmail(String email) {
    mAuth.getCurrentUser().updateEmail(email).addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void unused) {
            FirebaseDatabase.child("user").child(mAuth.getCurrentUser().getUid()).child("email").setValue(email);
            Toast.makeText(getApplicationContext(), text: "Email actualizado correctamente", Toast.LENGTH_SHORT).show();
            binding.etEditEmail.setText("");
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            String mensaje = "Error al actualizar el email. Reinicia la sesión o prueba con otra dirección de correo electrónico.";
            alertError(mensaje);
        }
    });
}

private void firebasePass(String pass) {
    mAuth.getCurrentUser().updatePassword(pass).addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void unused) {
            Toast.makeText(getApplicationContext(), text: "Contraseña actualizada correctamente", Toast.LENGTH_SHORT).show();
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            String mensaje = "Error al actualizar la contraseña. Reinicia sesión.";
            alertError(mensaje);
        }
    });
}
```

En el caso de que la actualización del email sea satisfactoria, se procederá a establecer el nuevo valor en la base de datos con el método `setValue()`.

- Registrar bicicleta con localización actual, eliminar bicicleta

Para registrar una nueva bicicleta, se necesitarán añadir los permisos necesarios para acceder a la ubicación en el archivo AndroidManifest.xml.

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Se va a necesitar una clave de API para acceso a los servicios de ubicación de Google. Si no se dispone de una, se puede crear accediendo a la consola de Google Cloud Platform. Además, se deberá activar la opción Maps SDK for Android en la biblioteca de APIs de del proyecto, si se quiere acceder a los servicios de mapas.

Para obtener la ubicación del dispositivo, se necesitará la clase LocationServices. Con el método getFusedLocationProviderClient.

A continuación, se solicitarán los permisos para acceder a la ubicación e invocar al método getLocation() cuando se obtengan los permisos.

```
fusedLocationClient = LocationServices.getFusedLocationProviderClient(getContext());

//verifica permiso de ubicación y lo solicita al usuario cuando sea necesario
if (ContextCompat.checkSelfPermission(getContext(), Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ContextCompat.checkSelfPermission(getContext(), Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    //mostrar el diálogo de permisos del sistema, se llama al método launch() de la instancia de ActivityResultLauncher
    locationPermissionRequest.launch(new String[]{
        Manifest.permission.ACCESS_FINE_LOCATION,
        Manifest.permission.ACCESS_COARSE_LOCATION
    });
} else {
    getLocation();
}

//respuesta de permisos
ActivityResultLauncher<String[]> locationPermissionRequest =
    registerForActivityResult(new ActivityResultContracts
        .RequestMultiplePermissions(), result -> {
            Boolean fineLocationGranted = result.getOrDefault(
                Manifest.permission.ACCESS_FINE_LOCATION, defaultValue: false);
            Boolean coarseLocationGranted = result.getOrDefault(
                Manifest.permission.ACCESS_COARSE_LOCATION, defaultValue: false);
            getLocation();
            if (fineLocationGranted != null && fineLocationGranted && coarseLocationGranted != null && coarseLocationGranted) {
                getLocation();
                // Precise location access granted.
            } else if (coarseLocationGranted != null && coarseLocationGranted) {
                getLocation();
                // Only approximate location access granted.
            } else {
                Toast.makeText(getContext(), text: "Se necesita permiso de ubicación para conocer las coordenadas de la bicicleta", Toast.LENGTH_SHORT).show();
                // No location access granted.
            }
        }
    );
}
```

Una vez que se haya creado el cliente de servicios de ubicación, se podrá obtener la ubicación más reciente del dispositivo usando el método getLastLocation().

El método getLastLocation() muestra un elemento Task que se puede usar para obtener un objeto Location con las coordenadas de latitud y longitud de una ubicación.

```

private void getLocation() {
    fusedLocationClient.getLastLocation()
        .addOnSuccessListener(getActivity(), new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                // Se obtiene la ultima ubicación conocida. Puede ser nulo
                if (location != null) {
                    latitude = location.getLatitude();
                    longitude = location.getLongitude();

                    //Geocodificación inversa: consiste en convertir una latitud y longitud de coordenadas geográficas en una dirección
                    Geocoder geocoder= new Geocoder(getApplicationContext(), Locale.getDefault());
                    List<Address> addresses;

                    try {
                        addresses = geocoder.getFromLocation(latitude, longitude, maxResults: 1);

                        city = addresses.get(0).getLocality();
                        country = addresses.get(0).getCountryName();
                        street = addresses.get(0).getThoroughfare() + "+" + addresses.get(0).getSubThoroughfare();

                    } catch (IOException e) {

```

Para obtener la, calle, ciudad y país se usará la geolocalización inversa, consiste en convertir una latitud y longitud de coordenadas geográficas en una dirección. Para ello se utilizará la clase Geocoder.

Para cargar la imagen en Cloud Storage, primero se crea una referencia a la ruta completa del archivo. Una vez que se haya creado la referencia adecuada, se llama al putFile() para cargar el archivo en Cloud Storage. Devuelve una UploadTask de carga que se puede usar para administrar y monitorear el estado de la carga.

```

public void cargaImagen(){
    id=mDatabase.child("bikes_list").push().getKey();
    StorageReference reference = storageRef.child(id);

    UploadTask uploadTask= reference.putFile(selectedImage);

    uploadTask.addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onProgress(@NonNull UploadTask.TaskSnapshot snapshot) {
            progressBar.setVisibility(View.VISIBLE);
            tvCargaFoto.setVisibility(View.VISIBLE);

            progressBar.setProgress((int) ((snapshot.getBytesTransferred()/snapshot.getTotalByteCount())*100));
        }
    }).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            tvCargaFoto.setText("Completado");
            image=taskSnapshot.getMetadata().getReference().toString(); //Ubicación en Storage

            registrar();
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(getApplicationContext(), text "Fallo al cargar la imagen", Toast.LENGTH_LONG).show();
        }
    });
}

```

- Listar una serie de bicicletas a partir de una fecha seleccionada, mapa con la localización, reservar y eliminar reserva

Una vez seleccionada la fecha mediante un DatePicker emergente, se mostrarán las bicicletas que no tengan ya una reserva para dicha fecha.

Se almacenarán en una lista las id de las bicicletas que tengan reserva, para después comprobar en la lista de bicicletas las que no tienen reserva, y las que no pertenecen al propio usuario.

```
private void reservaFechaSelecc(){
    mDatabase.child("reservas").orderByChild("fecha").equalTo(MainActivity.getFecha()).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            resrvFecha.clear();
            for (DataSnapshot productSnapshot : snapshot.getChildren()) {
                resrvFecha.add(productSnapshot.child("idBike").getValue().toString());
            }
            bicisFirebase();
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}

private void bicisFirebase() {
    mDatabase.child("bikes_list").addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            bicis.clear();
            for (DataSnapshot productSnapshot : snapshot.getChildren()) {
                if (!(productSnapshot.child("idUser").getValue().equals(mauth.getCurrentUser().getUid()))&&!(resrvFecha.contains(productSnapshot.ch
                    Bike bike = productSnapshot.getValue(Bike.class);
                    bicis.add(bike);

                    downloadPhoto(bike);
                }
            }
            mAdapter.notifyDataSetChanged();
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```

El gestor de almacenamiento Storage funciona también con referencias utilizando el objeto devuelto por FirebaseStorage.getInstance(). Al igual que con la base de datos RealTime Database, se pueden usar los métodos getReference para acceder a una URL y descargar el fichero. Para descargar la imagen se creará un fichero temporal que se pasa como parámetro al método getFile de la referencia a FireStore. Se añade un listener para actuar cuando se haya completado la descarga.

```

private void downloadPhoto(Bike c) {
    mStorageReference = FirebaseStorage.getInstance().getReferenceFromUrl(c.getImage());

    try {
        String timeStamp = new SimpleDateFormat( pattern: "yyyyMdd_HHmss").format(new Date());
        final File localFile = File.createTempFile( prefix: "PNG_" + timeStamp, suffix: ".png");
        mStorageReference.getFile(localFile).addOnSuccessListener<FileDownloadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
                //Insert the downloaded image in its right position at the ArrayList

                String url = "gs://" + taskSnapshot.getStorage().getBucket() + "/" + taskSnapshot.getStorage().getName();
                ;
                Log.d(TAG, msg: "Loaded " + url);
                for (Bike c : bicis) {
                    if (c.getImage().equals(url)) {
                        c.setImageBitmap(BitmapFactory.decodeFile(localFile.getAbsolutePath()));
                        mAdapter.notifyDataSetChanged();
                        Log.d(TAG, msg: "Loaded pic " + c.getImage() + ";" + url + localFile.getAbsolutePath());
                    }
                }
            }
        });
    } catch (IOException e) {

```

Al hacer click en el icono de reservar, se mostrará un mensaje de confirmación.

```

AlertDialog.Builder aDial = new AlertDialog.Builder(v.getContext());
aDial.setTitle("Reservar");
aDial.setMessage("¿Deseas reservar esta bicicleta?");
aDial.setPositiveButton( text: "Reservar", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {

        String idUser = FirebaseAuth.getInstance().getCurrentUser().getUid();
        String id = mDatabase.child("reserva").push().getKey();

        String idBike = mItem.getId();
        String city = mItem.getCity();
        String location = mItem.getLocation();

        Reserva reserva = new Reserva(id, idUser, idBike, MainActivity.getFecha(), city, location);

        addToDatabase(reserva, id, context);
        String msg = "Hola, he reservado su bicicleta con descripción " + mItem.getDescription() + ", en " + mItem.getCity() + ", " + mItem.getIdUser();
        sendMsg(msg, mItem.getIdUser());

        Toast.makeText(v.getContext(), text: "Reserva realizada", Toast.LENGTH_LONG).show();
    }
});
aDial.setNegativeButton( text: "Cancelar", new DialogInterface.OnClickListener() {
    ...

```

En caso de ser positivo, se añadirá la reserva a la base de datos, y se le comunicará al dueño de la bicicleta mediante un mensaje.

```

public void addToDatabase(Reserva reserva, String id, Context context) {
    DatabaseReference database = FirebaseDatabase.getInstance("https://biketheride-d83a4-default-rtdb.firebaseio.com").getReference()
        .child("reservas/" + id).setValue(reserva).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                Toast.makeText(context, "Reserva realizada", Toast.LENGTH_LONG).show();
            }
        }).addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
            }
        });
}

```

Se creará el chat con los id de ambos usuarios, el mensaje y una marca de tiempo. Además, se crearán las referencias entre los usuarios, en caso de no existir.

```

private void sendMsg(final String message, String idUserBike) {
    String myId = FirebaseAuth.getInstance().getCurrentUser().getUid();

    DatabaseReference databaseReference = FirebaseDatabase.getInstance("https://biketheride-d83a4-default-rtdb.firebaseio.com");
    String timestamp = String.valueOf(System.currentTimeMillis());
    HashMap<String, Object> hashMap = new HashMap<>();
    hashMap.put("sender", myId);
    hashMap.put("receiver", idUserBike);
    hashMap.put("message", message);
    hashMap.put("timestamp", timestamp);
    databaseReference.child("chats").push().setValue(hashMap);
    final DatabaseReference ref1 = FirebaseDatabase.getInstance("https://biketheride-d83a4-default-rtdb.firebaseio.com").getReference()
        .child("users/" + myId).addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (!dataSnapshot.exists()) {
                    ref1.child("id").setValue(myId);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
            }
        });

    final DatabaseReference ref2 = FirebaseDatabase.getInstance("https://biketheride-d83a4-default-rtdb.firebaseio.com").getReference()
        .child("users/" + idUserBike).addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (!dataSnapshot.exists()) {
                    ref2.child("id").setValue(idUserBike);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
            }
        });
}

```

En el caso de querer eliminar una reserva, se mostrará un mensaje de confirmación, mediante el cual, si se desea continuar con la acción, se borrará de la base de datos con el método `removeValue()` de la referencia a la reserva. Seguidamente se le enviará automáticamente un mensaje al dueño de la bicicleta, comunicándole el rechazo.

```

DatabaseReference mDatabase = FirebaseDatabase.getInstance("https://biketheride-d83a4.firebaseio.com/default-rtdb.firebaseio.com/.json");
AlertDialog.Builder aDial= new AlertDialog.Builder(v.getContext());
aDial.setTitle("Eliminar");
aDial.setMessage("Deseas eliminar la reserva?");
aDial.setPositiveButton( text: "Aceptar", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        mDatabase.child("reservas").child(mItem.getId()).addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                snapshot.getRef().removeValue();
                String msg="Disculpe, pero ya no deseó reservar su bicicleta con descripción "+descriptionBike+", en "+mItem.getCity()+".";

                sendMsg(msg);
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
            }
        });
    }
});
aDial.setNegativeButton( text: "Cancelar", new DialogInterface.OnClickListener() {
}
);

```

Para cargar el mapa con las bicicletas, se implementará la interfaz OnMapReadyCallback. La función de Callback onMapReady que se invoca cuando Google Map's ha cargado el mapa, proporciona una instancia no nula de la clase GoogleMap que se puede usar para actualizar el mapa. Recorriendo la lista de bicicletas, se pueden añadir marcadores con su latitud y longitud.

```

@Override
public void onMapReady(GoogleMap googleMap) {

    mMap = googleMap;

    LatLngBounds.Builder builder = new LatLngBounds.Builder();

    for (Bike c : bicis) {

        LatLng ll = new LatLng(Double.valueOf(c.getLatitude()), Double.valueOf(c.getLongitude()));
        //añade marcador al mapa con la posición de la bici
        mMap.addMarker(new MarkerOptions().position(ll).title(c.getCity()).snippet(String.valueOf(c.getLocation())));
        builder.include(ll);
    }
    LatLngBounds bounds = builder.build();
    CameraUpdate cu = CameraUpdateFactory.newLatLngBounds(bounds, padding: 100);
    mMap.animateCamera(cu);
}

```

- Contacto entre usuarios mediante chat

De igual manera que se envían mensajes automáticos al realizar o eliminar una reserva, también se puede contactar de forma “manual” con otro usuario. Hay varias formas para establecer una conversación con un usuario. Antes de realizar la reserva, desde una reserva ya realizada o desde la opción del menú “Mensajes” si ya hay establecida conversación previa. Todas estas formas llevan a la actividad ChatActivity, la cual mediante varios métodos se encargará de enviar y mostrar los mensajes.

```

private void leerMsgs() {
    // mostrar mensaje después de recuperar datos
    chatList = new ArrayList<>();
    DatabaseReference dbref = FirebaseDatabase.getInstance("https://biketheride-d83a4-default-rtdb.firebaseio.com/app/.json").getReference();
    dbref.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            chatList.clear();
            for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
                ModeloChat modeloChat = dataSnapshot1.getValue(ModeloChat.class);
                if (modeloChat.getSender().equals(myuid) &&
                    modeloChat.getReceiver().equals(uid) ||

                    modeloChat.getReceiver().equals(myuid) &&
                    modeloChat.getSender().equals(uid)) {
                    chatList.add(modeloChat); // añadir chat en chatlist
                }
            }
            adapterChat = new AdapterChat(context: ChatActivity.this, chatList);
            adapterChat.notifyDataSetChanged();
            recyclerView.setAdapter(adapterChat);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
    
```

Con el método leerChats() se obtendrá la lista de usuarios con los que se haya establecido conversación, y con el método ultimoMsg(), el último mensaje de la conversación.

```

private void leerChats() {
    usersList = new ArrayList<>();
    reference = FirebaseDatabase.getInstance("https://biketheride-d83a4-default-rtdb.firebaseio.com/app/.json");
    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            usersList.clear();
            for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
                ModeloUser user = dataSnapshot1.getValue(ModeloUser.class);
                for (ModeloChatList chatList : chatListList) {
                    if (user.getUid() != null && user.getUid().equals(chatList.getId())) {
                        usersList.add(user);
                        break;
                    }
                }
            }
            adapterChatList = new AdapterChatList(getActivity(), usersList);
            recyclerView.setAdapter(adapterChatList);

            // obtengo el ultimo mensaje del usuario
            for (int i = 0; i < usersList.size(); i++) {
                ultimoMsg(usersList.get(i).getUid());
            }
        }
    });
} 
```

```

private void ultimoMsg(final String uid) {
    DatabaseReference ref = FirebaseDatabase.getInstance("https://biketheride-d83a4-default-rtdb.firebaseio.com");
    ref.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            String ultMsg = "";
            for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
                ModeloChat chat = dataSnapshot1.getValue(ModeloChat.class);
                if (chat == null) {
                    continue;
                }
                String sender = chat.getSender();
                String receiver = chat.getReceiver();
                if (sender == null || receiver == null) {
                    continue;
                }

                ultMsg = chat.getMessage();
            }
            adapterChatList.setultimoMsgMap(uid, ultMsg);
            adapterChatList.notifyDataSetChanged();
        }
    });
}

```

6. Testeo y pruebas

6.1. Plan de pruebas

- Pruebas unitarias

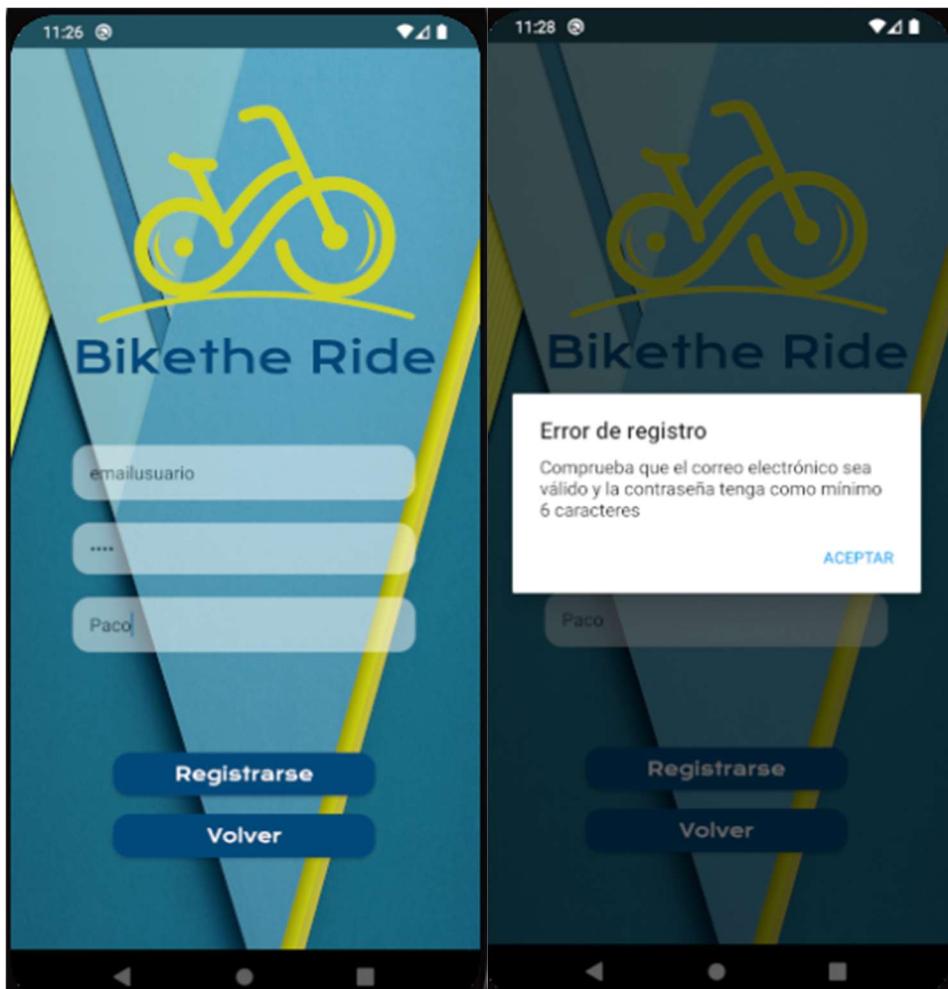
Las pruebas unitarias consisten en verificar el comportamiento de las unidades más pequeñas de su aplicación.

Técnicamente, eso sería una clase o incluso un método de clase en los lenguajes orientados a objetos, y un procedimiento o función en los lenguajes procedimentales y funcionales.

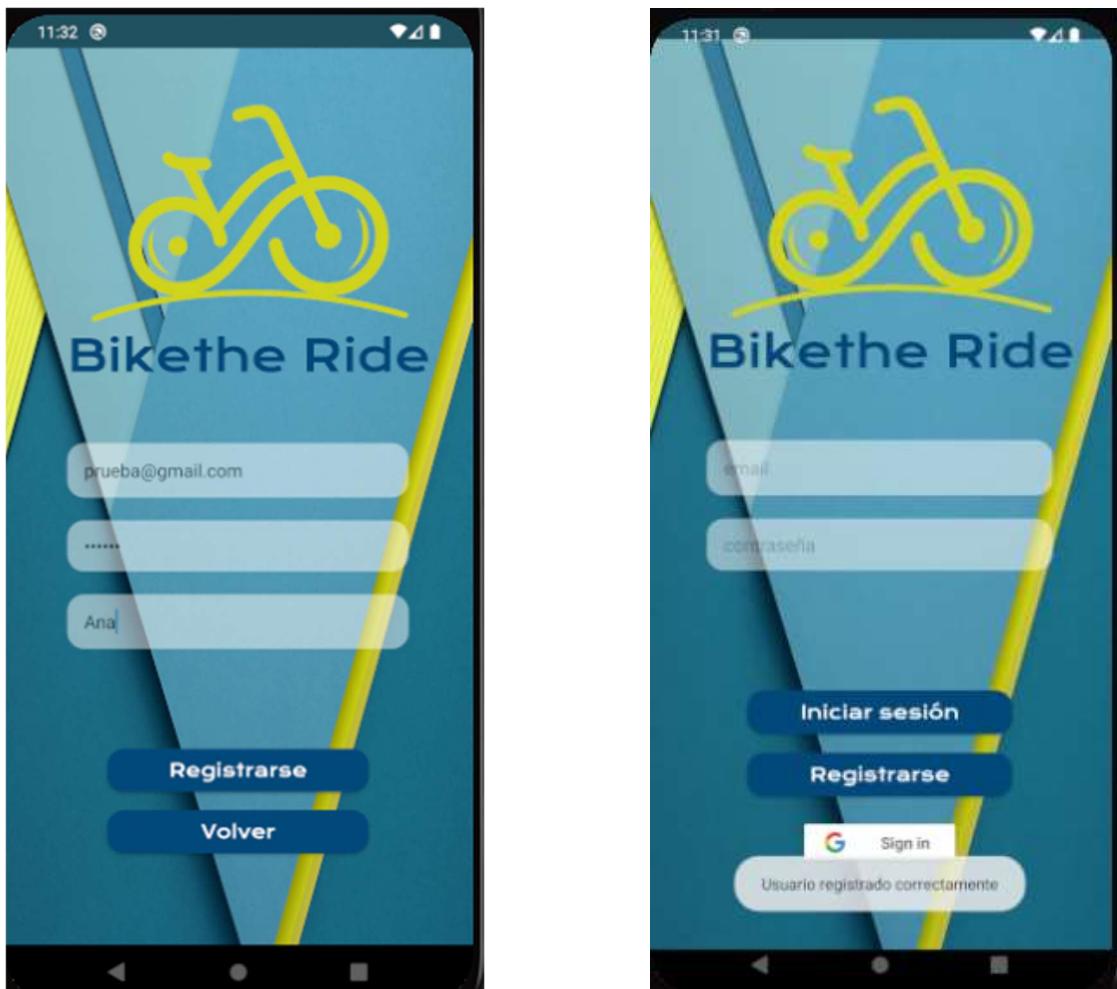
- Prueba 1

Se registrará un usuario en el servicio de autenticación de Firebase y en la base de datos de usuarios de realtime database de Firebase. Se comprobará que los datos de entrada son válidos, y que no se permite el registro de dos usuarios con el mismo email.

En la siguiente imagen se muestra la pantalla de registro y el mensaje de error que se muestra al no completar correctamente los campos.



A continuación, se registrará un usuario correctamente.



Comprobamos la autenticación en Firebase y el registro en la base de datos.

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
prueba@gmail.com	✉	11 dic 2022	11 dic 2022	q090TQAwhiYFwCjZ6OZQXMFnY0...

Realtime Database

[Datos](#) [Reglas](#) [Copias de seguridad](#) [Uso](#)

🔒 Protege tus recursos de Realtime Database contra los abusos, como fraudes de facturación o suplantación de identidad. [Configurar la Verificación de aplicaciones](#) X

```

  https://biketheride-d83a4-default-rtbd.firebaseio.europe-west1.firebaseio.app
  bikes_list
    user
      Ta29znZs34c1fzNBV6d1Nqy3g4r1
      f24y0pHgoKc4UGobCjsadMrZSfg1
      q890TQAwhiYFwCjZ6OZQXMFnY0b2
        email: "prueba@gmail.com"
        name: "Ana"
  
```

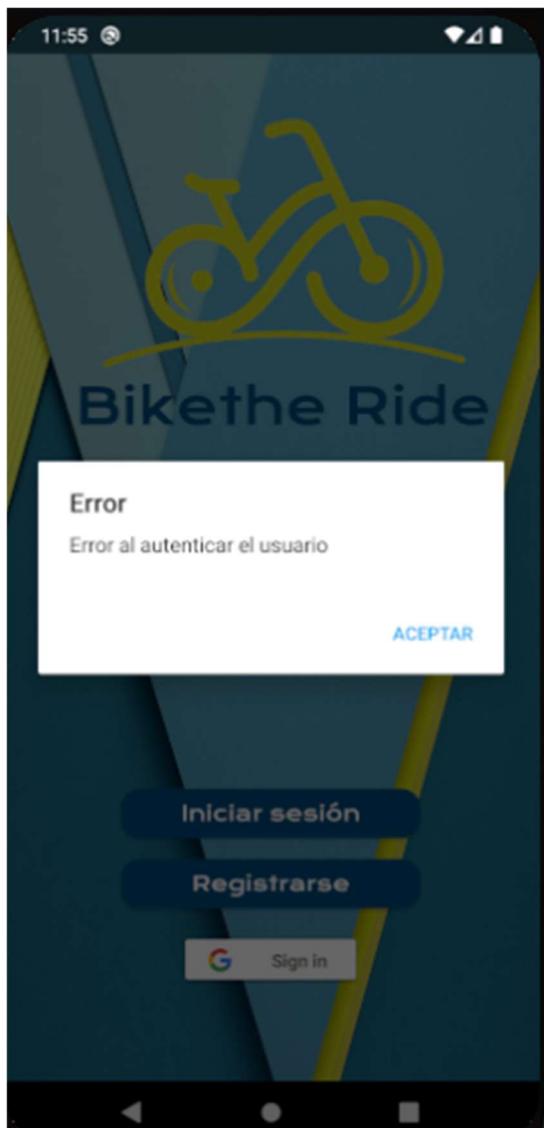
- Prueba 2

Para iniciar sesión se comprobará que los campos de email y contraseña no estén vacíos, que los datos son correctos y que el usuario ya está registrado.

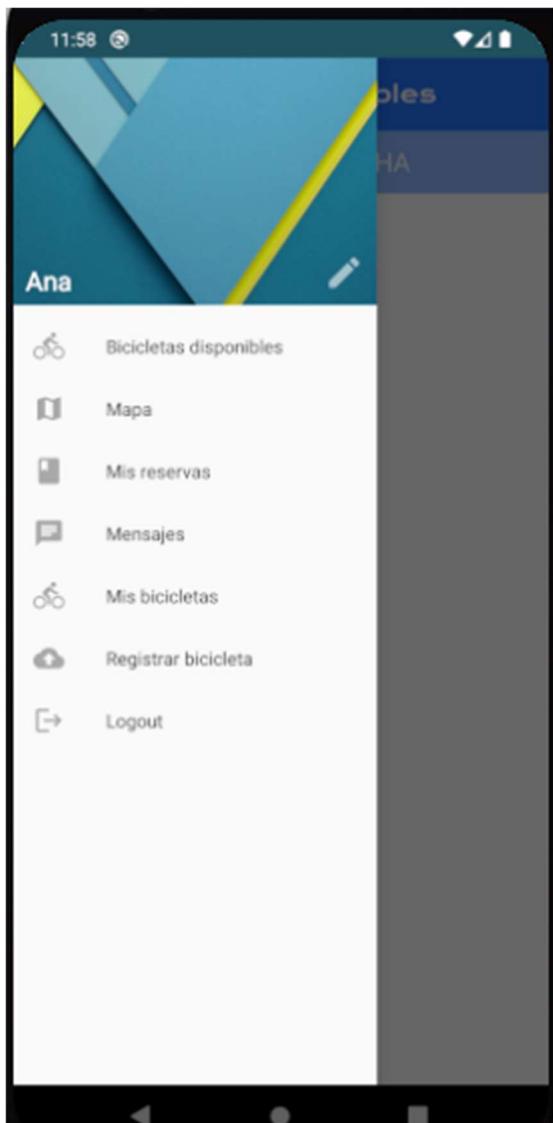
En la siguiente imagen, se muestra como se informa al usuario de que falta algún dato.



Al intentar iniciar sesión con datos incorrectos o con un email no registrado, aparecerá un mensaje de error.

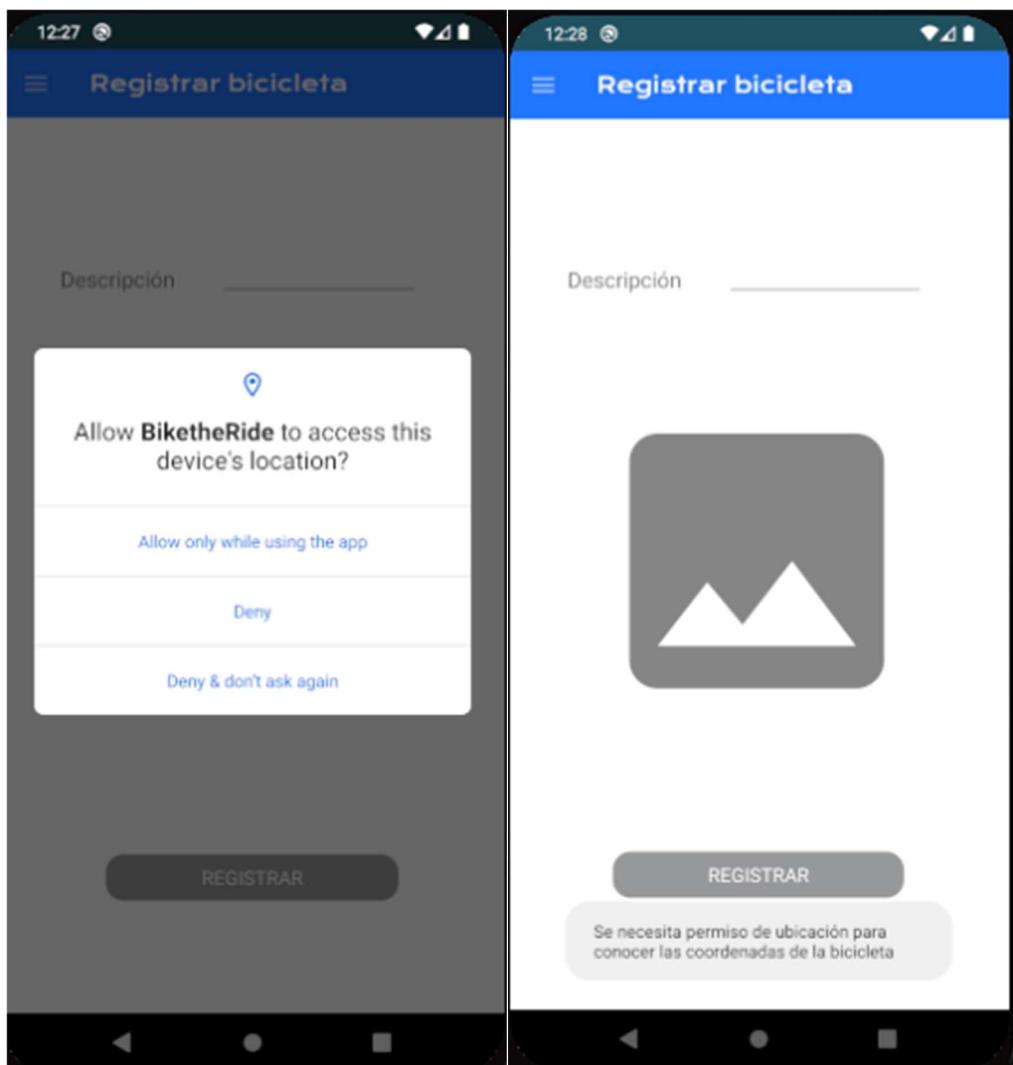


Si el inicio de sesión a sido satisfactorio, se accederá a la aplicación.

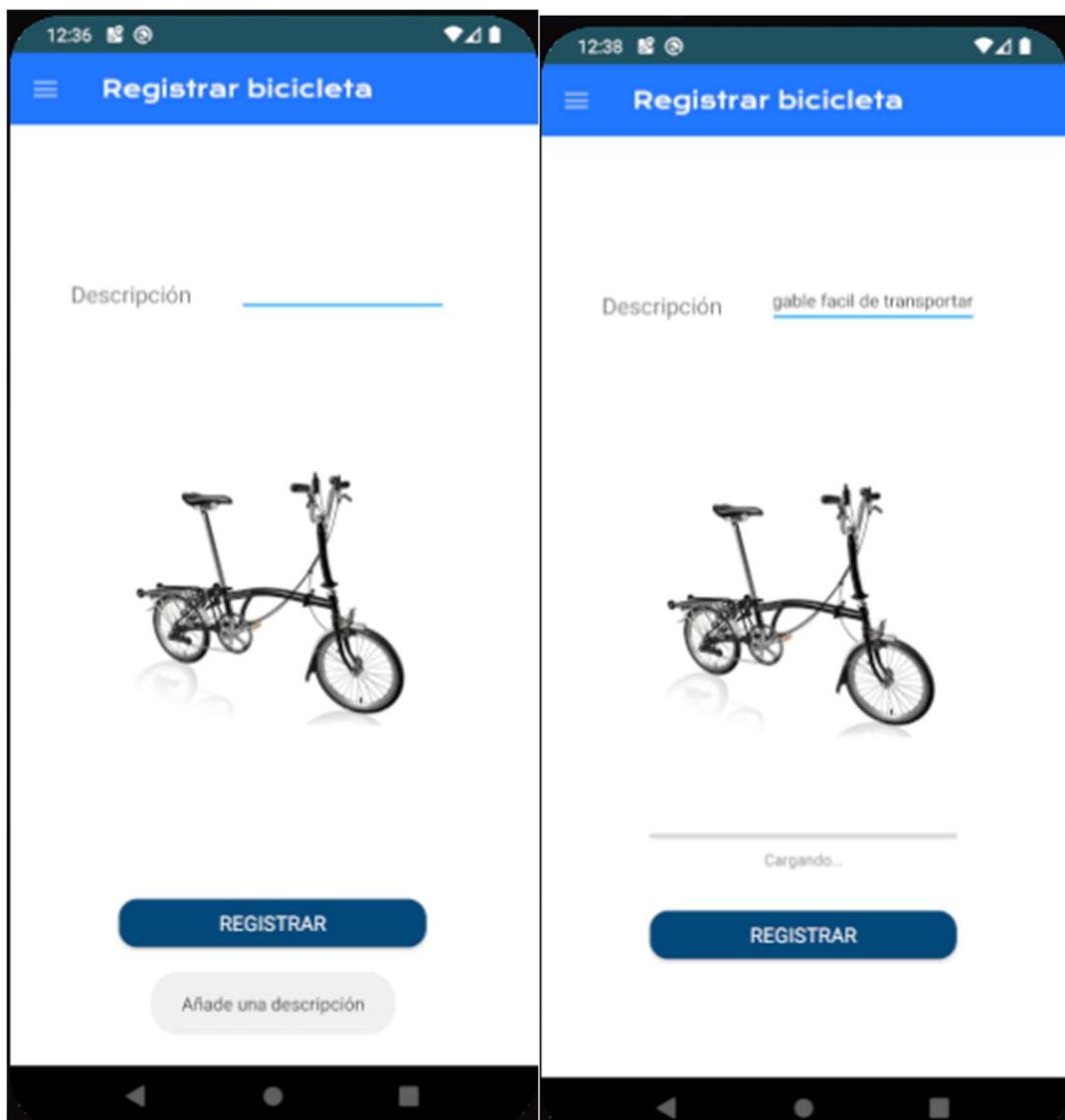


– Prueba 3

Para registrar una bicicleta se necesitarán permisos de ubicación. Si no se disponen de dichos permisos, no se podrá registrar la bicicleta.



Una vez aceptados los permisos, se comprobará que la bicicleta tiene una descripción antes de ser registrada.



Como se puede observar, se ha añadido correctamente a la base de datos y a Storage con el id de la bicicleta como nombre.

A screenshot of the Firebase Realtime Database interface. The URL in the address bar is <https://biketheride-d83a4-default-rtbd.firebaseio-west1.firebaseio.app>.

The database structure under "bikes_list" shows the following data:

- NJ06oN-_woV2z-L5o0v
- NJ0C-kbx3jSGo9QdG59
 - city: "Puertollano"
 - country: "Spain"
 - description: "Bicicleta plegable facil de transportar"
 - id: "-NJ0C-kbx3jSGo9QdG59"

gs://biketheride-d83a4.appspot.com

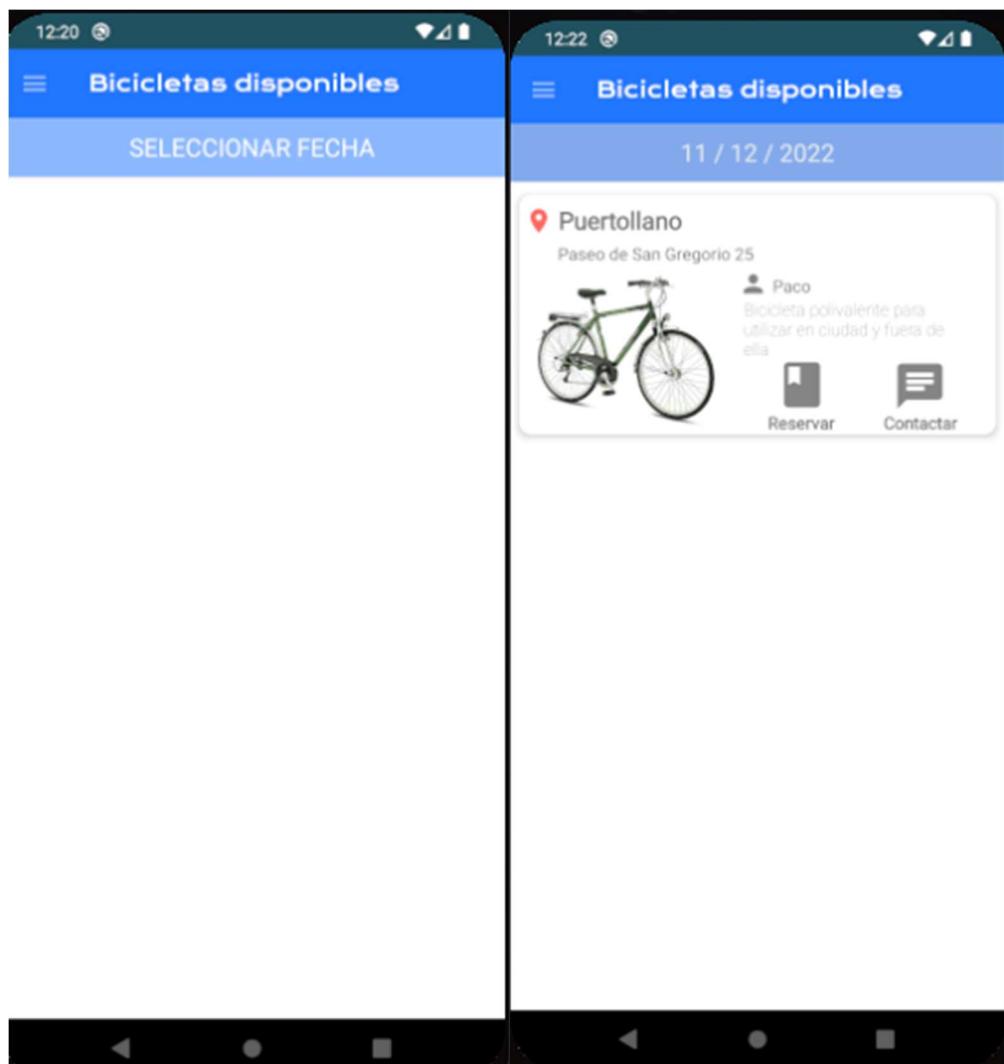
		Tamaño	Tipo	Modificación más reciente	
<input type="checkbox"/>	Nombre				<input type="button" value="Subir archivo"/> <input type="button" value="X"/>
<input type="checkbox"/>	-NJ060N_-woV2z-L5oOv	39.71 KB	image/jpeg	11 dic 2022	<input type="button" value="X"/>
<input type="checkbox"/>	-NJ0C-kbx3jSGo9QdG59	42.46 KB	image/jpeg	11 dic 2022	<input type="button" value="X"/>



Nombre
[-NJ0C-kbx3jSGo9QdG59](#)

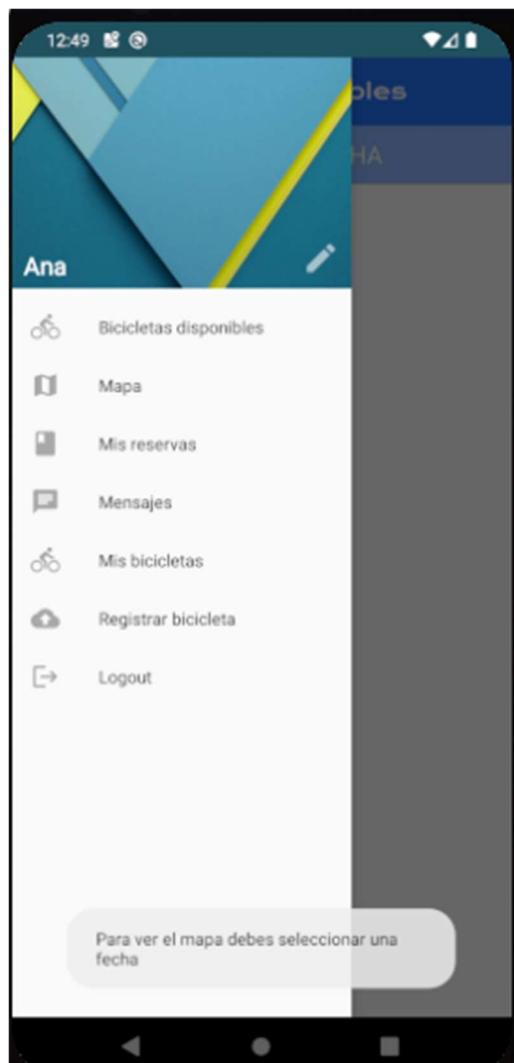
– Prueba 4

Para ver las bicicletas disponibles, se comprobará que se ha seleccionado previamente una fecha. Una vez seleccionada, se mostrarán las bicicletas disponibles.



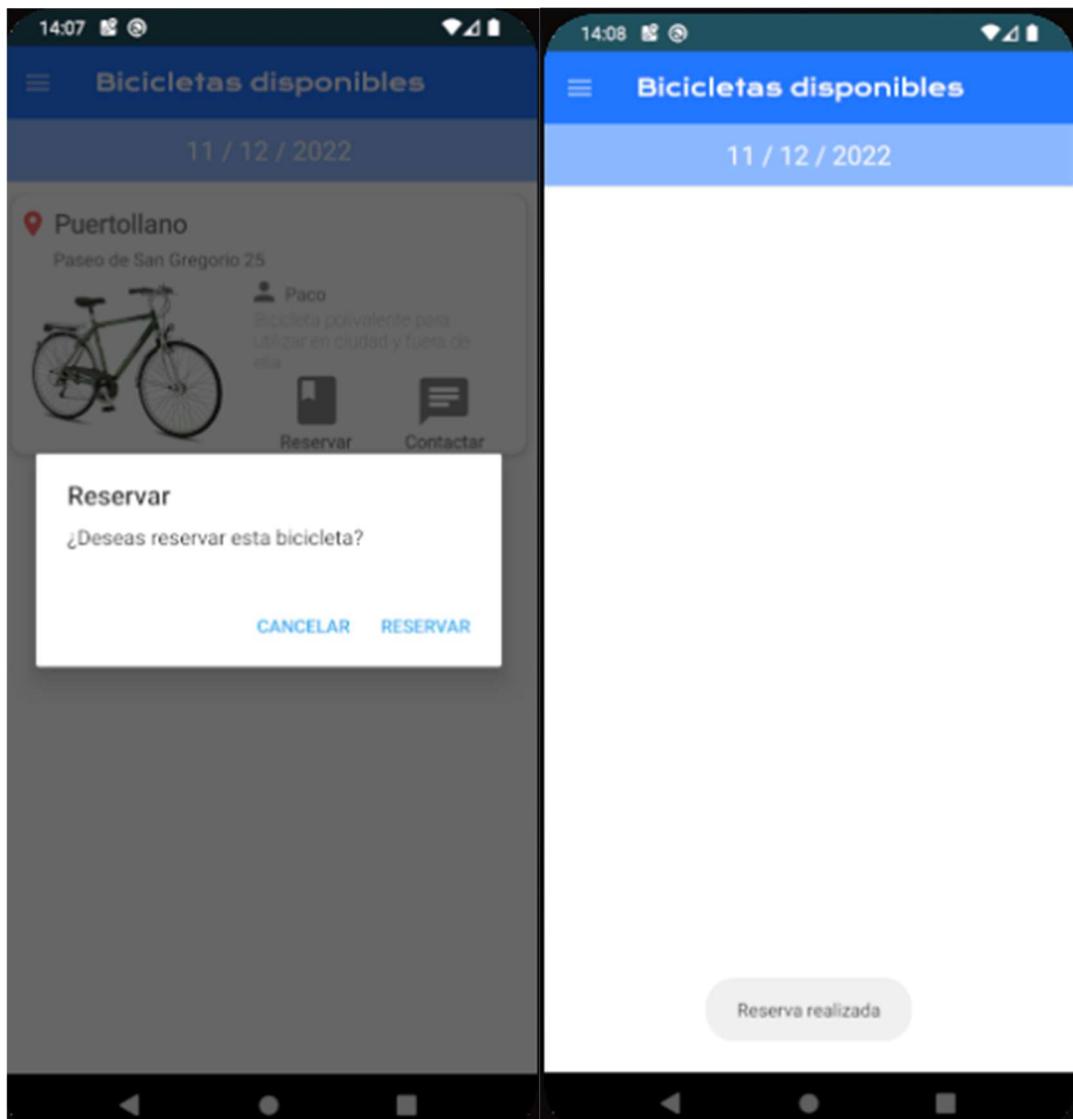
– Prueba 5

El mapa con la ubicación de las bicicletas no estará accesible si no hay una fecha seleccionada.



– Prueba 6

Al realizar la reserva se comprobará que se añade correctamente a la base de datos, y se le envía al dueño un mensaje de reserva. Así mismo, ya no estará disponible para esa fecha determinada.



A screenshot of the Firebase Realtime Database interface. The URL shown is <https://biketheride-d83a4-default-rtdb.europe-west1.firebaseio.app>. The database structure under the "reservas" node is as follows:

- NJ0Vvr7KH_IMq3fs7eN
 - city: "Puertollano"
 - fecha: "11 / 12 / 2022"
 - id: "-NJ0Vvr7KH_IMq3fs7eN"
 - idBike: "-NJo6oN_woV2z-L5o0v"
 - idUser: "q090TQAwHiYFwCjZ60ZQXMFnY0b2"
 - location: "Paseo de San Gregorio 25"

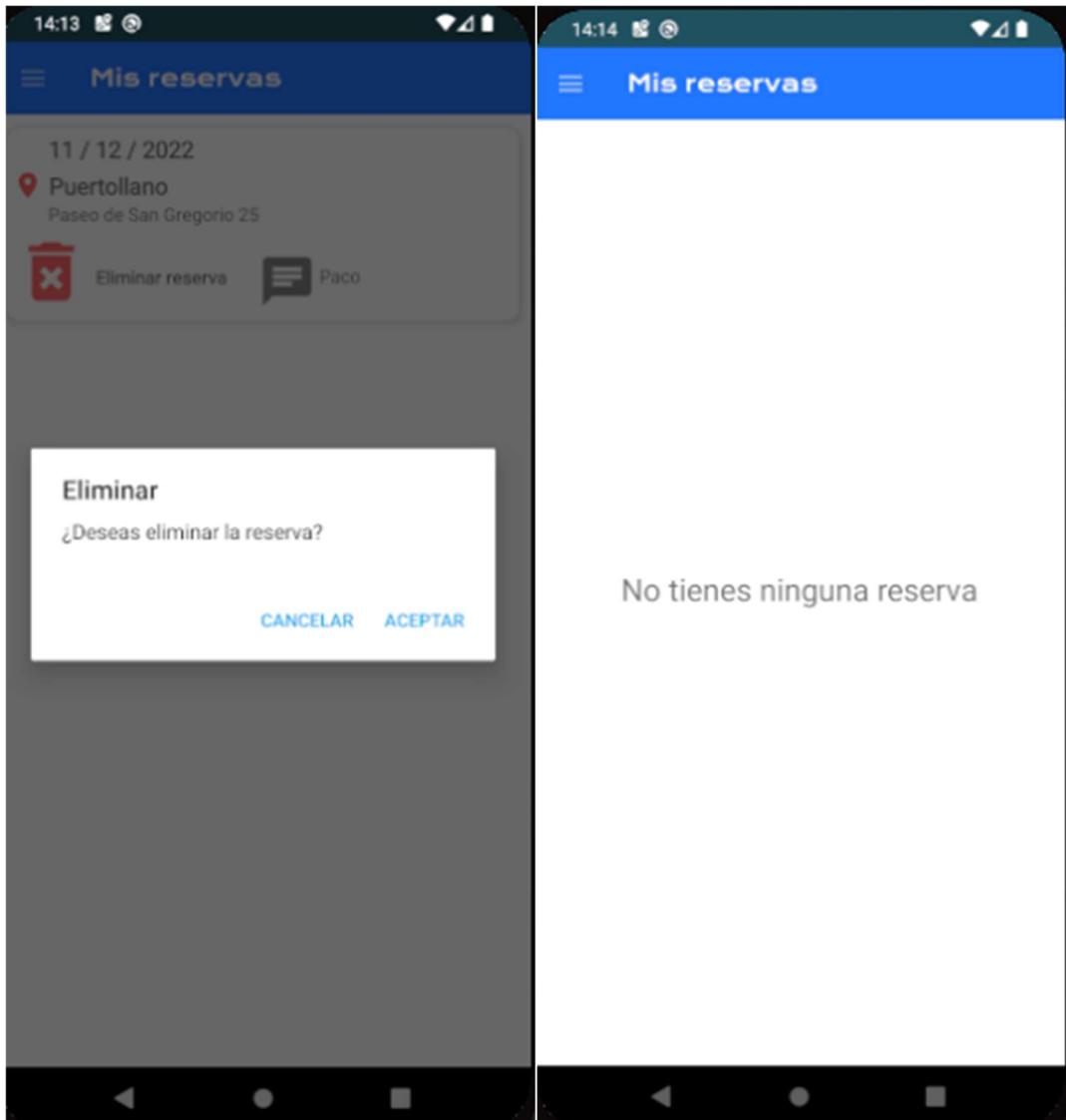
Below this, under the "chats" node, there is one entry:

- NJ0VvrC6U-KTTpziw2K
 - message: "Hola, he reservado su bicicleta con descripción Bicicleta polivalente para utilizar en ciudad y fuera de ella, en Puertollano, Paseo de San Gregorio 25"
 - receiver: "koRBg0goW4OjbVu3HYe6jfrlXyi1"
 - sender: "q090TQAwHiYFwCjZ60ZQXMFnY0b2"
 - timestamp: "1670767688862"

Se ha registrado correctamente la reserva, y se ha enviado el mensaje al dueño.

- Prueba 7

Al eliminar una reserva, se comprobará que se borra de la base de datos y que se le comunica al dueño.



Efectivamente, se ha eliminado la referencia en Realtime Database, y se le ha enviado un mensaje al dueño con la dicha decisión.

A screenshot of the Firebase Realtime Database console. The URL in the address bar is "https://biketheride-d83a4-default-rtbd.firebaseio-west1.firebaseio.database.app". The database structure is visible on the left, showing the following hierarchy:
- root
 - bikes_list
 - chatList
 - chats
 - user
A scroll bar is visible on the right side of the interface.

```

    sender: "q090TQAwhiYFwCjZ6OZQXMFnY0b2"
    timestamp: "167076768862"
    -NJ0XNLdP-LgXf7e7HPT + 🗑
    message: "Disculpe, pero ya no deseo reservar su bicicleta con descripción Bicicleta polivalente para utilizar en ciudad y fuera de ella, en Puertollano, Paseo de"
    receiver: "koRBgOgoW40jbVu3HYe6jfrlXyi1"
    sender: "q090TQAwhiYFwCjZ6OZQXMFnY0b2"
    timestamp: "1670768067707"

```

– Prueba 8

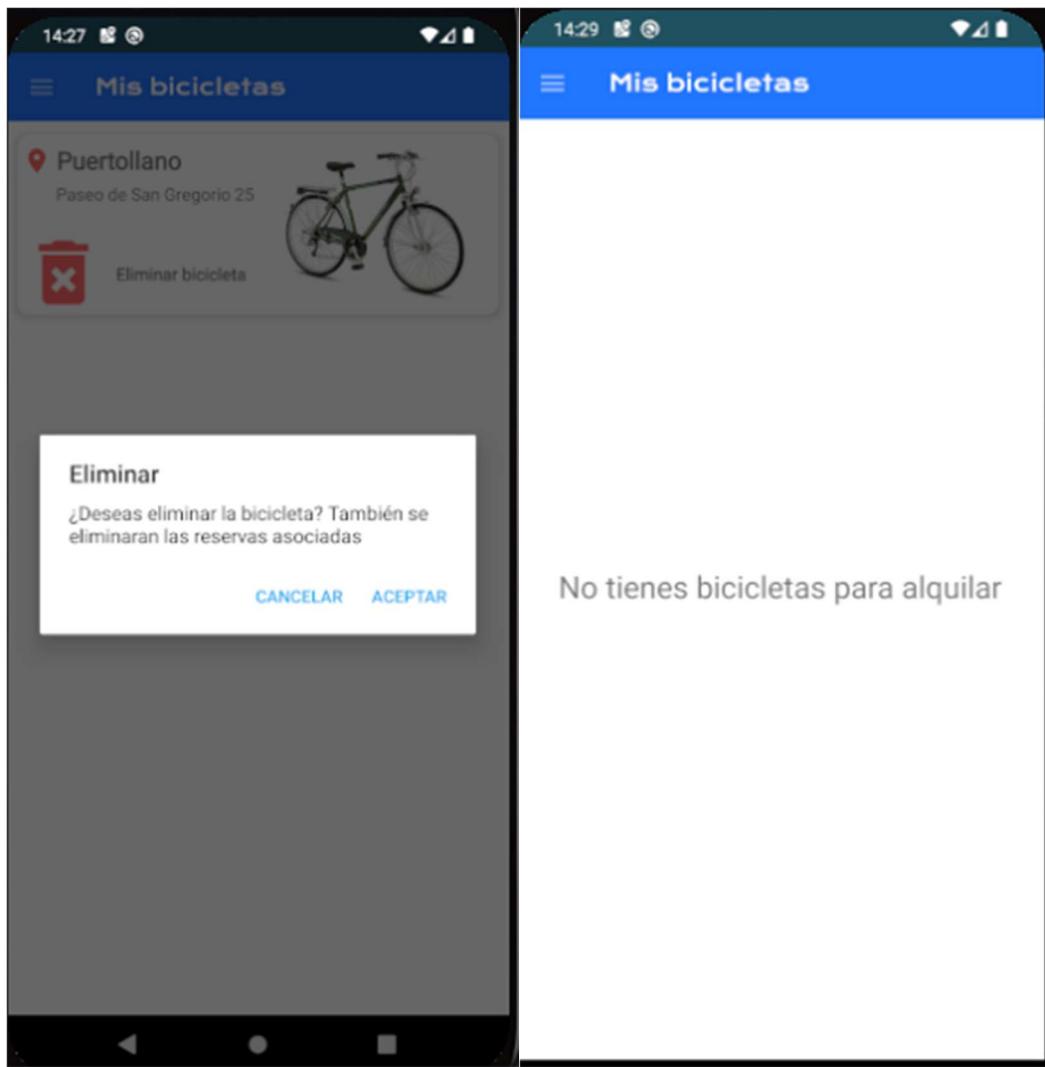
Para eliminar una bicicleta de la base de datos, primero se comprobarán las reservas que tiene, para así eliminarlas, y enviar un mensaje al usuario comunicándole que ya no está disponible la bicicleta. Seguidamente se eliminará de la base de datos la bicicleta y la imagen asociada de Storage.

reservas	
-NJ0Z8MvsjYYNP9xcqHd	city: "Puertollano" fecha: "11 / 12 / 2022" id: "-NJ0Z8MvsjYYNP9xcqHd" idBike: "-NJ06oN_-woV2z-L5o0v" 🗑 idUser: "q090TQAwhiYFwCjZ6OZQXMFnY0b2" location: "Paseo de San Gregorio 25"

bikes_list	
-NJ06oN_-woV2z-L5o0v	city: "Puertollano" country: "Spain" description: "Bicicleta polivalente para utilizar en ciudad y fuera de" id: "-NJ06oN_-woV2z-L5o0v" idUser: "koRBgOgoW40jbVu3HYe6jfrlXyi1"

Nombre	Tamaño	Tipo	Modificación más reciente
-NJ06oN_-woV2z-L5o0v	39.71 KB	image/jpeg	11 dic 2022
-NJOc-kbx3jSGe9QdG59	42.46 KB	image/jpeg	11 dic 2022

Nombre
-NJ06oN_-woV2z-L5o0v



Se puede observar que se ha eliminado correctamente de la base de datos la reserva y la bicicleta, y la imagen de Storage.

https://biketheride-d83a4-default-rtbd.firebaseio-west1.firebaseio.app/

```

https://biketheride-d83a4-default-rtbd.firebaseio-west1.firebaseio.app/
  +-- bikes_list
    +-- -NJ0C-kbx3jSGo9QdG59
  +-- chatList
  +-- chats
  +-- user
  
```

gs://biketheride-d83a4.appspot.com

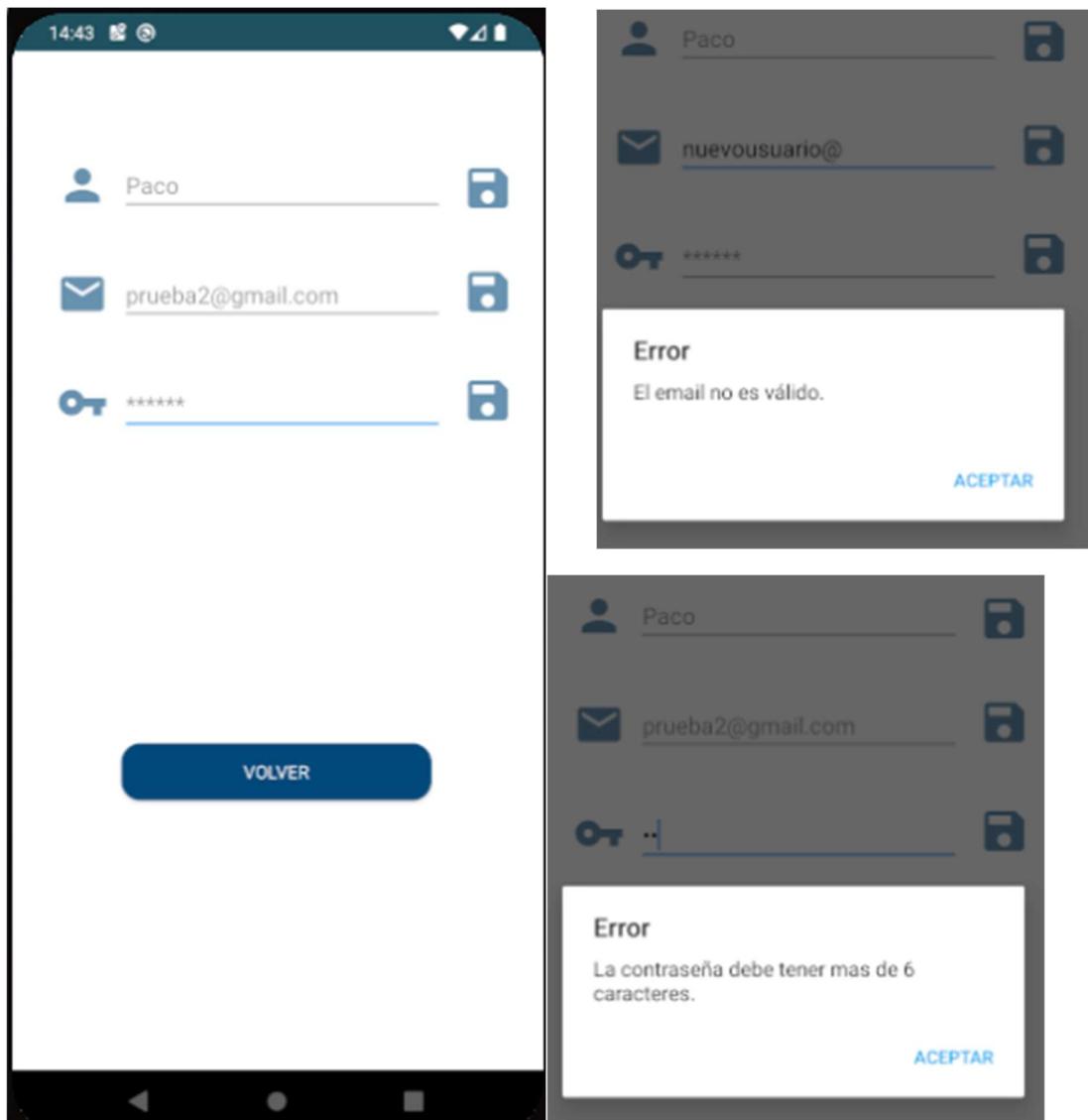
	Nombre	Tamaño	Tipo	Modificación más reciente
<input type="checkbox"/>	-NJ0C-kbx3jSGo9QdG59	42.46 KB	image/jpeg	11 dic 2022

Además, se le ha enviado un mensaje al usuario que reservó la bicicleta.

```
-NJ0_q-mHEv7DtR1a66E
  message: "Disculpe per la bicicleta con descripción Bicicleta polivalente para utilizar en ciudad y fuera de ella, en Puerto Llanillo, Paseo de San Gregorio 25ya no esta en alquiler"
  receiver: "q090TQAwHiYFwCjZ6OZQXMFnY0b2"
  sender: "korBgOgoW4OjbVu3HYe6jfrlXyi1"
  timestamp: "1670768975619"
```

– Prueba 9

Para cambiar los datos de perfil de usuario, se comprobará que la nueva dirección de correo electrónico es válida, y que la contraseña es mayor de 6 caracteres.



En caso de actualizar correctamente, se puede observar que se han modificado en Firebase Authentication y en Realtime Database.



VOLVER

Antes de editar:

Buscar por dirección de correo electrónico, número de teléfono o UID de usuario					Agregar usuario	⋮
Identificador	Proveedores	Fecha de creación	↓	Fecha de acceso	UID de usuario	⋮
prueba2@gmail.com	✉	11 dic 2022		11 dic 2022	koRBgOgoW4OjbVu3HYe6jfrlXyi1	⋮
prueba@gmail.com	✉	11 dic 2022		11 dic 2022	q090TQAwhiYFwCjZ6OZQXMFnY0...	⋮

Después de editar:

Buscar por dirección de correo electrónico, número de teléfono o UID de usuario					Agregar usuario	⋮
Identificador	Proveedores	Fecha de creación	↓	Fecha de acceso	UID de usuario	⋮
nuevousuario@gmail.com	✉	11 dic 2022		11 dic 2022	koRBgOgoW4OjbVu3HYe6jfrlXyi1	⋮
prueba@gmail.com	✉	11 dic 2022		11 dic 2022	q090TQAwhiYFwCjZ6OZQXMFnY0...	⋮



- Pruebas del sistema

Las pruebas del sistema tienen como objetivo ejercitar profundamente el sistema comprobando la integración del sistema globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica.

- Prueba 10

Como sea ya se ha comprobado anteriormente, no se le permite iniciar sesión a un usuario no autenticado.

- Prueba 11

Se ha comprobado con un usuario inexperto que la navegación por la aplicación es intuitiva, con visualizaciones claras y concisas de lo que se hace en cada sección.

7. Repositorio

Se encontrará el código fuente de la aplicación y toda la documentación en el repositorio de GitHub:

<https://github.com/Rruic/BiketheRide.git>

8. Bibliografía

https://es.wikipedia.org/wiki/Requisito_no_funcional

<https://www.juntadeandalucia.es/servicios/madeja/contenido/procedimiento/20>

<https://webescuela.com/estudio-de-mercado/>

<https://dafo.ipyme.org/Home>

<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/>

<https://www.programoergosum.es/tutoriales/flujo-de-trabajo-en-git-con-gitflow/>

<https://dev.to/gbumanzordev/flujo-de-trabajo-con-git-flow-fe>

<https://aulacm.com/precio-desarrollar-app-aplicacion-movil/>

<https://www.unir.net/ingenieria/revista/riesgos-laborales-informatica/>

<https://ingenieriasoft.webcindario.com/gestion-y-planificacion-de-proyectos/planificacion-de-proyectos-de-software/riesgo-del-software.html>

<https://www.autonomosyemprendedor.es/articulo/ayudas-y-subvenciones/autonomos-castilla-mancha-pueden-solicitar-ayudas-fondo-perdido-4200-euros/20220111133257025833.html>

https://es.wikipedia.org/wiki/Backend_as_a_service

<https://platzi.com/blog/serverless-cloud-functions/>

<https://es.wikipedia.org/wiki/Firebase>

<https://jelvix.com/blog/backend-as-a-service>

<https://profile.es/blog/que-es-ionic/>

https://es.wikipedia.org/wiki/React_Native

<https://inmediatum.com/blog/ingenieria/ventajas-y-desventajas-de-apps-desarrolladas-en-xamarin/>

<https://abamobile.com/web/desarrollo-aplicaciones-flutter-caracteristicas-ventajas/>

[https://es.wikipedia.org/wiki/Kotlin_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n))

<https://keepcoding.io/blog/ventajas-y-desventajas-de-kotlin/>