universidad de león

# Escuela de Ingenierías

# Industrial, Informática y Aeroespacial

## GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

Aplicación móvil para la gestión y reserva de taxis compartido.

Mobile application for the management and reservation of shared taxis.

Autor: Diego Fernández Velasco

Tutor: José Alberto Benítez Andrades

(Julio, 2022)

UNIVERSIDAD DE LEÓN

Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

**Alumno:** Diego Fernández Velasco

**Tutor:** José Alberto Benítez Andrades

**Título:** Aplicación móvil para la gestión y reserva de taxis compartido

**Title:** Mobile application for the management and reservation of shared taxis

**Convocatoria:** Julio, 2022

**Resumen:** Durante los últimos años, el uso del taxi en nuestro país ha decrecido provocado por diversos motivos, entre los que se encuentra el elevado precio que supone hacer uso de este servicio. Por ello, sobre todo en grandes ciudades donde determinados trayectos se repiten de forma constante por diferentes personas al mismo tiempo, se hace importante y necesario la existencia de un servicio que permita compartir taxi entre distintos viajeros de una forma rápida, permitiendo a cada uno de estos, abaratar el coste del trayecto.

Lo que se propone desde la aplicación "BlablaTaxi", es permitir a los usuarios compartir dichos viajes, consiguiendo así, rebajar el coste del mismo. Los trayectos los prefijarán las compañías de taxis. Las diferentes personas buscarán el recorrido que más se adecue a sus necesidades, se apuntarán a él, y en función del número de usuarios que finalmente se hayan anotado al viaje, se procederá a cobrar una cantidad u otra al cliente correspondiente. La aplicación estará disponible tanto en Google Play como en App Store, y su diseño será completamente igual en ambas plataformas.

**Abstract:** In recent years, the use of taxis in our country has decreased due to various reasons, among which is the high price of using this service. For this reason, especially in large cities where certain routes are constantly repeated by different people at the same time, it is important and necessary to have a service that allows taxis to be shared

quickly between different travelers, allowing each of them to these, reduce the cost of the journey.

What is proposed from the "BlablaTaxi" application is to allow users to share these trips, thus reducing the cost of it. The routes will be prefixed by the taxi companies. The different people will look for the route that best suits their needs, they will sign up for it, and depending on the number of users who have finally signed up for the trip, one amount or another will be charged to the corresponding client. The application will be available on both Google Play and the App Store, and its design will be completely the same on both platforms.

**Palabras clave:** Flutter, Dart, FrontEnd, BackEnd, NodeJS, MongoDB, taxi, aplicación móvil, aplicación híbrida.

# Índice de contenidos

Índice de figuras ..... 8

Índice de tablas ..... 10

Glosario ..... 14

1. Introducción ..... 2

1.1. Planteamiento del problema ..... 2

1.2. Objetivos ..... 2

1.3. Metodología ..... 3

1.4. Tecnologías y herramientas ..... 5

1.5. Estructura del trabajo ..... 6

2. Estudio del problema ..... 8

2.1. El contexto del problema ..... 8

2.2. El estado del arte ..... 8

2.2.1. Aplicaciones similares ..... 8

2.2.2. Opinión sobre el estado del arte ..... 10

2.3. La definición del problema ..... 10

3. Gestión del proyecto software ..... 12

3.1. Alcance del proyecto ..... 12

3.1.1. Definición del proyecto ..... 12

3.1.2. Objetivos del proyecto ..... 13

3.1.3. Límites del proyecto ..... 13

3.1.4. Productos entregables ..... 14

3.1.5. Criterios de aceptación ..... 14

3.1.6. Restricciones ..... 14

3.2. Plan de trabajo ..... 15

3.2.1. Identificación de fases ..... 15
3.2.2. Identificación de tareas ..... 15
3.2.3. Estimación y planificación de tareas ..... 17
3.3. Gestión de recursos ..... 24
3.3.1. Especificación de recursos ..... 24
3.4. Presupuesto ..... 25
3.4.1. Costes de Hardware ..... 25
3.4.2. Costes de Software ..... 25
3.4.3. Costes de recursos humanos ..... 25
3.4.4. Costes totales ..... 26
3.5. Gestión de riesgos ..... 26
3.5.1. Identificación de riesgos ..... 26
3.5.2. Análisis de riesgos ..... 27
3.5.3. Estimación de probabilidad e impacto ..... 30
3.5.4. Exposición al riesgo ..... 31
4. Definición de la solución ..... 33
4.1. Descripción de la solución ..... 33
4.1.1. Descomposición en subsistemas ..... 33
4.2. Análisis del sistema ..... 35
4.2.1. Especificación de requisitos ..... 35
4.2.2. Diagrama de casos de uso ..... 44
4.2.3. Especificación de casos de uso ..... 46
4.3. Diseño del sistema ..... 54
4.3.1. Arquitectura de software ..... 54
4.3.2. Arquitectura del sistema ..... 55
4.3.3. Diseño de la base de datos ..... 56

4.3.4. Diseño del logo ..... 60
4.4. Implementación ..... 62
4.4.1. Tecnologías empleadas ..... 62
4.4.2. Limitaciones existentes ..... 62
4.4.3. Estructura general de la aplicación ..... 63
4.4.4. Base de datos ..... 65
4.4.5. Implementación del FrontEnd ..... 66
4.4.6. Implementación del BackEnd ..... 66
4.4.7. Publicación en las tiendas. ..... 68
4.4.8. Posibles mejoras ..... 69
4.5. Pruebas ..... 69
4.5.1. Pruebas unitarias ..... 70
4.5.1.1. Pruebas de caja blanca ..... 70
4.5.1.2. Pruebas de caja negra ..... 71
5. Normativa vigente ..... 74
5.1. Ley de protección de datos ..... 74
5.2. Ley de Propiedad Intelectual ..... 75
6. Producto final ..... 76
6.1. Inicio de sesión ..... 76
6.2. Registrarse ..... 77
6.3. Cerrar sesión ..... 77
6.4. Buscar viajes ..... 78
6.5. Consultar mis viajes ..... 81
6.6. Perfil de usuario ..... 82
7. Conclusiones ..... 84
7.1. Aportaciones realizadas ..... 84

7.2. Trabajos futuros ..... 84
7.3. Problemas encontrados ..... 85
7.4. Opinión personal ..... 85
Agradecimientos ..... 86
Bibliografía ..... 87
Anexos ..... 90
Anexo I: Controlador de versiones ..... 90
Anexo II: Manual de usuario ..... 93

# Índice de figuras

Figura 1.1: Desarrollo iterativo e incremental ..... 4

Figura 3.1: Diagrama de Gantt de las fases del proyecto ..... 17

Figura 3.2: Diagrama de Gantt del plan inicial ..... 18

Figura 3.3: Diagrama de Gantt del proceso iterativo ..... 19

Figura 3.4: Diagrama de Gantt de la iteración 1 ..... 20

Figura 3.5: Diagrama de Gantt de la iteración 2 ..... 20

Figura 3.6: Diagrama de Gantt de la iteración 3 ..... 21

Figura 3.7: Diagrama de Gantt de la iteración 4 ..... 22

Figura 3.8: Diagrama de Gantt de la iteración 5 ..... 22

Figura 3.9: Diagrama de Gantt de la iteración 6 ..... 23

Figura 3.10: Planificación de la revisión y puesta en marcha ..... 24

Figura 4.1: Subsistemas de la aplicación ..... 34

Figura 4.2: Diagrama de casos de uso del usuario ..... 45

Figura 4.3: Diagrama de casos de uso de la aplicación o sistema ..... 46

Figura 4.4: Diagrama de casos de uso de los conductores ..... 46

Figura 4.5: Patrón MVVM ..... 55

Figura 4.6: Esquema de la base de datos ..... 58

Figura 4.7: Logo de BlablaTaxi ..... 61

Figura 4.8: Letrero de indicación del estado del taxi ..... 61

Figura 4.9: Estructura general del proyecto ..... 63

Figura 4.10: Estructura carpeta lib ..... 64

Figura 4.11: Estructura de app y subdirectorio core ..... 64

Figura 4.12: Estructura del directorio ui ..... 65

Figura 4.13: Ejemplo de API - mostrar usuarios ..... 65

Figura 4.14: Ejemplo de petición para obtener un usuario de la API ..... 66

Figura 4.15: Conexión del servidor con la base de datos ..... 67
Figura 4.16: Estructura del servidor NodeJS ..... 68
Figura 5.1: Cambios de la RPGD ..... 74
Figura 6.1: Pantalla de inicio de sesión ..... 76
Figura 6.2: Pantalla de registro de usuarios ..... 77
Figura 6.3: Pantalla de búsqueda de viajes ..... 78
Figura 6.4: Ejemplo de error en la búsqueda de viajes ..... 79
Figura 6.5: Ejemplo de búsqueda de viajes ..... 80
Figura 6.6: Ejemplo de viajes no encontrados ..... 81
Figura 6.7: Pantalla de mis viajes ..... 82
Figura 6.8: Pantalla de perfil ..... 83
Figura 0.1: Logo de Git ..... 90
Figura 0.2: Logo de GitHub ..... 90
Figura 0.3: Número de commits del repositorio BackEnd ..... 91
Figura 0.4: Número de commits del repositorio FrontEnd ..... 91
Figura 0.5: Ejemplo de las ramas usadas ..... 92

# Índice de tablas

Tabla 3.1: Planificación de fases del proyecto ..... 17

Tabla 3.2: Planificación del plan inicial ..... 18

Tabla 3.3: Planificación del proceso iterativo ..... 18

Tabla 3.4: Planificación de la iteración 1 ..... 19

Tabla 3.5: Planificación de la iteración 2 ..... 20

Tabla 3.6: Planificación de la iteración 3 ..... 21

Tabla 3.7: Planificación de la iteración 4 ..... 21

Tabla 3.8: Planificación de la iteración 5 ..... 22

Tabla 3.9: Planificación de la iteración 6 ..... 23

Tabla 3.10: Planificación de la revisión y puesta en marcha..... 23

Tabla 3.11: Costes de recursos humanos..... 24

Tabla 3.12: Costes de Hardware..... 25

Tabla 3.13: Costes de Software ..... 25

Tabla 3.14: Costes de recursos humanos..... 26

Tabla 3.15: Costes totales..... 26

Tabla 3.16: Identificación de riesgos..... 27

Tabla 3.17: RI-01 Errores de estimación para el tiempo de desarrollo ..... 28

Tabla 3.18: RI-02 Errores en la estimación del presupuesto..... 28

Tabla 3.19: RI-03 Errores de estimación en el personal necesario ..... 28

Tabla 3.20: RI-04 Baja experiencia y conocimiento de los trabajadores ..... 29

Tabla 3.21: RI-05 Fallos de seguridad..... 29

Tabla 3.22: RI-06 Modificación de los requisitos ..... 29

Tabla 3.23: RI-07 Adaptación a cambios del mercado ..... 30

Tabla 3.24: Tabla de probabilidades de aparición de los riesgos ..... 30

Tabla 3.25: Tabla de definición del impacto del riesgo ..... 30
Tabla 3.26: Asociación de probabilidad e impacto por riesgo ..... 31
Tabla 3.27: Tabla con la exposición al riesgo ..... 32
Tabla 4.1: Tipos de prioridades según la metodología MoSCoW..... 35
Tabla 4.2: RF01 (Inicio de sesión por medio de email y contraseña) ..... 36
Tabla 4.3: RF02 (Inicio de sesión por medio de cuenta Google)..... 36
Tabla 4.4: RF03 (Mantener sesión iniciada) ..... 36
Tabla 4.5: RF04 (Cerrar sesión de usuario)..... 36
Tabla 4.6: RF05 (Registro de usuarios)..... 37
Tabla 4.7: RF06 (Registro de usuarios por medio de cuenta Google) ..... 37
Tabla 4.8: RF07 (Modificación de perfiles de usuario) ..... 37
Tabla 4.9: RF08 (Eliminación de perfiles de usuario)..... 37
Tabla 4.10: RF09 (Control de errores en registro) ..... 38
Tabla 4.11: RF10 (Control de contraseña en el registro) ..... 38
Tabla 4.12: RF11 (Control de errores en el inicio de sesión)..... 38
Tabla 4.13: RF12 (Creación de trayectos)..... 38
Tabla 4.14: RF13 (Reserva de trayectos) ..... 39
Tabla 4.15: RF14 (Cancelación de trayectos) ..... 39
Tabla 4.16: RF15 (Confirmación de trayectos)..... 39
Tabla 4.17: RF16 (Estado de trayectos) ..... 39
Tabla 4.18: RF17 (Consulta de trayectos realizados)..... 40
Tabla 4.19: RF18 (Notificación de confirmación de trayectos)..... 40
Tabla 4.20: RF19 (Notificación de cancelación de trayectos)..... 40
Tabla 4.21: RF20 (Creación de usuarios con el rol de conductor)..... 40
Tabla 4.22: RF21 (Creación de trayectos)..... 41
Tabla 4.23: RF22 (Confirmación de trayectos)..... 41

Tabla 4.24: RF23 (Cancelación de trayectos) ..... 41
Tabla 4.25: RF24 (Modificación de perfiles de conductor) ..... 41
Tabla 4.26: RF25 (Eliminación de perfiles de conductor) ..... 42
Tabla 4.27: FNR01 (Aplicación móvil adaptable) ..... 42
Tabla 4.28: RNF02 (Aplicación móvil accesible) ..... 42
Tabla 4.29: RNF03 (Diseño intuitivo y simple) ..... 42
Tabla 4.30: RNF04 (Tiempo de respuesta) ..... 43
Tabla 4.31: RNF05 (Concurrencia de usuarios) ..... 43
Tabla 4.32: RNF06 (Privacidad de sesión) ..... 43
Tabla 4.33: RNF07 (Privacidad del usuario) ..... 43
Tabla 4.34: RNF08 (Acceso a la aplicación) ..... 44
Tabla 4.35: RNF09 (Privacidad de los conductores) ..... 44
Tabla 4.36: RNF10 (Compatibilidad con Android y iOS) ..... 44
Tabla 4.37: RNF11 (Ley de protección de datos) ..... 44
Tabla 4.38: CU01 (Iniciar sesión) ..... 47
Tabla 4.39: CU02 (Inicio de sesión con Google) ..... 47
Tabla 4.40: CU03 (Cerrar sesión) ..... 48
Tabla 4.41: CU04 (Registrarse) ..... 48
Tabla 4.42: CU05 (Registrarse con Google) ..... 48
Tabla 4.43: CU06 (Modificar perfil) ..... 49
Tabla 4.44: CU07 (Eliminar perfil) ..... 49
Tabla 4.45: CU08 (Reservar trayectos) ..... 49
Tabla 4.46: CU09 (Cancelar trayecto) ..... 50
Tabla 4.47: CU10 (Consultar trayectos) ..... 50
Tabla 4.48: CU12 (Crear trayectos) ..... 50
Tabla 4.49: CU13 (Confirmar trayectos) ..... 50

Tabla 4.50: CU14 (Cancelar trayectos) ..... 51
Tabla 4.51: CU15 (Notificación de confirmación)..... 51
Tabla 4.52: CU16 (Notificación de cancelación) ..... 51
Tabla 4.53: CU17 (Control de errores de registro)..... 51
Tabla 4.54: CU18 (Control de errores de inicio de sesión) ..... 52
Tabla 4.55: CU19 (Creación de conductores) ..... 52
Tabla 4.56: CU20 (Crear trayectos) ..... 52
Tabla 4.57: CU21 (Confirmar trayectos)..... 53
Tabla 4.58: CU22 (Cancelar trayectos) ..... 53
Tabla 4.59: CU23 (Modificar perfil) ..... 53
Tabla 4.60: CU24 (Eliminar perfil) ..... 54
Tabla 4.61: Colección de usuarios..... 59
Tabla 4.62: Colección de viajes..... 59
Tabla 4.63: Colección de conductores..... 60

## Glosario

- **Flutter:** es SDK desarrollado por Google para crear aplicaciones móviles tanto para Android como para iOS (Apple) (Aures Tic, 2021) .
- **Dart:** es un lenguaje open source desarrollado en Google con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos y con análisis estático de tipo (Diví, 2019).
- **NodeJS:** es un entorno en tiempo de ejecución multiplataforma para la capa del servidor basado en JavaScript. Es un entorno controlado por eventos diseñado para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo.
- **MongoDB:** es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico (Robledano, 2019).
- **FrontEnd:** es la parte del desarrollo que se dedica a la parte frontal de un sitio web, aplicación, etc. Es decir, del diseño del mismo, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos (García, 2021).
- **BackEnd:** es la parte del desarrollo que se dedica al procesado de la información que se usa en el parte del FrontEnd (García, 2021).
- **Aplicación móvil:** aplicación diseña para ser ejecutada en un dispositivo móvil.
- **Aplicación híbrida:** son aquellas aplicaciones que se diseñan para ser utilizadas en diferentes sistemas operativos (Barrera, 2020).
- **Amazon Web Services (AWS):** es un proveedor de servicios en la nube, nos permite disponer de almacenamiento, recursos de computación, aplicaciones móviles, bases de datos y un largo etcétera en modalidad de Cloud Computing (Gimenez, 2020)

# 1. Introducción

Este proyecto consiste en el planteamiento, desarrollo y puesta en marcha de una aplicación móvil que permitirá a los usuarios de una ciudad, compartir taxi para realizar determinados viajes con el objetivo de ahorrarse dinero. Esta aplicación estará disponible tanto para dispositivos bajo el sistema operativo Android como el sistema operativo iOS. En esta memoria, se detallan todos los pasos desde el surgimiento de la idea, contacto con el cliente para establecer los requisitos de la misma, análisis técnico, desarrollo de la aplicación, pruebas de funcionamiento y puesta en marcha.

En este apartado se comenta, de forma breve, los siguientes apartados: primero, un breve planteamiento del problema a solucionar con el proyecto, posteriormente se mencionan los objetivos a lograr, a continuación, la metodología de trabajo junto con las tecnologías y herramientas requeridas para llevar a cabo dichos objetivos y, por último, se plantea la estructura de trabajo.

## 1.1. Planteamiento del problema

Este problema viene dado por la idea de un cliente, en concreto, una empresa de taxis madrileña, que quería incorporar alguna forma para que sus clientes pudieran compartir taxis en viajes que fuera repetitivos, como por ejemplo desde la estación de Atocha hasta una terminal de barajas.

La idea, no solo reside en hacer que sus clientes ahorren dinero al compartir taxi, sino en atraer a clientes a este formato de transporte que en condiciones normales no lo tomarían por considerarlo caro, pero que, con esta reducción de precio, los clientes valorarán esta opción de transporte.

En conclusión, en este Trabajo de Fin de Grado se desarrolla una aplicación móvil que es capaz de ofrecer estos viajes compartidos en base a una serie de requisitos que se explicarán posteriormente.

## 1.2. Objetivos

El objetivo principal de este proyecto reside en el desarrollo de una aplicación híbrida móvil, que permita a los usuarios reservar taxis para realizar trayectos prefijados por las compañías, con el objetivo de abaratar los costes de estos viajes.

Para conseguir esto, se desarrollará una aplicación móvil que disponga de una interfaz intuitiva, sencilla, accesible y, sobre todo, fácil de usar, incluso para aquellas personas con bajos conocimientos informáticos.

Además, de la reserva de viajes, la aplicación ofrecerá otros dos apartados, uno para revisar todos los viajes realizados por el usuario, y otro, para la revisión y corrección de la información personal del usuario (nombre, apellidos, teléfono, etc.).

### 1.3. Metodología

Para la realización de este proyecto, no se ha utilizado un desarrollo en cascada propio de las metodologías tradicionales, que, a lo largo del tiempo, han demostrado tener un desarrollo lento y muy propenso a grandes errores y de difícil solución. En base a esto, se ha decidido hacer uso de las metodologías ágiles (IEBS, 2021), que aportan una mayor calidad en producto software creado, al contar con una estructura más ordenada y disciplinada.

En concreto, se ha seguido el modelo de desarrollo iterativo e incremental, donde el principal objetivo, es que al final de cada una de las iteraciones se disponga una aplicación funcional, donde se puedan ver distintos avances, es decir, distintas funcionalidades.

El desarrollo iterativo e incremental, consta de tres fases principales y diferenciadas: por un lado, el plan inicial, donde se recopila información respecto al proyecto a realizar, se instalan los diferentes servicios, entornos de trabajo a utilizar, etc. Por otro lado, el propio desarrollo iterativo, donde se procede al desarrollo de la aplicación. En esta fase, se incluyen pruebas de funcionamiento para comprobar que se cumplen los requisitos previamente establecidos. Por último, la puesta en marcha, fase que consiste en la instalación del trabajo resultante de la fase anterior en el entorno de trabajo del cliente, de forma que se puede comprobar si han cumplido, y, de forma exitosa, los requisitos previamente establecidos, y, además, de forma correcta, es decir, sin errores.

El plan inicial:

- Inicialmente, se procede a analizar el problema encontrado, se plantean destinar alternativas para dar solución a dicho problema y se compara esta solución con las alternativas presentes en el mercado.

- Posteriormente, se redactan todos los requisitos, tantos los funcionales como los no funcionales.
- Por último, se procede a la instalación y preparación de todos los servicios y entornos de trabajo a utilizar.

El desarrollo iterativo:

- **Análisis:** se escogen aquellos requisitos que se han de implementar en esta iteración, y, también se han de corregir los errores presentes de la iteración anterior, o aquellos detalles que se quieran cambiar, ya sea por mal funcionamiento o porque se ha visto que no satisfacen al 100% los requisitos establecidos.
- **Diseño:** se refina la visión general previamente establecida y se detalle cómo se van a incorporar los requisitos establecidos en la fase anterior.
- **Implementación:** se procede a la implementación de los requisitos decididos en fase de análisis.
- **Pruebas:** se procede a la validación por medio de diversas pruebas acerca del funcionamiento y correcto diseño de aplicación en dicha iteración.

La puesta en marcha:

- Simplemente, se procede a la instalación del producto resultante de la iteración en un entorno de trabajo del cliente. En este entorno se procede a comprobar que el funcionamiento es correcto, y, además, satisface los requisitos del cliente. Estas pruebas deben ser realizadas por el cliente o, en su defecto, por personal sin grandes conocimientos informáticos, de forma que ofrezca otra perspectiva sobre el funcionamiento de la aplicación.

*Figura 1.1: Desarrollo iterativo e incremental*

Cabe destacar, que, en un proyecto real, cada una de las iteraciones tiene un tiempo límite para su realización. Este tiempo viene determinado por varios factores como son la dificultad, número de personas que forman parte del equipo, etc. Sin embargo, al tener que compaginar este proyecto con otras asignaturas que alteraban el flujo de trabajo, en varias ocasiones los tiempos límites han sido expandidos.

### 1.4. Tecnologías y herramientas

En el siguiente apartado se comentarán las principales herramientas utilizadas durante el desarrollo del proyecto.

- **FrontEnd:** Se ha utilizado el SDK de Flutter junto con el lenguaje de programación Dart para el desarrollo del FrontEnd. Para programar esta parte del desarrollo, se ha hecho uso del entorno de desarrollo integrado Android Studio en primeras instancias del desarrollo, y, finalmente, IntelliJ Ultimate.
- **BackEnd:** Se ha utilizado NodeJS para el desarrollo del servidor que utiliza la aplicación desarrollada. En este servidor se almacenan las diferentes API's con los datos necesarios para ser consumidos por el FrontEnd. Para el desarrollo del BackEnd se ha hecho uso del editor de código Visual Studio Code, que por medio de su herramienta de ssh permite conectarte con el proyecto almacenado en el servidor de Amazon Web Services.
- **Servidor remoto:** Se ha utilizado Amazon Web Services (Gimenez, 2020), AWS, como servicio de almacenamiento remoto del servidor de la aplicación, lo que permite, que esta no este conectada de forma local al servidor.
- **Controlador de versiones:** Se ha utilizado como controlador de versiones Git y para el almacenamiento remoto de las versiones se ha hecho uso de GitHub. Para el control de las diferentes ramas de trabajo y su correcta fusión se ha hecho uso de GitHub Desktop.
- **Documentación:** Se ha utilizado Microsoft Word para el desarrollo de la memoria del proyecto. Microsoft Excel para el desarrollo de tablas y gráficas. yED Graph Editor para el desarrollo gráfico de los casos de uso y otros dibujos realizados para la redacción de la memoria. Por último, para la creación y desarrollo de los prototipos se ha utilizado una tableta gráfica que permite crearlos con el máximo detalle posible.
- **Dispositivos:** Para el desarrollo completo de este proyecto se ha hecho uso de un ordenador personal para el desarrollo del código y de la memoria. Además,

para las pruebas del código desarrollado se ha usado un dispositivo móvil personal, así como diferentes versiones de emuladores con diferentes resoluciones de pantallas para comprobar que el FrontEnd escalaba correctamente.

### 1.5. Estructura del trabajo

En este apartado de la memoria, se procede a redactar las áreas principales que cubren el ciclo de vida del desarrollo de sistemas software que se ha utilizado para el desarrollo de este Trabajo de Fin de Grado.

- **Estudio del problema:** se realiza un análisis previo que trata de valorar tanto el problema en sí mismo, como el desarrollo de la solución a este problema. En este apartado se tratan diversas cuestiones, entre otras se plantea el contexto del problema, como son las tecnologías existentes, mercado de desarrollo, etc. También, se redacta el problema, que nuestra solución viene a solventar, en este caso, el problema de los altos costes de los taxis, que haciendo uso de esta aplicación se abaratan los costes.
- **Gestión del proyecto software:** una vez analizado el problema, se plantea el alcance del proyecto, se organiza un plan para el desarrollo del mismo, se estudian los recursos existentes y los necesarios para llevar a cabo el proyecto y, por último, se analizan los posibles riegos para disminuirlos.
- **Solución:** este es el apartado más importante de la memoria, es el núcleo del proyecto. En primer lugar, se define y redacta de la forma más detalla posible, la solución al problema anteriormente planteado. Además, se han de incluir en esta definición requisitos, tanto funcionales como no funcionales, casos de uso, gráficos, todo aquello que ayude a definir de la mejor forma posible la solución al problema planteado. También, se detallan aspectos relacionados con la implementación de la aplicación y problemas encontrados durante su desarrollo. Por último, de detallarán las pruebas a realizar, destacando las pruebas unitarias.
- **Normativa vigente:** se explica, de una forma no muy detalla, la normativa actual que guarda relación con la temática del proyecto, tanto desde el punto de vista legislativo como desde el punto de vista informático.
- **Producto final:** en este apartado, se mostrará por medio de texto, imágenes e incluso videos el producto resultando del desarrollo del problema y que es la

solución planteada para el mismo. Se corresponde con la entrega que se hace al cliente.

- **Evaluación:** en este apartado final de la memoria del proyecto, se demuestra la validez del mismo respecto al cumplimiento de los requisitos, tanto funcionales como no funcionales, y respecto a la resolución final del problema.

# 2. Estudio del problema

En el apartado que se trata a continuación, se analiza el problema objetivo del proyecto de forma breve, es decir, desde un punto de vista general.

## 2.1. El contexto del problema

Este problema viene dado por la idea de un cliente, en concreto, una empresa de taxis madrileña, que quería incorporar alguna forma para que sus clientes pudieran compartir taxis en viajes que fuera repetitivos, como por ejemplo desde la estación de Atocha hasta una terminal de Barajas.

La idea, no solo reside en hacer que sus clientes ahorren dinero al compartir taxi, sino en atraer a clientes a este formato de transporte que en condiciones normales no lo tomarían por considerarlo caro, pero que, con esta reducción de precio, los clientes valorarán esta opción de transporte.

En conclusión, es este Trabajo de Fin de Grado de desarrolla una aplicación móvil que es capaz de ofrecer estos viajes compartidos en base a una serie de requisitos que se explicarán posteriormente

## 2.2. El estado del arte

En la actualidad, se encuentra en funcionamiento un gran número de plataformas preparadas para transportar a los usuarios al destino deseado. En esta sección, veremos las desventajas y ventajas de cada una de ellas, para realizar una conclusión final donde se explicará porque el usuario acabará decantándose por nuestra plataforma.

### 2.2.1. Aplicaciones similares

En esta sección iremos viendo una a una a los principales competidores que presenta nuestro proyecto.

#### 2.2.1.1. Taxis tradicionales.

En cuanto a este medio de transporte poco hay que explicar, ya que, estamos hablando de uno de los principales medios de transporte de la actualidad.

- **Ventajas:** La principal ventaja de los taxis reside en su disponibilidad, las 24 horas del día los 365 días del año.

- **Desventajas:** Actualmente, los taxis cuentan con un precio que no está al alcance de todos los bolsillos. Además, es muy complicado encontrar alguien con quién compartir taxi si no lo conoces previamente.

2.2.1.2. Blablacar

En cuanto a plataforma que permite compartir transporte para abaratar gastos, en Blablacar encontraríamos la principal competencia. Pero, Blablacar está orientada a grandes viajes y no a pequeños trayectos urbanos como es el caso de BlablaTaxi (BlaBlaCar, 2021).

- **Ventajas:** Permite compartir medio de transporte abaratando costes entre sus usuarios.
- **Desventajas:** Esta orientada a grandes trayectos o viajes, y no a trayectos urbanos para usuarios del día a día.

Esta aplicación se puede encontrar tanto en la App Store como en Google Play.

2.2.1.3. Cabify

En este caso, hablamos de una competencia directa los taxis, que se ha instaurado en nuestro país recientemente (Xataka, 2019).

- **Ventajas:** Disponibilidad las 24 horas del día los 365 días del año. Más comodidad y lujo en los trayectos.
- **Desventajas:** De nuevo se hace muy complicado abaratar costes por usuario encontrando otros para la misma ruta.

En este caso, para poder hacer uso de este servicio, se puede realizar una reserva en la web de la empresa o descarga la app que se encuentra tanto en la App Store como en Google Play.

2.2.1.4. Autobuses y metro

En ambos casos, hablamos del transporte público por excelencia.

- **Ventajas:** Si el usuario desea el precio más reducido este es medio de transporte. Además, cuenta con unos horarios fijos y predefinidos.
- **Desventajas:** En muchas ocasiones, el usuario debe de utilizar más de una línea de metro o de bus para llegar a su destino final, haciendo que este transporte sea

más lento que la competencia. Muchas ocasiones no hay posibilidad de terminar el trayecto en el destino deseado y el usuario debe terminar la ruta caminando.

#### 2.2.2. Opinión sobre el estado del arte

Después de esta breve exposición de ventajas e inconvenientes de las principales alternativas que se presupone que tendrá nuestro proyecto, se observa y se concluye que todas ellas carecen de la principal ventaja y funcionalidad de BlablaTaxi, que es la posibilidad de compartir un trayecto o viaje urbano con otros usuarios anónimos, abaratando así los costes para cada uno de los mismos.

Por otro lado, encontramos la alternativa del autobús o del metro, en función de la ciudad, donde podemos compartir transporte público con los menores costes para el usuario, pero este tipo de transporte solo nos sirve para pequeños viajes, ya que en su mayoría el usuario tendría que utilizar más de uno de estos transportes, que, además, normalmente, son más lentos.

Finalmente, encontraremos plataformas como Cabify que ofrecen una mayor comodidad y lujo en los trayectos, sin embargo, su precio hace que el usuario más común termine usando nuestra plataforma por el ahorro que supone tanto a corto como a largo plazo.

### 2.3. La definición del problema

Como bien se ha ido definiendo en los apartados de esta memoria, los taxis, actualmente, no cuentan con un modelo de negocio que les permita hacer una competencia directa y real, a la hora de compartir trayectos entre usuarios para abaratar costes.

Existen múltiples alternativas que permiten a los usuarios compartir transporte público para hacer trayectos largos, como puede ser Blablacar, pero ninguna empresa de este mercado de los transportes está, ni preparada, ni orientada para compartir transporte en pequeños trayectos que se pueden considerar cotidianos. El objetivo es hacerse con un nicho de mercado específico.

En definitiva, el objetivo de los taxis reside en lograr una competencia directa a las distintas alternativas de transporte público, en especial, a las VTC (Xataka, 2019). Con esta aplicación, lograrán competir en el ámbito de los viajes prefijados y, además, con

precios que podrían llegar a ser, en función del número de viajeros, bastante más económicos que los ofrecidos por estas alternativas.

# 3. Gestión del proyecto software

En este apartado del documento se procede a detallar como se ha realizado el desarrollo del proyecto, pero desde un punto de vista empresarial. Teniendo en cuenta, que la idea del proyecto es personal y que el desarrollo del mismo se ha llevado a cabo por un solo individuo, el objetivo reside en realizar una simulación de un entorno empresarial.

Para realizar esta simulación se tratarán diversos apartados como son: alcance del proyecto, plan de trabajo, gestión de recursos, presupuesto y gestión de riesgos.

## 3.1. Alcance del proyecto

En este primer apartado de la simulación se tratarán los siguientes aspectos: definición del proyecto, objetivos del proyecto, límites del proyecto, productos entregables, criterios de aceptación y las diferentes restricciones a tener en cuenta durante el desarrollo del proyecto.

### 3.1.1. Definición del proyecto

El proyecto consiste en el desarrollo de una aplicación híbrida móvil para compartir taxis entre usuarios, para trayecto previamente fijados, con el objetivo de disminuir los costes por viajero.

Al ser una aplicación híbrida debe tener compatibilidad con Android y con iOS. Además, se desarrollará una interfaz intuitiva y lo más accesible posible a todos los usuarios, incluidos aquellos con escasos conocimientos informáticos.

Inicialmente, la aplicación se implementará en un único lenguaje, el castellano, aunque no se descarta que en posteriores versiones se incorporen nuevos lenguajes como el inglés, francés, etc.

La versión final de la aplicación deberá permitir a los usuarios buscar trayectos, reservarlos y cancelarlos, siempre y cuando cumplan los criterios para poder hacer tales acciones. Además, el usuario tendrá una pantalla donde podrá revisar sus viajes y otra para poder modificar la información correspondiente con su perfil si así fuera necesario.

Por último, respecto a la seguridad de la aplicación, al hacer uso de información personal se debe garantizar que el proyecto cumpla con los máximos requisitos de seguridad para evitar filtraciones de esos datos.

### 3.1.2. Objetivos del proyecto

Los objetivos básicos a partir de los cuales podemos consideras como finalizado el proyecto, son los siguientes:

- Desarrollo de una aplicación móvil que te permita compartir taxis entre diferentes usuarios.
- Debe ser una aplicación híbrida, es decir, compatible tanto con Android como con iOS.
- Debe contar con una interfaz intuitiva, sencilla y accesible para todo tipo de usuarios.
- La aplicación debe permitir la reserva de trayectos para los usuarios.
- La aplicación deberá permitir eliminar los perfiles de usuarios, eliminando todos sus datos.
- Se deben realizar pruebas de forma periódica con el objetivo de encontrar errores y, en caso de encontrarlos, corregirlos para mejorar la aplicación.
- Se debe garantizar la máxima seguridad en el proyecto, garantizando el cumplimiento de la ley de protección de datos, entre otras leyes y normativas.
- La aplicación debe tener el FrontEnd y BackEnd claramente diferenciados, usando para cada uno de ellos los lenguajes de programación más efectivos.
- El BackEnd deberá forma un servidor al cual el FrontEnd deberá conectarse para poder obtener datos de la base de datos. Este servidor se encontrará alojado en un repositorio online.
- Se deberá contar con una base de datos, la cual surtirá de datos al FrontEnd por medio de API’s.

### 3.1.3. Límites del proyecto

En esta sección se procede a comentar las limitaciones y restricciones del proyecto. Esto permite evitar el surgimiento de imprevistos que pueden provocar el fracaso del proyecto.

- Se asegura el funcionamiento de la aplicación en aquellos móviles con acceso a internet.
- Se asegura el correcto funcionamiento de todos los Widgets en las versiones más recientes de Android y iOS.

- Para el correcto funcionamiento de la aplicación no será necesario proceder con la instalación de algún tipo de programa, servicio o aplicación extra.

### 3.1.4. Productos entregables

Para asegurar un correcto desarrollo del proyecto, se propone la entrega periódica de diversas versiones el producto software realizado al cliente. El objetivo de estas entregas es mantener un seguimiento constante del producto desarrollado y asegurarse de que el producto que se está desarrollando cumple con los requisitos del cliente.

Teniendo en cuenta que este producto se está desarrollando dentro del marco de las metodologías ágiles, se ha propuesto la entrega de una versión del producto software cada 3 semanas, es decir, la duración de los incrementos es de 3 semanas. Teniendo en cuenta que el proyecto comenzó el 1 de febrero del 2022 se podrán realizar un total de 6 incrementos, es decir, un total de 6 entregas.

Por otro lado, al inicio del proyecto, se hizo entrega de una documentación inicial, prepuesto, requisitos, etc. Esta documentación inicial debe ser aprobada por el cliente para poder proceder al desarrollo del producto software.

### 3.1.5. Criterios de aceptación

Tras una serie de conversaciones con el cliente, se ha decidido que para que el proyecto sea aceptado como válido se han de cumplir los siguientes puntos:

- Comprobar y validar tantos los requisitos funcionales como los no funcionales.
- El cliente podrá probar el producto a medida que se va desarrollando con el objetivo de encontrar errores y de implementar soluciones a estos errores, así como, mejoras de funcionamiento que el cliente vea conveniente implementar.
- Se deberán cumplir todos los objetivos marcados en la sección de objetivos, que previamente fueron consensuados con el cliente.
- Todas las pruebas de caja negra y caja blanca han de tener un resultado exitoso respecto a los resultados esperados.

### 3.1.6. Restricciones

Se han previsto dos restricciones a nivel de planificación del proyecto:

- Por un lado, el coste final del proyecto puede superar, como máximo, en un 3% al previsto inicialmente.

- Por otro lado, la duración total del proyecto puede excederse, como máximo, en un total de 4 semanas respecto a lo pactado en la planificación inicial.

## 3.2. Plan de trabajo

En este aparatado se procede a describir las tareas a realizar necesarias para el éxito del proyecto, así como estimaciones sobre la duración de las mismas acompañadas de diferentes diagramas que faciliten el entendimiento visual y rápido del plan de trabajo.

### 3.2.1. Identificación de fases

Recordando que este proyecto se realiza siguiendo la estructura de las metodologías ágiles, podemos dividir el mismo en 3 fases principales:

- **Plan inicial del proyecto:** en esta fase el objetivo consiste en redactar tanto los requisitos como los casos de uso. Para ello, se deberán concertar una serie de reuniones con el cliente para detallarlos. En definitiva, se realiza un estudio del problema actual. En este caso, durante esta fase, también se procede al aprendizaje de las tecnologías necesarias (Flutter y Dart).
- **Proceso iterativo:** como se comentó anteriormente, este proyecto consta de 6 iteraciones. Antes de comenzar cada una de ellas, se deberá detallar aquellos requisitos que se van a implementar o aquellos objetivos que se deseen cumplir en dicha iteración. Además, al acabar la misma se procede a la revisión y realización de pruebas.
- **Revisión y puesta en marcha:** una vez se considera que el producto está listo, se procede a su entrega al cliente.

### 3.2.2. Identificación de tareas

En este apartado se procede a comentar que tareas se realizan por cada una de las fases identificadas en el apartado anterior.

#### 3.2.2.1. Plan inicial del proyecto

En el caso de este proyecto, el plan inicial conlleva la tarea extra de realizar el aprendizaje de la tecnología a emplear, que es Flutter y Dart.

Además, se realizarán una serie de reuniones con el cliente para realizar el estudio del problema, analizar el mercado y la competencia y desarrollar tanto los requisitos como los casos de usuario.

Por último, se procede a la instalación de los entornos de trabajo necesarios.

3.2.2.2. Proceso iterativo

Como comentamos en apartado anteriores, el proyecto consta de 6 iteraciones, pero para comentar este apartado específico añadiremos una extra, que parte de ella corresponde a la fase anterior.

- **Iteración 0:** además de la especificación de los requisitos y de los casos de uso, se procede realizar el aprendizaje las tecnologías a usar, Flutter y Dart. Estas tareas, son más bien parte del plan inicial del proyecto, por ello las he incluido como iteración 0.
- **Iteración 1:** esta es la primera iteración del proceso iterativo. En ella, se procederá al diseño e implementación de la base de datos, creación y conexión del BackEnd con la base de datos y la instalación de los plugins necesarios para desarrollo del proyecto. Además, se procederá a la implementación del registro de usuarios.
- **Iteración 2:** esta es la primera iteración donde se procede al desarrollo e implementación de los primeros requisitos y casos de uso. En concreto, se procede a la implementación del *login* en la aplicación y realización de pruebas unitarias. Además, han corregirse los errores provenientes del registro.
- **Iteración 3:** en esta iteración se procede a la implementación del proceso de visualización de los trayectos realizados por el usuario, corrección de errores provenientes del *login* y realización de pruebas unitarias.
- **Iteración 4:** en esta iteración se procede a la implementación del proceso de visualización y modificación del perfil de usuario, corrección de errores provenientes de la visualización de los trayectos realizados por el usuario y realización de pruebas unitarias.
- **Iteración 5:** en esta iteración se procede a la implementación del proceso de visualización, reserva, cancelación y confirmación de los trayectos, corrección de errores provenientes del perfil de usuario y realización de pruebas unitarias.
- **Iteración 6:** en esta iteración se procede a la implementación de aquellas funcionalidades que en iteraciones anteriores no hayan sido desarrolladas correcta y completamente, corrección de errores provenientes de la reserva de trayectos y realización de pruebas unitarias.

3.2.2.3. Revisión y puesta en marcha

En esta fase final, se procede a realizar las pruebas de caja negra necesarias para verificar que requisitos se han cumplido. Se procede a corregir aquellos fallos que se encuentren y, por último, se lleva a cabo una reunión con el cliente para la entrega final.

### 3.2.3. Estimación y planificación de tareas

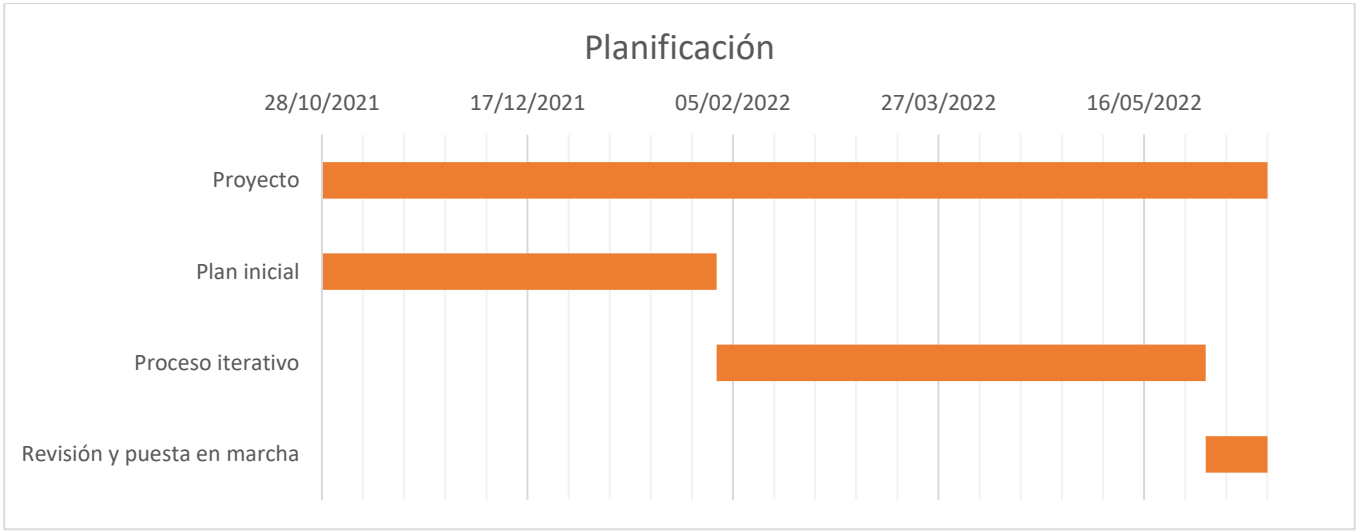
Para la estimación y planificación de tareas se han realizado una serie de tablas y diagramas de Gantt.

Este proyecto se inicia el 28 de octubre del 2021 y finaliza el 15 de junio del 2022, es decir, tiene una duración de 230 días.

*Tabla 3.1: Planificación de fases del proyecto*

| Nombre de la tarea | Duración | Comienzo | Fin |
|---|---|---|---|
| **Proyecto** | 230 | 28/10/2021 | 15/06/2022 |
| **Plan inicial** | 96 | 28/10/2021 | 01/02/2022 |
| **Proceso iterativo** | 119 | 01/02/2022 | 31/05/2022 |
| **Revisión y puesta en marcha** | 15 | 31/05/2022 | 15/06/2022 |

*Figura 3.1: Diagrama de Gantt de las fases del proyecto*



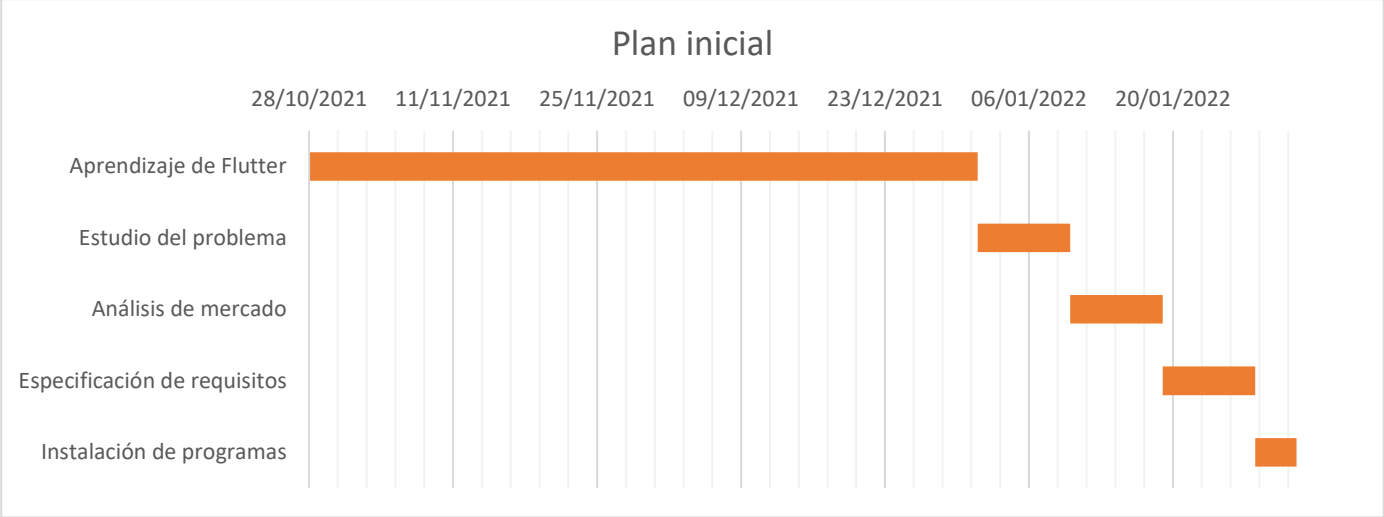

#### 3.2.3.1. Plan inicial

Las tareas para realizar en esta fase dan comienzo el 28 de octubre del 2021 y finalizan el 01 de febrero del 2022.

*Tabla 3.2: Planificación del plan inicial*

| Plan inicial | | | |
|---|---|---|---|
| **Nombre de la tarea** | **Duración** | **Comienzo** | **Fin** |
| **Aprendizaje de Flutter** | 65 | 28/10/2021 | 01/01/2022 |
| **Estudio del problema** | 9 | 01/01/2022 | 10/01/2022 |
| **Análisis de mercado** | 9 | 10/01/2022 | 19/01/2022 |
| **Especificación de requisitos** | 9 | 19/01/2022 | 28/01/2022 |
| **Instalación de programas** | 4 | 28/01/2022 | 01/02/2022 |

*Figura 3.2: Diagrama de Gantt del plan inicial*




3.2.3.2. Proceso iterativo

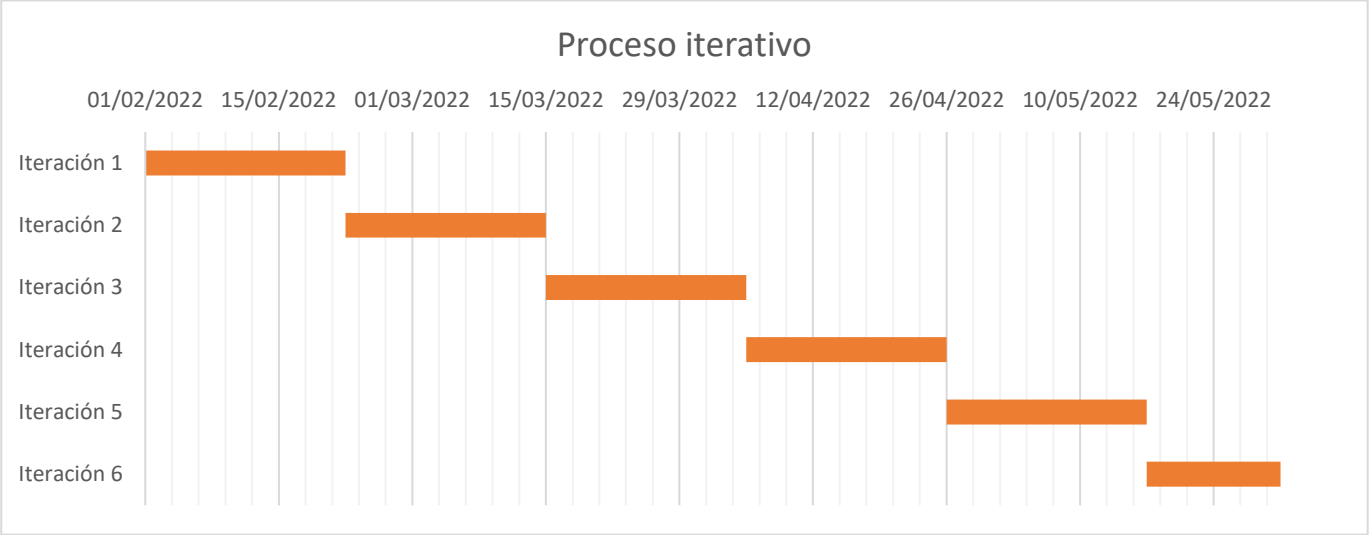
Las tareas que se realizan en esta fase comienzan el 01 de febrero del 2022 y finalizan el 31 de mayo del 2022. En esta fase encontraremos 6 iteraciones de una duración similar entre ellas, a excepción de la última.

Para cada una de estas iteraciones, también se han especificado las tareas correspondientes.

*Tabla 3.3: Planificación del proceso iterativo*

| Proceso iterativo | | | |
|---|---|---|---|
| **Nombre de la tarea** | **Duración** | **Comienzo** | **Fin** |
| **Iteración 1** | 21 | 01/02/2022 | 22/02/2022 |
| **Iteración 2** | 21 | 22/02/2022 | 15/03/2022 |
| **Iteración 3** | 21 | 15/03/2022 | 05/04/2022 |
| **Iteración 4** | 21 | 05/04/2022 | 26/04/2022 |
| **Iteración 5** | 21 | 26/04/2022 | 17/05/2022 |
| **Iteración 6** | 14 | 17/05/2022 | 31/05/2022 |

*Figura 3.3: Diagrama de Gantt del proceso iterativo*




A continuación, se procede a definir cada una de las iteraciones.

*Tabla 3.4: Planificación de la iteración 1*

| Iteración 1 | | | |
|---|---|---|---|
| **Nombre de la tarea** | **Duración** | **Comienzo** | **Fin** |
| **Especificación de casos de uso** | 2 | 01/02/2022 | 03/02/2022 |
| **Diseño de la base de datos** | 3 | 03/02/2022 | 06/02/2022 |
| **Implementación BBDD** | 1 | 06/02/2022 | 07/02/2022 |
| **Creación FrontEnd** | 2 | 07/02/2022 | 09/02/2022 |
| **Creación BackEnd** | 2 | 09/02/2022 | 11/02/2022 |
| **Conexión BackEnd-BBDD** | 2 | 11/02/2022 | 13/02/2022 |
| **Implementación Firebase** | 1 | 13/02/2022 | 14/02/2022 |
| **Implementación registro usuarios** | 7 | 14/02/2022 | 21/02/2022 |
| **Pruebas** | 1 | 21/02/2022 | 22/02/2022 |

*Figura 3.4: Diagrama de Gantt de la iteración 1*

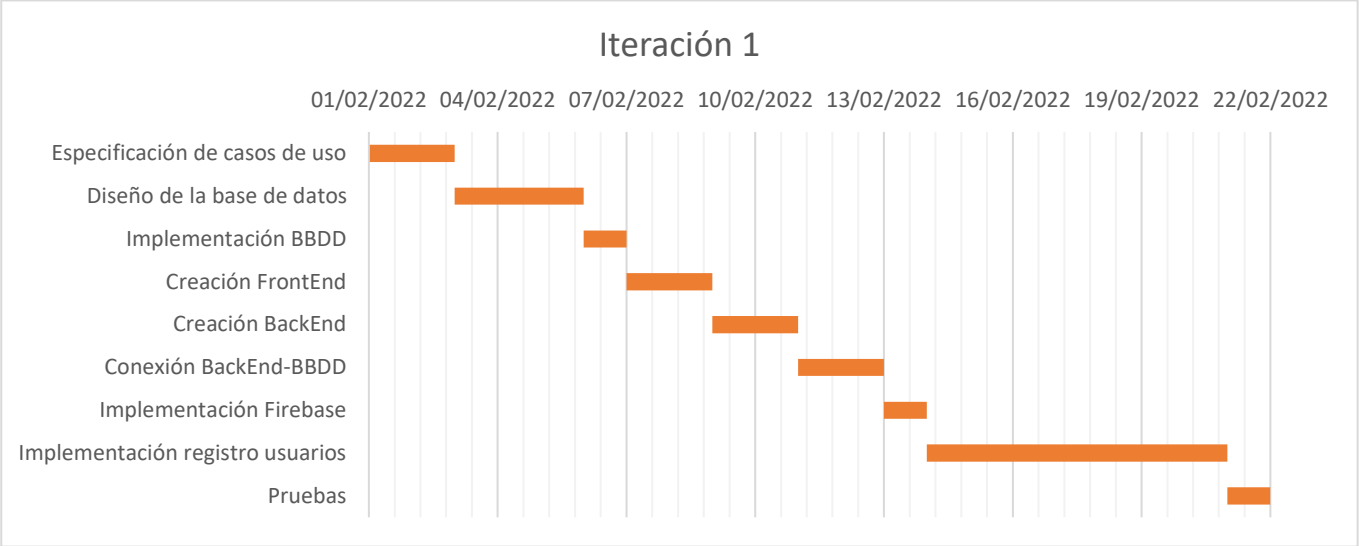



*Tabla 3.5: Planificación de la iteración 2*

| Iteración 2 | | | |
|---|---|---|---|
| **Nombre de la tarea** | **Duración** | **Comienzo** | **Fin** |
| **Corrección de errores** | 6 | 22/02/2022 | 28/02/2022 |
| **Especificación de casos de uso** | 2 | 28/02/2022 | 02/03/2022 |
| **Implementación login usuarios** | 11 | 02/03/2022 | 13/03/2022 |
| **Pruebas** | 2 | 13/03/2022 | 15/03/2022 |

*Figura 3.5: Diagrama de Gantt de la iteración 2*

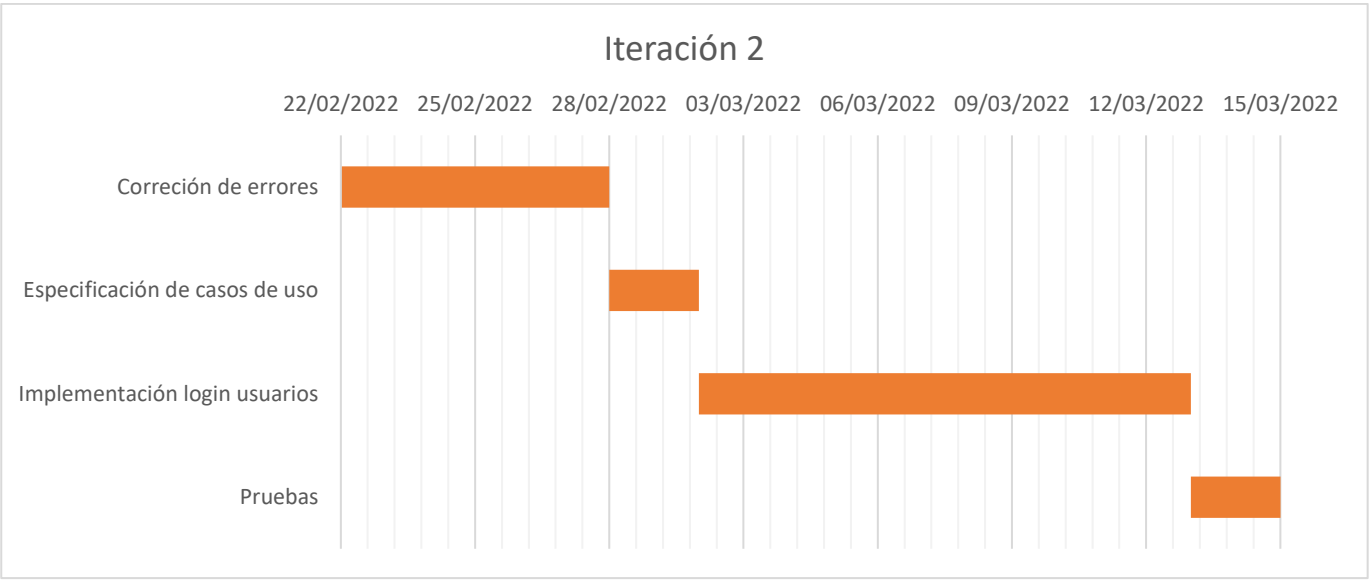


*Tabla 3.6: Planificación de la iteración 3*

| Iteración 3 | | | |
|---|---|---|---|
| **Nombre de la tarea** | **Duración** | **Comienzo** | **Fin** |
| **Corrección de errores** | 6 | 15/03/2022 | 21/03/2022 |
| **Especificación de casos de uso** | 1 | 21/03/2022 | 22/03/2022 |
| **Implementación ver viajes** | 12 | 22/03/2022 | 03/04/2022 |
| **Pruebas** | 2 | 03/04/2022 | 05/04/2022 |

*Figura 3.6: Diagrama de Gantt de la iteración 3*

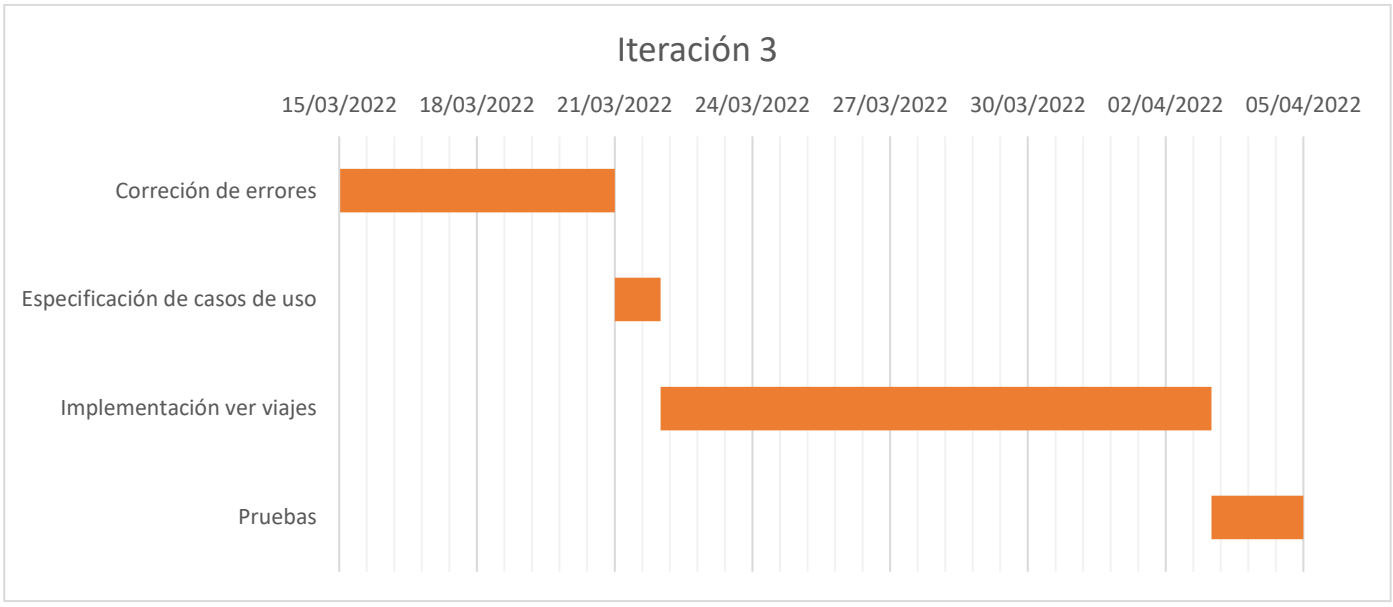



*Tabla 3.7: Planificación de la iteración 4*

| Iteración 4 | | | |
|---|---|---|---|
| **Nombre de la tarea** | **Duración** | **Comienzo** | **Fin** |
| **Corrección de errores** | 6 | 05/04/2022 | 11/04/2022 |
| **Especificación de casos de uso** | 1 | 11/04/2022 | 12/04/2022 |
| **Implementación perfil usuario** | 12 | 12/04/2022 | 24/04/2022 |
| **Pruebas** | 2 | 24/04/2022 | 26/04/2022 |

*Figura 3.7: Diagrama de Gantt de la iteración 4*

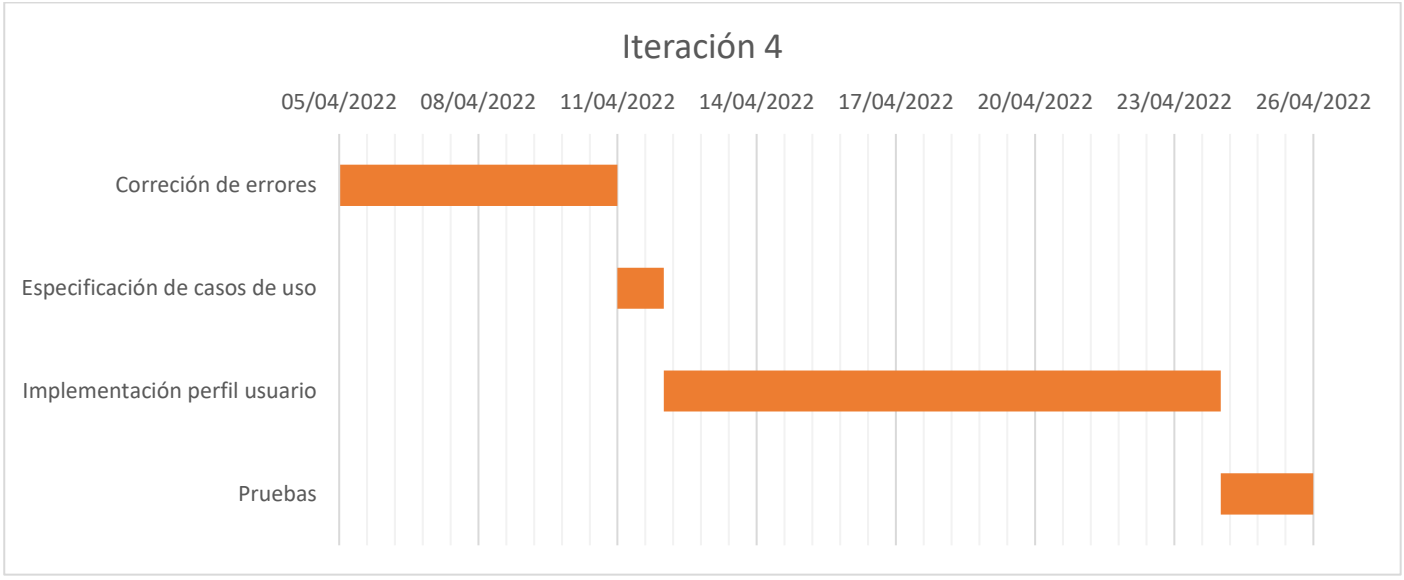



*Tabla 3.8: Planificación de la iteración 5*

| Iteración 5 | | | |
|---|---|---|---|
| **Nombre de la tarea** | **Duración** | **Comienzo** | **Fin** |
| **Corrección de errores** | 6 | 26/04/2022 | 02/05/2022 |
| **Especificación de casos de uso** | 1 | 02/05/2022 | 03/05/2022 |
| **Implementación gestión viajes** | 12 | 03/05/2022 | 15/05/2022 |
| **Pruebas** | 2 | 15/05/2022 | 17/05/2022 |

*Figura 3.8: Diagrama de Gantt de la iteración 5*

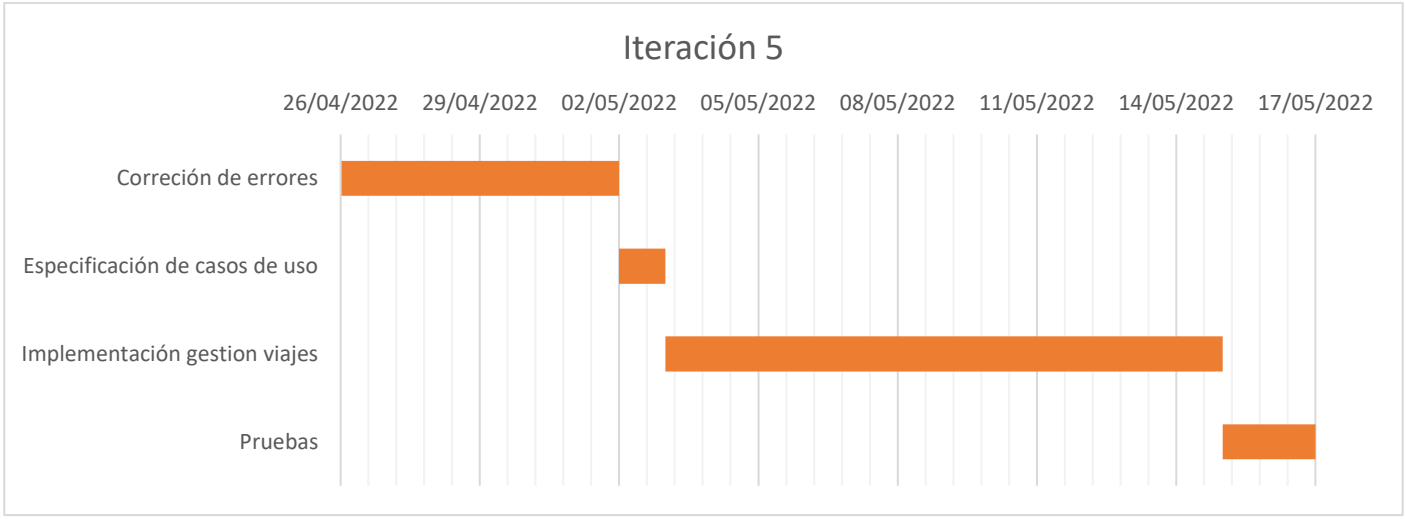
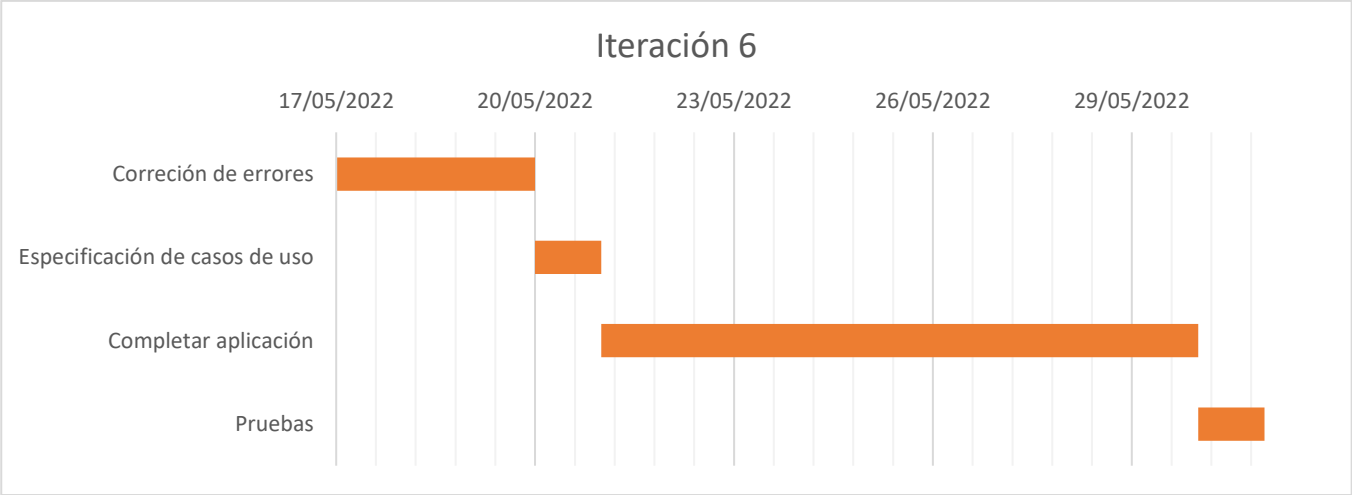


*Tabla 3.9: Planificación de la iteración 6*

| Iteración 6 | | | |
|---|---|---|---|
| **Nombre de la tarea** | **Duración** | **Comienzo** | **Fin** |
| **Corrección de errores** | 3 | 17/05/2022 | 20/05/2022 |
| **Especificación de casos de uso** | 1 | 20/05/2022 | 21/05/2022 |
| **Completar aplicación** | 9 | 21/05/2022 | 30/05/2022 |
| **Pruebas** | 1 | 30/05/2022 | 31/05/2022 |

*Figura 3.9: Diagrama de Gantt de la iteración 6*



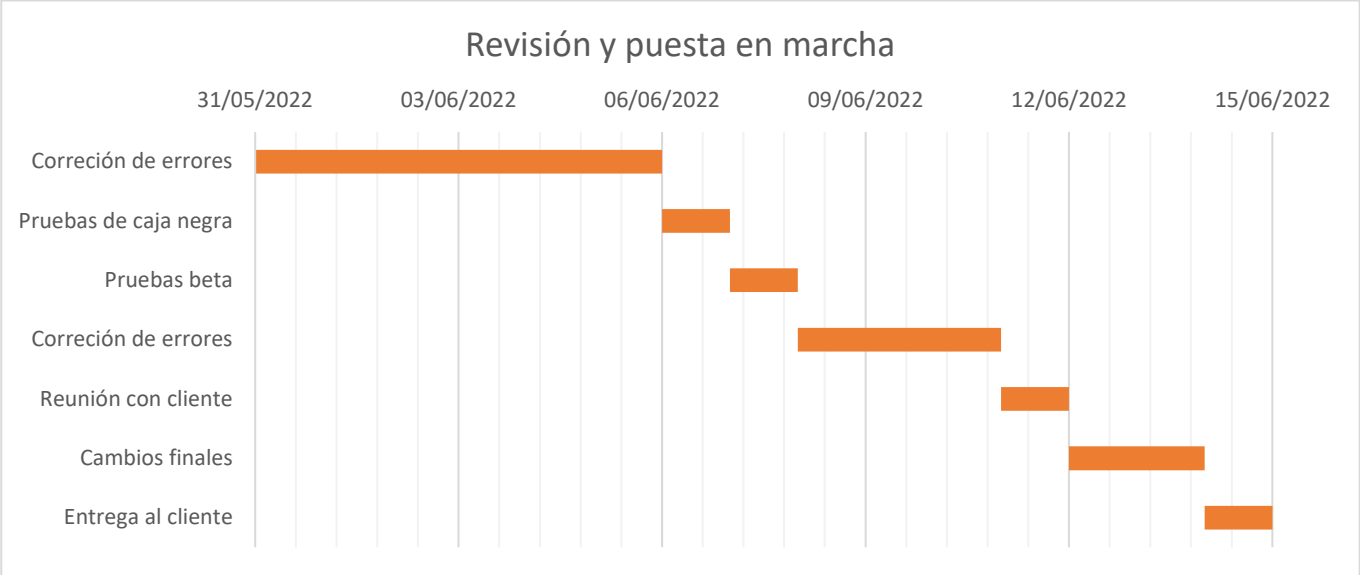

3.2.3.3. Revisión y puesta en marcha

Esta última fase del proyecto comienza el 31 de mayo del 2022 y finaliza el 15 de junio del 2022, día en el que está prevista la entrega al cliente.

*Tabla 3.10: Planificación de la revisión y puesta en marcha*

| Revisión y puesta en marcha | | | |
|---|---|---|---|
| **Nombre de la tarea** | **Duración** | **Comienzo** | **Fin** |
| **Corrección de errores** | 6 | 31/05/2022 | 06/06/2022 |
| **Pruebas de caja negra** | 1 | 06/06/2022 | 07/06/2022 |
| **Pruebas beta** | 1 | 07/06/2022 | 08/06/2022 |
| **Corrección de errores** | 3 | 08/06/2022 | 11/06/2022 |
| **Reunión con cliente** | 1 | 11/06/2022 | 12/06/2022 |
| **Cambios finales** | 2 | 12/06/2022 | 14/06/2022 |
| **Entrega al cliente** | 1 | 14/06/2022 | 15/06/2022 |

*Figura 3.10: Planificación de la revisión y puesta en marcha*




## 3.3. Gestión de recursos

En el siguiente apartado, se han detallado los recursos necesarios para la correcta realización del proyecto, es decir, para logar un éxito total en el mismo.

Cabe recalcar, que esto es una simple simulación de los recursos que se creen necesarios para llevar a cabo este proyecto a gran escala.

### 3.3.1. Especificación de recursos

*Tabla 3.11: Costes de recursos humanos*

| Recursos humanos | |
|---|---|
| **Recurso** | **Salario/año** |
| **Técnico de sistemas** | 23.461,00 € |
| **Desarrollador FrontEnd** | 33.175,00 € |
| **Desarrollador BackEnd** | 36.283,00 € |
| **Tester** | 29.083,00 € |
| **Jefe de proyecto** | 34.114,00 € |

Para calcular el salario de cada uno de los recursos necesarios, se ha utilizado la aplicación web *indeed*. A través de ella se han obtenido los diferentes salarios medios de: técnico de sistemas (Indeed, 2022), desarrollador de FrontEnd (Indeed, 2022), desarrollador de BackEnd (Indeed, 2022), Tester (Indeed, 2022) y jefe de proyecto (Indeed, 2022), que son necesarios para la correcta y exitosa realización del proyecto .

## 3.4. Presupuesto

En el apartado que se desarrolla a continuación, se ha realizado una descripción, a modo de estimación, de los costes derivados de realizar el proyecto. Para realizar una estimación lo más fiel posible, los costes se han divido en 3 secciones diferentes: Hardware, Software y recursos humanos.

### 3.4.1. Costes de Hardware

Para poder desarrollar el proyecto de forma exitosa se necesitará una inversión de: 5.500€, que está repartida entre los siguientes recursos hardware:

*Tabla 3.12: Costes de Hardware*

| Costes de Hardware | | | |
|---|---|---|---|
| **Recurso** | **Coste unitario** | **Unidades** | **Coste total** |
| **PCs de desarrollo** | 650,00 € | 5 | 3.250,00 € |
| **Monitor** | 125,00 € | 10 | 1.250,00 € |
| **Teclado** | 75,00 € | 5 | 375,00 € |
| **Ratón** | 25,00 € | 5 | 125,00 € |
| **Router** | 50,00 € | 1 | 50,00 € |
| **Smartphone** | 150,00 € | 3 | 450,00 € |
| **Coste total** | | | 5.500,00 € |

### 3.4.2. Costes de Software

Para el desarrollo correcto y legal del proyecto, se necesita adquirir una serie de licencias, en formato de empresa, para poder desarrollar el proyecto (Microsoft, 2022), (Microsoft, 2022) y (JetBrains, 2022). El coste de estas licencias asciende a 6.685€.

*Tabla 3.13: Costes de Software*

| Costes de Software | | | |
|---|---|---|---|
| **Recurso** | **Coste unitario** | **Unidades** | **Coste total** |
| **Licencia de Windows 10** | 259,00 € | 5 | 1.295,00 € |
| **Licencia de Microsoft Office** | 579,00 € | 5 | 2.895,00 € |
| **Licencia IntelliJ** | 499,00 € | 5 | 2.495,00 € |
| **Coste total** | | | **6.685,00 €** |

### 3.4.3. Costes de recursos humanos

Como se especificó en el apartado 2.3 los costes de recursos humanos son los siguientes:

*Tabla 3.14: Costes de recursos humanos*

| Recursos humanos | |
|---|---|
| **Recurso** | **Salario/año** |
| **Técnico de sistemas** | 23.461,00 € |
| **Desarrollador FrontEnd** | 33.175,00 € |
| **Desarrollador BackEnd** | 36.283,00 € |
| **Tester** | 29.083,00 € |
| **Jefe de proyecto** | 34.114,00 € |

Teniendo en cuenta, que contando con este número de trabajadores el proyecto podría ser realizado en torno a 4 meses, el coste de recursos humanos resultantes asciende a 52.038,67€.

### 3.4.4. Costes totales

Sumando los costes de cada uno de los apartados mencionados anteriormente, el coste de realizar el proyecto asciende a 64.223,67€.

*Tabla 3.15: Costes totales*

| Coste total | |
|---|---|
| **Tipo de coste** | **Coste** |
| **Recursos humanos** | 52.038,67 € |
| **Hardware** | 5.500,00 € |
| **Software** | 6.685,00 € |
| **Total** | **64.223,67 €** |

## 3.5. Gestión de riesgos

En este último apartado acerca de la gestión del software, se procede a detallar y describir con el máximo rigor posible un plan de gestión de riesgos.

Para detallar de la mejor forma posible dicho plan, se han de realizar una serie de tareas: identificación de riesgos, análisis de riesgos, estimación de probabilidad e impacto y, por último, exposición al riesgo.

### 3.5.1. Identificación de riesgos

Para esta sección, se sigue la metodología SMR, es decir, Strategic Risk Management (Kuna, 2008).

Para la identificación de los riesgos se ha creado una tabla, donde se puede observar una breve explicación de cada riesgo, la categoría a la que pertenece y si se trata de un riesgo genérico o específico.

La columna de la tabla ocupada por la categoría puede tener los siguientes valores: jefe de equipo, equipo de trabajo o cliente.

*Tabla 3.16: Identificación de riesgos*

| ID | Riesgo | Categoría | Tipo |
|---|---|---|---|
| **RI-01** | Errores de estimación para el tiempo de desarrollo | Jefe de equipo | Genérico |
| **RI-02** | Errores en la estimación del presupuesto | Jefe de equipo | Genérico |
| **RI-03** | Errores de estimación en el personal necesario | Jefe de equipo | Genérico |
| **RI-04** | Baja experiencia y conocimiento de los trabajadores | Equipo de trabajo | Específico |
| **RI-05** | Fallos de seguridad | Equipo de trabajo | Específico |
| **RI-06** | Modificación de los requisitos | Cliente | Genérico |
| **RI-07** | Adaptación a cambios del mercado | Las 3 categorías | Genérico |

### 3.5.2. Análisis de riesgos

Para este punto se procede a desarrollar una pequeña tabla con por riesgo identificado. Esta tabla cuenta con los siguientes campos:

- Riesgo para analizar.
- Consecuencia provocada por la aparición del riesgo.

*Tabla 3.17: RI-01 Errores de estimación para el tiempo de desarrollo*

| **RI-01** | **Errores de estimación para el tiempo de desarrollo** |
|---|---|
| **Riesgo** | Error en el cálculo del tiempo necesario para el desarrollo de la aplicación o de funcionalidades concretas. |
| **Consecuencias** | Se puede retrasar la entrega del proyecto. Se puede entregar el proyecto en el marco de tiempo establecido, pero con una funcionalidad reducida, o con una calidad inferior a la debida. |

*Tabla 3.18: RI-02 Errores en la estimación del presupuesto*

| **RI-02** | **Errores en la estimación del presupuesto** |
|---|---|
| **Riesgo** | Error en el cálculo del presupuesto necesario. Tanto a nivel de recursos humanos como de componentes y licencias necesarias. |
| **Consecuencias** | En caso de ser de personal, posible sobrecarga de los trabajadores contratados o, que alguno de los contratados no trabaje lo suficiente.<br>Puede ser necesario aplicar recortes para poder llevar el proyecto adelante con el presupuesto fijado. |

*Tabla 3.19: RI-03 Errores de estimación en el personal necesario*

| **RI-03** | **Errores de estimación en el personal necesario** |
|---|---|
| **Riesgo** | Error en el cálculo del personal necesario para llevar a cabo el proyecto. |
| **Consecuencias** | En caso de ser insuficiente el personal, se producirá una sobrecarga de los contratados, además de, un posible retraso en la entrega del producto final. |

*Tabla 3.20: RI-04 Baja experiencia y conocimiento de los trabajadores*

| **RI-04** | **Baja experiencia y conocimiento de los trabajadores** |
|---|---|
| **Riesgo** | Falta de conocimiento en la tecnología empleada para llevar a cabo el proyecto. |
| **Consecuencias** | Las tareas prefijadas, es probable, que terminen necesitando más tiempo del previsto inicialmente, provocando retrasos en la entrega del producto. |

*Tabla 3.21: RI-05 Fallos de seguridad*

| **RI-05** | **Fallos de seguridad** |
|---|---|
| **Riesgo** | Falta de conocimiento aplicado a la seguridad de la aplicación, paquetes o plugins mal instalados, brechas de seguridad en la aplicación, etc. |
| **Consecuencias** | Vulnerabilidad de los datos con los que trabaja la aplicación. Pero, lo más importante, es la pérdida de tiempo derivada de la reconstrucción del sistema a realizar cuando se detecten estos fallos. |

*Tabla 3.22: RI-06 Modificación de los requisitos*

| **RI-06** | **Modificación de los requisitos** |
|---|---|
| **Riesgo** | Planteamiento inicial erróneo de los requisitos, falta de entendimiento entre desarrollador y cliente o nuevas necesidades surgidas después de la especificación inicial de los mismos. |
| **Consecuencias** | Aumento del tiempo necesario para el desarrollo y entrega final de la aplicación. Además, de un posible aumento del prepuesto necesario para desarrollar el producto. |

*Tabla 3.23: RI-07 Adaptación a cambios del mercado*

| RI-07 | Adaptación a cambios del mercado |
|---|---|
| **Riesgo** | Cambios en los requisitos, en la forma de tratar la información, en cómo realizar las reservas de los viajes, provocado por cambios en el mercado o en la legalidad vigente. |
| **Consecuencias** | Aumento del tiempo necesario para el desarrollo y entrega final de la aplicación. Además, de un posible aumento del prepuesto necesario para desarrollar el producto. |

### 3.5.3. Estimación de probabilidad e impacto

A continuación, se procede a definir los valores de probabilidad e impacto, y, posteriormente, a dar cada riesgo un valor de probabilidad de aparición, así como de impacto. En primer lugar, definimos una tabla para las probabilidades de aparición. Después, otra tabla para la definición del impacto del riesgo.

*Tabla 3.24: Tabla de probabilidades de aparición de los riesgos*

| Expresión | Probabilidad | Valor numérico |
|---|---|---|
| Muy baja | 1% → 19% | 1 |
| Baja | 20% → 39% | 2 |
| Media | 40% → 59% | 3 |
| Alta | 60% → 79% | 4 |
| Muy alta | 80% → 99% | 5 |

*Tabla 3.25: Tabla de definición del impacto del riesgo*

| Expresión | Impacto | Valor numérico |
|---|---|---|
| Insignificante | 1 semana | 1 |
| Mínimo | 2 semanas | 2 |
| Medio | 1 mes | 3 |
| Importante | 2 meses | 4 |
| Catastrófico | Más de 2 meses | 5 |

Una vez definidas las probabilidades y el impacto, se ha de establecer particularmente ambas por riesgo definido.

*Tabla 3.26: Asociación de probabilidad e impacto por riesgo*

| ID | Riesgo | Probabilidad | Impacto |
|---|---|---|---|
| **RI-01** | Errores de estimación para el tiempo de desarrollo | Baja | Catastrófico |
| **RI-02** | Errores en la estimación del presupuesto | Muy Baja | Importante |
| **RI-03** | Errores de estimación en el personal necesario | Baja | Importante |
| **RI-04** | Baja experiencia y conocimiento de los trabajadores | Media | Medio |
| **RI-05** | Fallos de seguridad | Muy baja | Importante |
| **RI-06** | Modificación de los requisitos | Alta | Mínimo |
| **RI-07** | Adaptación a cambios del mercado | Alta | Mínimo |

### 3.5.4. Exposición al riesgo

Una vez definidos la probabilidad e impacto por riesgo, se ha de establecer la exposición a la cual está sometido cada riesgo. Esta exposición sale de una simple cuenta matemática que es: *Exposición = probabilidad * impacto.*

Para la probabilidad se toman valores medios de los porcentajes definidos en el apartado anterior.

*Tabla 3.27: Tabla con la exposición al riesgo*

| ID | Riesgo | Probabilidad | Impacto | Exposición |
|---|---|---|---|---|
| **RI-01** | Errores de estimación para el tiempo de desarrollo | Baja | Catastrófico | 1,5 |
| **RI-02** | Errores en la estimación del presupuesto | Muy Baja | Importante | 0,4 |
| **RI-03** | Errores de estimación en el personal necesario | Baja | Importante | 1,2 |
| **RI-04** | Baja experiencia y conocimiento de los trabajadores | Media | Medio | 1,5 |
| **RI-05** | Fallos de seguridad | Muy baja | Importante | 0,4 |
| **RI-06** | Modificación de los requisitos | Alta | Mínimo | 1,4 |
| **RI-07** | Adaptación a cambios del mercado | Alta | Mínimo | 1,4 |

Teniendo en cuenta esta tabla podemos determinar que los riesgos a tener más en cuenta son el 01 y el 04, que el 03, el 06, y el 07, también hay que tenerlos muy en cuenta y que el 02 y el 05 no hay que olvidarlos, pero estamos menos expuestos a ellos.

# 4. Definición de la solución

## 4.1. Descripción de la solución

La solución propuesta y adoptada, al problema encontrado, consiste en desarrollar una aplicación móvil que permita a los usuarios reservar el trayecto que se adecue a sus necesidades tratando de compartirlo con el máximo número de personas posible.

La aplicación desarrollada, ofrecerá al usuario diversos viajes en función de una serie de parámetros que él deberá de determinar. Estos parámetros serán: fecha, rango de horas, destino, etc. Una vez definidos estos valores, el sistema ofrecerá al usuario los mejores viajes posibles.

El usuario reservará una plaza en ese trayecto, de forma que otros pueden reservar otras plazas hasta completar el taxi. El precio total del viaje se dividirá entre las plazas reservadas, pagando cada uno la parte correspondiente.

Además, la aplicación ofrecerá la opción de visualizar los viajes realizados y reservados. Esto permitirá al usuario tener un resumen de los trayectos realizados usando la aplicación.

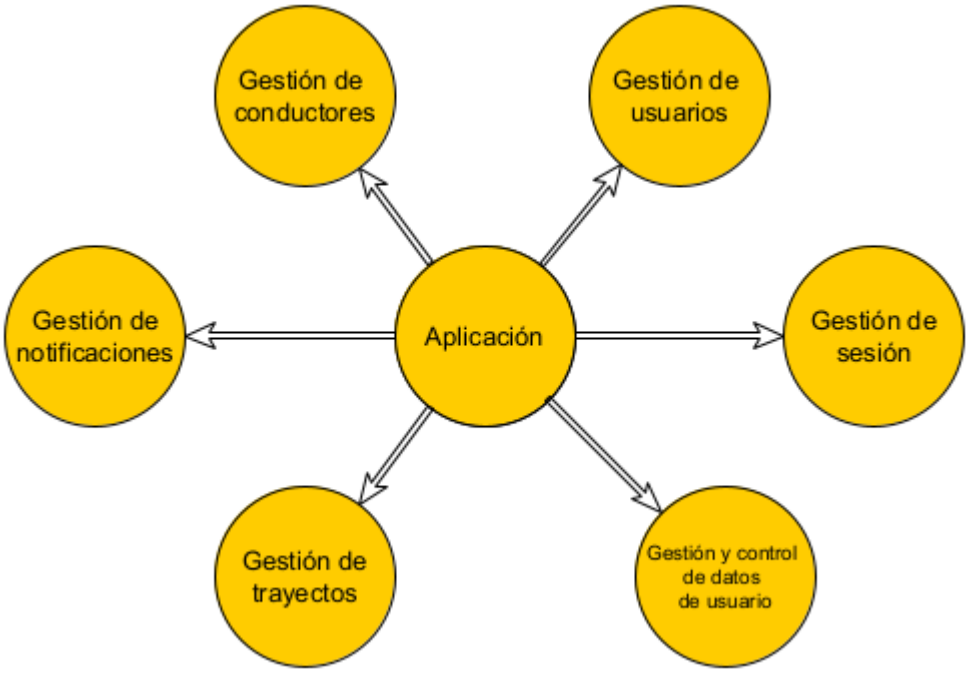
También, se ofrece la opción de visualizar el perfil de usuario, donde se podrá ver la información almacenada e incluso modificarla en caso de existir datos erróneos.

Por último, se desarrollará una pantalla de administración, que solo estará disponibles para usuarios administradores. Estos usuarios, serán dados a los conductores de taxis. La pantalla, básicamente, consistirá en un panel de control, donde el conductor podrá añadir los viajes que vea conveniente, confirmarlos, eliminarlos y modificar la información almacenada sobre ellos, coche, matrícula, etc.

### 4.1.1. Descomposición en subsistemas

Para facilitar la consecución de los objetivos marcados previamente, se ha procedido a descomponer el proyecto en 6 subsistemas. El objetivo de estos subsistemas es disponer de un desarrollo más controlado y eficiente.

*Figura 4.1: Subsistemas de la aplicación*




- **Gestión de sesión:** pequeño subsistema que se encarga de controlar el inicio de sesión a través de varios métodos, mantener la sesión iniciada y cerrarlo en aquel momento que el usuario lo veo preciso.
- **Gestión de usuarios:** este subsistema se encarga de todo los relativo a los usuarios, crear sus perfiles mediante diversos métodos, dar la posibilidad de modificarlos en caso de que la creación no haya sido realizada con éxito e, incluso, de eliminar los perfiles de los mismo que así lo deseen.
- **Gestión y control de datos de usuario:** pequeño subsistema que se encarga de controlar y gestionar como deben estar formados los datos que se introducen al sistema. Es decir, que el teléfono tenga 9 números y ninguna letra, que la fecha de nacimiento y, por tanto, la edad sea superior a 12 años, etc. El objetivo es marcar unas normas para cada uno de los datos.
- **Gestión de trayectos:** subsistema que se encargará de todo lo que guarda alguna relación los mismos. Desde crear un trayecto, guardarlo y cancelarlo hasta seleccionar trayectos en función de los parámetros dados por el usuario.
- **Gestión de notificaciones:** se encargará de controlar él envió de las notificaciones cuando sea preciso. El objetivo es enviar notificaciones con la información más relevante de cara al usuario. Por ejemplo, confirmando que un trayecto se realiza, o avisando de la cancelación de un viaje.

- **Gestión de conductores:** se encargará de gestionar el desarrollo de la parte de control de conductores, los cuales podrán crear trayectos, cancelarlos, ponerse en contacto con los usuarios, etc.

## 4.2. Análisis del sistema

### 4.2.1. Especificación de requisitos

Previamente al inicio del desarrollo de la aplicación, se ha procedido a la redacción y validación de los requisitos de la misma. Estos requisitos, se han dividido en dos clasificaciones diferentes: funcionales y no funcionales.

Además, para la descripción y desarrollo individual de cada requisito, se ha optado por seguir un formato de tabla que consta de los siguientes apartados:

- **Título:** Se dota al requisito de un número y se escribe un título que resuma la idea del requisito.
- **Descripción:** Breve explicación del requisito, que sirve para entender mejor sobre la idea del mismo.
- **Prioridad:** Para el establecimiento de esta característica se ha decido implementar la técnica de prioridades de requisitos conocida como técnica MoSCoW, la cual establece una clasificación de los requisitos en 4 expresiones diferentes. El objetivo es establecer aquellos requisitos que obligatoriamente se han de desarrollar, para comenzar con el desarrollo de los mismo lo antes posible en el periodo de desarrollo del proyecto (Overti, 2016).

*Tabla 4.1: Tipos de prioridades según la metodología MoSCoW*

| Expresión | Definición |
|---|---|
| **Must – Debe tener** | Requisitos fundamentales y obligatorios para satisfacer las necesidades del negocio. Claves en el éxito del proyecto. |
| **Should – Debería tener** | Requisitos que deberían ser cumplidos en la medida de lo posible. El éxito del proyecto no dependerá directamente del cumplimiento de estos. |
| **Could – Podría tener** | Requisitos que son interesantes que cumpla el proyecto, se desarrollarán en caso de disponer de tiempo extra para ello. |
| **Won't – No tendrá** | Requisitos que se ha decidido no implementar de momento, pero que en el futuro es probable que si se implementen. |

- **Subsistema:** Hace referencia a la categoría en la que se encuadra el requisito dentro del desarrollo Software.
- **Estado:** Hay 2 posibles estados al finalizar el desarrollo de la aplicación: Desarrollado o no desarrollado.

#### 4.2.1.1. Especificación de requisitos funcionales.

##### 4.2.1.1.1. Gestión de sesión.

*Tabla 4.2: RF01 (Inicio de sesión por medio de email y contraseña)*

| **RF 01** | **Inicio de sesión por medio de email y contraseña** |
|---|---|
| **Descripción** | Para hacer uso de la aplicación los usuarios deberán iniciar sesión en la misma, podrán usar, para ello, un email y una contraseña. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión de sesión |
| **Estado** | Desarrollado |

*Tabla 4.3: RF02 (Inicio de sesión por medio de cuenta Google)*

| **RF 02** | **Inicio de sesión por medio de cuenta Google** |
|---|---|
| **Descripción** | Para hacer uso de la aplicación los usuarios deberán iniciar sesión en la misma, podrán usar, para ello, una cuenta de Google. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión de sesión |
| **Estado** | Desarrollado |

*Tabla 4.4: RF03 (Mantener sesión iniciada)*

| **RF 03** | **Mantener sesión iniciada** |
|---|---|
| **Descripción** | La aplicación mantendrá la sesión de los usuarios iniciada por defecto, para cerrar sesión el usuario tendrá que usar el botón preparado para ello. |
| **Prioridad** | Debería tener |
| **Subsistema** | Gestión de sesión |
| **Estado** | Desarrollado |

*Tabla 4.5: RF04 (Cerrar sesión de usuario)*

| **RF 04** | **Cerrar sesión de usuario** |
|---|---|
| **Descripción** | La aplicación permitirá a los usuarios cerrar a la sesión en dichos dispositivos. Con ello, si el usuario quiere volver a acceder a la aplicación deberá iniciar sesión de nuevo. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión de sesión |
| **Estado** | Desarrollado |

##### 4.2.1.1.2. Gestión de usuarios

*Tabla 4.6: RF05 (Registro de usuarios)*

| RF 05 | Registro de usuarios |
|---|---|
| **Descripción** | La aplicación permitirá a nuevos usuarios registrarse en la misma. Para ello deberán aportar datos como nombre completo, fecha de nacimiento, correo electrónico, teléfono, etc. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión de usuarios |
| **Estado** | Desarrollado |

*Tabla 4.7: RF06 (Registro de usuarios por medio de cuenta Google)*

| RF 06 | Registro de usuarios por medio de cuenta Google |
|---|---|
| **Descripción** | La aplicación permitirá a nuevos usuarios registrarse en la misma haciendo uso de su cuenta de Google |
| **Prioridad** | Debería tener |
| **Subsistema** | Gestión de usuarios |
| **Estado** | Desarrollado |

*Tabla 4.8: RF07 (Modificación de perfiles de usuario)*

| RF 07 | Modificación de perfiles usuarios |
|---|---|
| **Descripción** | Cualquier dato introducido en el sistema por parte de los usuarios podrá ser modificado. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión de usuarios |
| **Estado** | En proceso |

*Tabla 4.9: RF08 (Eliminación de perfiles de usuario)*

| RF 08 | Eliminación de perfiles de usuario |
|---|---|
| **Descripción** | La aplicación permitirá a los usuarios borrar sus perfiles de la misma. Con ello podrá borrar todos los datos almacenados sobre sí mismo en la aplicación. |
| **Prioridad** | Podría tener |
| **Subsistema** | Gestión de usuarios |
| **Estado** | En proceso |

#### 4.2.1.1.3. Gestión y control de datos de usuario

*Tabla 4.10: RF09 (Control de errores en registro)*

| RF 09 | Control de errores en registro |
|---|---|
| **Descripción** | La aplicación deberá controlar que los datos introducidos para registrarse sean válidos, es decir: que el número de teléfono contenga 9 caracteres numéricos, que el email cumpla con los requisitos requeridos, etc. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión y control de datos de usuario |
| **Estado** | Desarrollado |

*Tabla 4.11: RF10 (Control de contraseña en el registro)*

| RF 10 | Control de contraseña en el registro |
|---|---|
| **Descripción** | La aplicación deberá controlar que la contraseña introducida por el usuario cumpla con unos requisitos mínimo en seguridad: De un número de caracteres alto, que intercale mayúsculas y minúscula y, por último, que introduzca algún carácter numérico o algún símbolo. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión y control de datos de usuario |
| **Estado** | Desarrollado |

*Tabla 4.12: RF11 (Control de errores en el inicio de sesión)*

| RF 11 | Control de errores en el inicio de sesión |
|---|---|
| **Descripción** | La aplicación avisará al usuario cuando al iniciar sesión introduzca de forma errónea el usuario o la contraseña. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión y control de datos de usuario |
| **Estado** | Desarrollado |

#### 4.2.1.1.4. Gestión de trayectos

*Tabla 4.13: RF12 (Creación de trayectos)*

| RF 12 | Creación de trayectos |
|---|---|
| **Descripción** | Los trayectos vendrán creados por la empresa encarga del transporte, de forma que presentará un calendario de trayectos con ubicaciones de origen y destino, viajeros máximos, precio, etc. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión de trayectos |
| **Estado** | En proceso |

*Tabla 4.14: RF13 (Reserva de trayectos)*

| RF 13 | Reserva de trayectos |
|---|---|
| **Descripción** | Para poder reservar viajes el usuario deberá aplicar unos filtros para la búsqueda del mismo: fecha, origen, destino, franja de horas, equipaje, acompañantes, etc. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión de trayectos |
| **Estado** | En proceso |

*Tabla 4.15: RF14 (Cancelación de trayectos)*

| RF 14 | Cancelación de trayectos |
|---|---|
| **Descripción** | Se podrá cancelar un trayecto con un mínimo de 2 horas de antelación, a partir de ese momento el pago será efectivo. |
| **Prioridad** | Debe tener |
| **Subsistema** | Gestión de trayectos |
| **Estado** | En proceso |

*Tabla 4.16: RF15 (Confirmación de trayectos)*

| RF 15 | Confirmación de trayectos |
|---|---|
| **Descripción** | El trayecto se confirmará con 2 horas de antelación, momento en el que el pago se hace efectivo y no se puede cancelar el mismo. |
| **Prioridad** | Debería tener |
| **Subsistema** | Gestión de trayectos |
| **Estado** | En proceso |

*Tabla 4.17: RF16 (Estado de trayectos)*

| RF 16 | Estado de trayectos |
|---|---|
| **Descripción** | El usuario podrá en todo momento ver el estado de los trayectos que tenga reservados en cada momento. Podrá consultar el número de acompañantes, la hora del trayecto, etc. |
| **Prioridad** | Podría tener |
| **Subsistema** | Gestión de trayectos |
| **Estado** | En proceso |

*Tabla 4.18: RF17 (Consulta de trayectos realizados)*

| RF 17 | Consulta de trayectos realizados |
|---|---|
| **Descripción** | El usuario podrá consultar todos los trayectos que ha realizado. Estos viajes serán aquellos que ya se hayan realizado. Podrá consultar toda la información acerca de los mismos: precio, hora, acompañantes, etc. |
| **Prioridad** | Podría tener |
| **Subsistema** | Gestión de trayectos |
| **Estado** | En proceso |

#### 4.2.1.1.5. Gestión de notificaciones

*Tabla 4.19: RF18 (Notificación de confirmación de trayectos)*

| RF 18 | Notificación de confirmación de trayectos |
|---|---|
| **Descripción** | Cuando un trayecto reservado por un usuario se confirme, el sistema deberá enviar una notificación de confirmación de trayecto al usuario. |
| **Prioridad** | Podría tener |
| **Subsistema** | Gestión de notificaciones |
| **Estado** | En proceso |

*Tabla 4.20: RF19 (Notificación de cancelación de trayectos)*

| RF 19 | Notificación de cancelación de trayectos |
|---|---|
| **Descripción** | Cuando un trayecto reservado por un usuario se cancele, el sistema deberá enviar una notificación de cancelación de trayecto al usuario. |
| **Prioridad** | Podría tener |
| **Subsistema** | Gestión de notificaciones |
| **Estado** | En proceso |

#### 4.2.1.1.6. Gestión de conductores

*Tabla 4.21: RF20 (Creación de usuarios con el rol de conductor)*

| RF 20 | Creación de usuarios con el rol de conductor |
|---|---|
| **Descripción** | El sistema podrá crear usuarios con el rol “conductor”. Este rol, hará que el usuario, cuando inicie sesión, observe una pantalla donde podrá realizar diversas acciones. |
| **Prioridad** | Baja |
| **Subsistema** | Gestión de conductores |
| **Estado** | En proceso |

*Tabla 4.22: RF21 (Creación de trayectos)*

| RF 21 | Creación de trayectos |
|---|---|
| **Descripción** | El conductor podrá crear trayectos en cualquier momento. Podrá elegir si estos trayectos se queden de forma periódica en la aplicación o exclusivamente de forma puntual. |
| **Prioridad** | Baja |
| **Subsistema** | Gestión de conductores |
| **Estado** | En proceso |

*Tabla 4.23: RF22 (Confirmación de trayectos)*

| RF 22 | Confirmación de trayectos |
|---|---|
| **Descripción** | El sistema permitirá al conductor, en cualquier momento, confirmar un trayecto y cerrarlo, es decir, prohibir que más usuarios puedan reservar más plazas en el taxi. |
| **Prioridad** | Baja |
| **Subsistema** | Gestión de conductores |
| **Estado** | En proceso |

*Tabla 4.24: RF23 (Cancelación de trayectos)*

| RF 23 | Cancelación de trayectos |
|---|---|
| **Descripción** | El conductor podrá en cualquier momento cancelar un trayecto. Cuando el conductor cancele un trayecto con menos de dos horas de antelación el sistema deberá de proveer a los usuarios de una solución para su viaje. Además, se deberá notificar a los usuarios del cambio. |
| **Prioridad** | Baja |
| **Subsistema** | Gestión de conductores |
| **Estado** | En proceso |

*Tabla 4.25: RF24 (Modificación de perfiles de conductor)*

| RF 24 | Modificación de perfiles de conductor |
|---|---|
| **Descripción** | El sistema permitirá al conductor modificar los datos almacenados sobre él en la aplicación, ya sea para notificar del cambio de un vehículo, matrícula, licencia, etc. |
| **Prioridad** | Baja |
| **Subsistema** | Gestión de conductores |
| **Estado** | En proceso |

*Tabla 4.26: RF25 (Eliminación de perfiles de conductor)*

| RF 25 | Eliminación de perfiles de conductor |
|---|---|
| **Descripción** | El sistema permitirá la eliminación de los perfiles de conductor, ya sea porque esa persona deja de ejercer como conductor de taxi o por otras razones. |
| **Prioridad** | Baja |
| **Subsistema** | Gestión de conductores |
| **Estado** | En proceso |

4.2.1.2. Especificación de requisitos no funcionales.

4.2.1.2.1. Interfaz y usabilidad

*Tabla 4.27: FNR01 (Aplicación móvil adaptable)*

| RNF 01 | Aplicación móvil adaptable |
|---|---|
| **Descripción** | La aplicación deberá ser responsive y será capaz de adaptarse a todo tipo de móviles, en función de su tamaño y resolución. |
| **Prioridad** | Debe tener |
| **Subsistema** | Interfaz y usabilidad |
| **Estado** | En proceso |

*Tabla 4.28: RNF02 (Aplicación móvil accesible)*

| RNF 02 | Aplicación móvil accesible |
|---|---|
| **Descripción** | La aplicación deberá estar preparada para usuarios con problemas relacionados con accesibilidad siguiendo los patrones de diseño establecidos para lograr dicho objetivo. |
| **Prioridad** | Podría tener |
| **Subsistema** | Interfaz y usabilidad |
| **Estado** | En proceso |

*Tabla 4.29: RNF03 (Diseño intuitivo y simple)*

| RNF 03 | Diseño intuitivo y simple |
|---|---|
| **Descripción** | La interfaz de la aplicación deberá tener un diseño simple e intuitivo con el objetivo de que el usuario no tenga problemas al usarla. |
| **Prioridad** | Debe tener |
| **Subsistema** | Interfaz y usabilidad |
| **Estado** | En proceso |

4.2.1.2.2. Rendimiento

*Tabla 4.30: RNF04 (Tiempo de respuesta)*

| RNF 04 | Tiempo de respuesta |
|---|---|
| **Descripción** | El sistema deberá tener, ante cualquier petición, un tiempo de respuesta inferior a los 3 segundos. |
| **Prioridad** | Debe tener |
| **Subsistema** | Rendimiento |
| **Estado** | En proceso |

*Tabla 4.31: RNF05 (Concurrencia de usuarios)*

| RNF 05 | Concurrencia de usuarios |
|---|---|
| **Descripción** | La aplicación mantendrá un correcto funcionamiento con un número de usuarios cercano a los 150. |
| **Prioridad** | Debería tener |
| **Subsistema** | Rendimiento |
| **Estado** | En proceso |

#### 4.2.1.2.3. Seguridad

*Tabla 4.32: RNF06 (Privacidad de sesión)*

| RNF 06 | Privacidad de sesión |
|---|---|
| **Descripción** | El sistema deberá asegurarse de que ningún usuario pueda acceder, modificar o reservar viajes con la cuenta de otro usuario. |
| **Prioridad** | Debe tener |
| **Subsistema** | Seguridad |
| **Estado** | En proceso |

*Tabla 4.33: RNF07 (Privacidad del usuario)*

| RNF 07 | Privacidad del usuario |
|---|---|
| **Descripción** | Los usuarios no tendrán conocimiento de datos sobre los otros usuarios con los que compartirá taxi durante el trayecto. |
| **Prioridad** | Debe tener |
| **Subsistema** | Seguridad |
| **Estado** | En proceso |

*Tabla 4.34: RNF08 (Acceso a la aplicación)*

| RNF 08 | Acceso a la aplicación |
|---|---|
| **Descripción** | Cualquier persona que desee hacer uso de la aplicación deberá iniciar sesión en la misma, para lo cual es necesario que previamente este registrado. |
| **Prioridad** | Debe tener |
| **Subsistema** | Seguridad |
| **Estado** | En proceso |

*Tabla 4.35: RNF09 (Privacidad de los conductores)*

| RNF 09 | Privacidad de los conductores |
|---|---|
| **Descripción** | Los usuarios no tendrán conocimiento de datos sobres los conductores de los taxis que realizarán el trayecto. |
| **Prioridad** | Debe tener |
| **Subsistema** | Seguridad |
| **Estado** | En proceso |

#### 4.2.1.2.4. Compatibilidad

*Tabla 4.36: RNF10 (Compatibilidad con Android y iOS)*

| RNF 10 | Compatibilidad con Android y iOS |
|---|---|
| **Descripción** | La aplicación deberá poder ser instalada y tener una funcionalidad completa y correcta tanto en dispositivos Android con en dispositivos iOS. |
| **Prioridad** | Debe tener |
| **Subsistema** | Compatibilidad |
| **Estado** | En proceso |

#### 4.2.1.2.5. Legal

*Tabla 4.37: RNF11 (Ley de protección de datos)*

| RNF 11 | Ley de protección de datos |
|---|---|
| **Descripción** | La aplicación cumplirá con todos los principios de la ley de protección de datos para aportar una mayor sensación de seguridad al usuario. |
| **Prioridad** | Debe tener |
| **Subsistema** | Legal y seguridad |
| **Estado** | En proceso |

### 4.2.2. Diagrama de casos de uso

Teniendo en cuenta los casos de uso especificados en el punto anterior y las restricciones encontradas, se han definidos 3 actores diferentes a la hora de realizar los casos de uso de la aplicación.

Estos actores son los siguiente: usuario, aplicación o sistema y conductor.

*Figura 4.2: Diagrama de casos de uso del usuario*

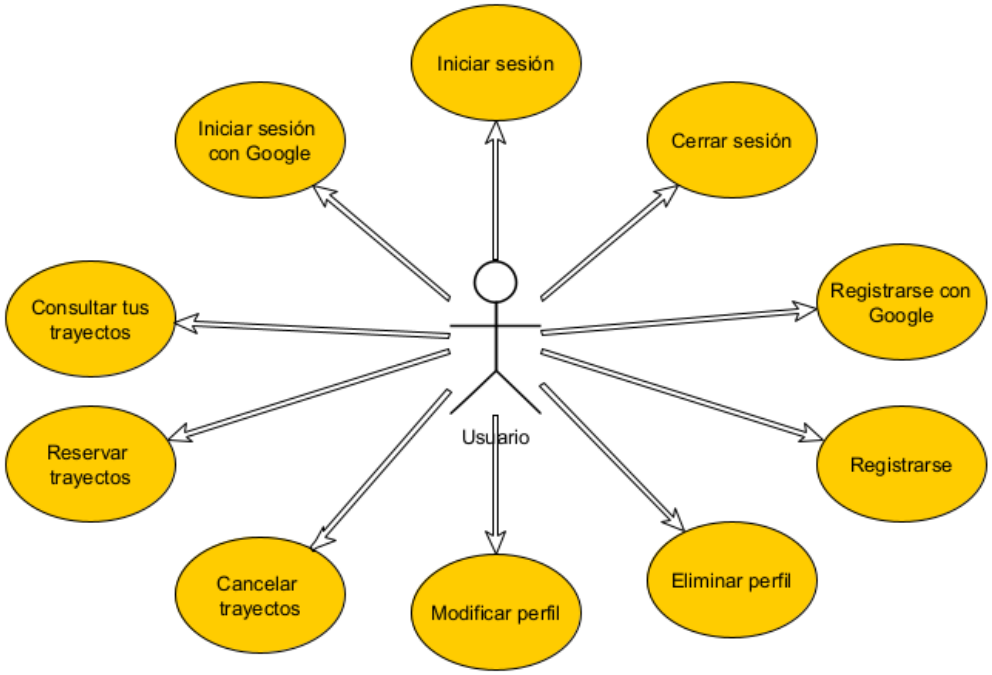
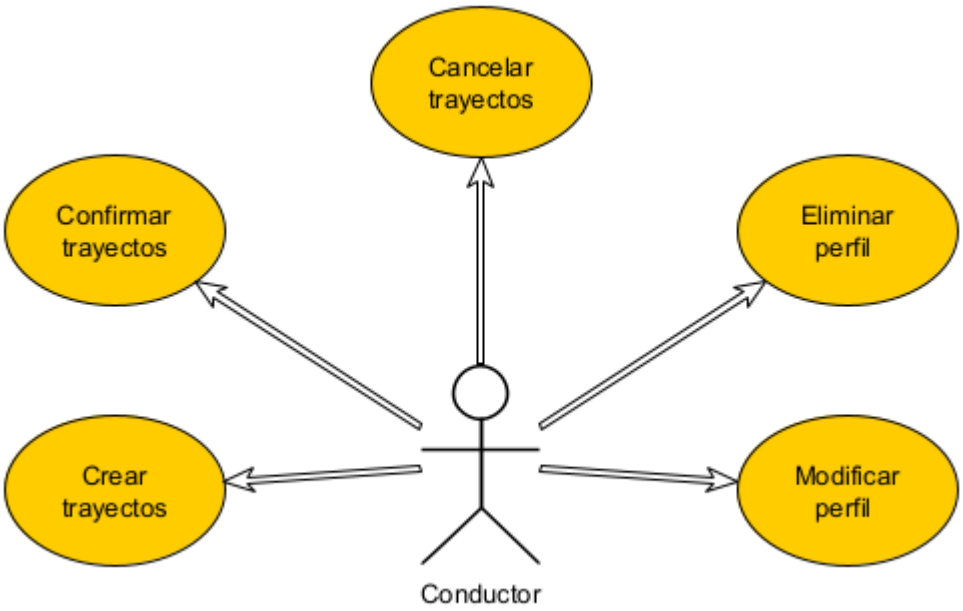


*Figura 4.3: Diagrama de casos de uso de la aplicación o sistema*


*Figura 4.4: Diagrama de casos de uso de los conductores*




### 4.2.3. Especificación de casos de uso

Para describir los diferentes casos de uso se ha desarrollado unas tablas, donde se procede a describir cada uno de estos. La tabla contiene los siguientes campos:

- El número del caso de uso junto con el título o nombre del mismo.
- **Actor:** usuario principal.
- **Descripción:** se detalla de forma breve el objetivo del caso de uso.
- **Precondiciones:** condiciones que deben cumplirse de forma previa al caso.
- **Escenario de éxito:** pasos necesarios a realizar para lograr éxito en el caso de uso.
- **Postcondiciones:** condiciones que se cumplirán tras el caso de uso.

*Tabla 4.38: CU01 (Iniciar sesión)*

| **CU 01** | **Iniciar sesión** |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El usuario inicia sesión en la aplicación como usuario o como conductor |
| **Precondiciones** | Tener una cuenta registra en la aplicación |
| **Escenario de éxito** | 1) El usuario accede a la aplicación<br>2) Introduce su usuario y contraseña<br>3) Se comprueba que es correcto<br>4) Accede a la aplicación |
| **Postcondiciones** | El usuario ha accedido a la aplicación |

*Tabla 4.39: CU02 (Inicio de sesión con Google)*

| **CU 02** | **Iniciar sesión con Google** |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El usuario inicia sesión, haciendo uso de su cuenta de Google en la aplicación como usuario o como conductor |
| **Precondiciones** | Tener una cuenta de Google, que previamente haya sido registrada en la aplicación |
| **Escenario de éxito** | 1) El usuario accede a la aplicación<br>2) Toca el botón de “Inicia sesión con Google”<br>3) Selecciona la cuenta correcta<br>4) Se validan los datos<br>5) Accede a la aplicación |
| **Postcondiciones** | El usuario ha accedido a la aplicación |

*Tabla 4.40: CU03 (Cerrar sesión)*

| CU 03 | Cerrar sesión |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El usuario cierra la sesión en la aplicación, de forma que la próxima vez que desee acceder, deberá introducir los datos de inicio de sesión |
| **Precondiciones** | Haber iniciado sesión en la aplicación |
| **Escenario de éxito** | 1) El usuario toca en el botón de cerrar sesión<br>2) El usuario ha cerrado la sesión |
| **Postcondiciones** | El usuario ha salido y cerrado la sesión en la aplicación |

*Tabla 4.41: CU04 (Registrarse)*

| CU 04 | Registrarse |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El objetivo es registrar a nuevos usuarios en la aplicación. Este método es solo válido para usuarios no para conductores |
| **Precondiciones** | Ninguna |
| **Escenario de éxito** | 1) Se toca en el botón regístrate<br>2) Se rellenan los campos necesarios<br>3) Se comprueban y validan los datos introducidos<br>4) Se registra al usuario<br>5) Se accede a la aplicación |
| **Postcondiciones** | El usuario ha sido registrado en la aplicación |

*Tabla 4.42: CU05 (Registrarse con Google)*

| CU 05 | Registrarse con Google |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El objetivo es registrar a nuevos usuarios en la aplicación a través de una cuenta de Google. Este método es solo válido para usuarios no para conductores |
| **Precondiciones** | Ninguna |
| **Escenario de éxito** | 1) Se toca en el botón “Inicia sesión con Google”, ya que, si no está asociada ninguna cuenta, la crea<br>2) Se validan los datos<br>3) Se registra al usuario<br>4) Se accede a la aplicación |
| **Postcondiciones** | El usuario ha sido registrado en la aplicación |

*Tabla 4.43: CU06 (Modificar perfil)*

| CU 06 | Modificar perfil |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El usuario podrá actualizar los datos almacenados en la aplicación |
| **Precondiciones** | Haber iniciado sesión en la aplicación |
| **Escenario de éxito** | 1) El usuario accede a la pantalla de "Perfil"<br>2) Toca en el icono de editar<br>3) Edita los campos correspondientes<br>4) Toca en el icono de guardar cambios<br>5) Se validan los cambios<br>6) Se guardan en base de datos los cambios |
| **Postcondiciones** | Los datos del usuario han sido modificados |

*Tabla 4.44: CU07 (Eliminar perfil)*

| CU 07 | Eliminar perfil |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El usuario podrá eliminar su perfil en la aplicación |
| **Precondiciones** | Haber iniciado sesión en la aplicación |
| **Escenario de éxito** | 1) En la AppBar de la aplicación toca el botón de borrar perfil<br>2) Confirma que desea borrar el perfil en la aplicación<br>3) Se borra el perfil<br>4) Se saca al login al usuario |
| **Postcondiciones** | El perfil ha sido borrado y el usuario ha perdido la sesión en la aplicación |

*Tabla 4.45: CU08 (Reservar trayectos)*

| CU 08 | Reservar trayectos |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El usuario reserva un taxi para su trayecto |
| **Precondiciones** | Haber iniciado sesión en la aplicación |
| **Escenario de éxito** | 1) El usuario accede a la pantalla "Buscar"<br>2) Introduce los parámetros de búsqueda<br>3) El sistema selecciona las mejores opciones<br>4) El usuario elige la mejor opción<br>5) Se confirma la reserva |
| **Postcondiciones** | El usuario ha reservado un trayecto |

*Tabla 4.46: CU09 (Cancelar trayecto)*

| CU 09 | Cancelar trayecto |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El usuario puede cancelar un trayecto que había reservado |
| **Precondiciones** | El usuario debe tener un trayecto reservado, y que falte al menos 1 hora para que se lleve a cabo |
| **Escenario de éxito** | 1) El usuario accede a "Mis viajes"<br>2) Busca el trayecto deseado<br>3) Pulsa en el botón cancelar trayecto |
| **Postcondiciones** | El usuario ha cancelado el trayecto |

*Tabla 4.47: CU10 (Consultar trayectos)*

| CU 10 | Consultar trayectos |
|---|---|
| **Actor** | Usuario |
| **Descripción** | El usuario puede ver todos los trayectos que ha reservado a lo largo del tiempo, tanto los finalizados como los no realizados aun |
| **Precondiciones** | Haber iniciado sesión en la aplicación |
| **Escenario de éxito** | 1) El usuario accede a "Mis viajes"<br>2) El sistema devuelve todos los viajes asociados a ese usuario |
| **Postcondiciones** | El usuario ha visualizado sus trayectos |

*Tabla 4.48: CU12 (Crear trayectos)*

| CU 12 | Crear trayectos |
|---|---|
| **Actor** | Aplicación o sistema |
| **Descripción** | El sistema puede crear trayectos que pueden ser reservados por los usuarios de la aplicación |
| **Precondiciones** | Ninguna |
| **Escenario de éxito** | 1) Se introducen los parámetros necesarios del trayecto<br>2) La BBDD confirma el ingreso del trayecto |
| **Postcondiciones** | El trayecto ha sido creado con éxito |

*Tabla 4.49: CU13 (Confirmar trayectos)*

| CU 13 | Confirmar trayectos |
|---|---|
| **Actor** | Aplicación o sistema |
| **Descripción** | El sistema confirmará la realización de un trayecto a sus viajeros o usuarios |
| **Precondiciones** | El trayecto debe estar creado y completo |
| **Escenario de éxito** | 1) Una vez completas las plazas para dicho trayecto el sistema procederá a cerrar el mismo<br>2) Se notifica a los usuarios |
| **Postcondiciones** | El trayecto se ha confirmado |

*Tabla 4.50: CU14 (Cancelar trayectos)*

| CU 14 | Cancelar trayectos |
|---|---|
| **Actor** | Aplicación o sistema |
| **Descripción** | El sistema cancelará la realización de un trayecto si no se cumplen las condiciones necesarias para hacerlo |
| **Precondiciones** | El trayecto debe estar creado |
| **Escenario de éxito** | 1) El trayecto no ha sido reservado por ningún usuario o por número de usuario inferior el mínimo para realizar dicho trayecto<br>2) Se cancela el trayecto<br>3) Se notifica a los usuarios correspondiente si procede |
| **Postcondiciones** | El trayecto se ha cancelado |

*Tabla 4.51: CU15 (Notificación de confirmación)*

| CU 15 | Notificación de confirmación |
|---|---|
| **Actor** | Aplicación o sistema |
| **Descripción** | El sistema notificará a los usuarios acerca de la confirmación de realización de un trayecto |
| **Precondiciones** | El trayecto ha sido confirmado |
| **Escenario de éxito** | 1) Se lanzará una notificación a todos los usuarios que hayan reservado el trayecto |
| **Postcondiciones** | Se ha notificado al usuario |

*Tabla 4.52: CU16 (Notificación de cancelación)*

| CU 16 | Notificación de cancelación |
|---|---|
| **Actor** | Aplicación o sistema |
| **Descripción** | El sistema notificará a los usuarios acerca de la cancelación de realización de un trayecto |
| **Precondiciones** | La cancelación ha sido confirmada |
| **Escenario de éxito** | 1) Se lanzará una notificación a todos los usuarios que hayan reservado el trayecto |
| **Postcondiciones** | Se ha notificado al usuario |

*Tabla 4.53: CU17 (Control de errores de registro)*

| CU 17 | Control de errores de registro |
|---|---|
| **Actor** | Aplicación o sistema |
| **Descripción** | El sistema controlará el formato de los datos introducidos por el usuario en el momento del registro |
| **Precondiciones** | Ninguna |
| **Escenario de éxito** | 1) El usuario introduce los datos requeridos<br>2) El sistema evalúa el formato de los mismos<br>3) Si es incorrecto o inválido, avisa del error |
| **Postcondiciones** | Se marca como error el dato introducido |

*Tabla 4.54: CU18 (Control de errores de inicio de sesión)*

| CU 18 | Control de errores de inicio de sesión |
|---|---|
| **Actor** | Aplicación o sistema |
| **Descripción** | El sistema controlará el formato de los datos introducidos por el usuario en el momento del registro |
| **Precondiciones** | Ninguna |
| **Escenario de éxito** | 1) El usuario introduce los datos requeridos<br>2) El sistema evalúa el formato de los mismos<br>3) Si es incorrecto o inválido, avisa del error |
| **Postcondiciones** | Se marca como error el dato introducido |

*Tabla 4.55: CU19 (Creación de conductores)*

| CU 19 | Creación de conductores |
|---|---|
| **Actor** | Aplicación o sistema |
| **Descripción** | El sistema podrá crear perfiles de conductores, estos perfiles estarán asociados a otro perfil de administrador que les permitirá realizar diversas acciones |
| **Precondiciones** | Ninguna |
| **Escenario de éxito** | 1) Se introducen en el sistema los datos del conductor<br>2) El crean tanto el perfil de conductor como el de administrador, con sus diferencias<br>3) El perfil ha sido creado en la BBDD |
| **Postcondiciones** | Se ha creado el perfil de conductor |

*Tabla 4.56: CU20 (Crear trayectos)*

| CU 20 | Crear trayectos |
|---|---|
| **Actor** | Conductor |
| **Descripción** | El conductor podrá crear desde una interfaz básica un trayecto nuevo, con todos sus parámetros |
| **Precondiciones** | Haber iniciado sesión como conductor o administrador en la aplicación |
| **Escenario de éxito** | 1) El usuario accede al panel de creación de trayectos<br>2) Introduce los valores para cado uno de los parámetros<br>3) Se validan dichos valores<br>4) Se confirma la creación del trayecto |
| **Postcondiciones** | El trayecto ha sido creado |

*Tabla 4.57: CU21 (Confirmar trayectos)*

| CU 21 | Confirmar trayectos |
|---|---|
| **Actor** | Conductor |
| **Descripción** | El conductor podrá confirmar cualquier trayecto cuyo conductor sea él mismo |
| **Precondiciones** | Haber iniciado sesión como conductor o administrador en la aplicación |
| **Escenario de éxito** | 1) El conductor accede al panel de "Mis viajes"<br>2) Accede al viaje deseado<br>3) Pulsa en el botón confirmar trayecto<br>4) El trayecto es confirmado<br>5) Se notifica a los usuarios |
| **Postcondiciones** | El trayecto ha sido confirmado |

*Tabla 4.58: CU22 (Cancelar trayectos)*

| CU 22 | Cancelar trayectos |
|---|---|
| **Actor** | Conductor |
| **Descripción** | El conductor podrá cancelar cualquier trayecto cuyo conductor sea él mismo |
| **Precondiciones** | Haber iniciado sesión como conductor o administrador en la aplicación |
| **Escenario de éxito** | 1) El conductor accede al panel de "Mis viajes"<br>2) Accede al viaje deseado<br>3) Pulsa en el botón cancelar trayecto<br>4) El trayecto es cancelado<br>5) Se notifica a los usuarios |
| **Postcondiciones** | El trayecto ha sido cancelado |

*Tabla 4.59: CU23 (Modificar perfil)*

| CU 23 | Modificar perfil |
|---|---|
| **Actor** | Conductor |
| **Descripción** | El conductor podrá modificar su perfil. Modificando datos erróneos introducidos o datos que hayan podido cambiar como su coche, matrícula, etc. |
| **Precondiciones** | Haber iniciado sesión como conductor o administrador en la aplicación |
| **Escenario de éxito** | 1) El conductor accede al panel "Perfil"<br>2) Pulsa en el icono de editar datos<br>3) Realiza los cambios necesarios<br>4) Pulsa en el botón de guardar cambios<br>5) Se validan los cambios<br>6) Se confirma la modificación del perfil |
| **Postcondiciones** | El perfil ha sido modificado |

*Tabla 4.60: CU24 (Eliminar perfil)*

| CU 24 | Eliminar perfil |
|---|---|
| **Actor** | Conductor |
| **Descripción** | El conductor podrá eliminar su perfil de la aplicación en cualquier momento. |
| **Precondiciones** | Haber iniciado sesión como conductor o administrador en la aplicación |
| **Escenario de éxito** | 1) El conductor pulsa en el botón de la AppBar de "Borrar perfil"<br>2) Confirma que quiere borrar el perfil<br>3) Se borra su perfil de administrado y de conductor<br>4) Se borran los trayectos asociados a dicho conductor<br>5) Se cancelan los viajes asociados<br>6) Se notifica a los usuarios<br>7) Se confirma el borrado del perfil<br>8) Se cierra la sesión del conductor |
| **Postcondiciones** | El perfil ha sido eliminado y la sesión del conductor ha sido cerrada |

## 4.3. Diseño del sistema

En esta sección, se pretende explicar todo lo relacionado con las decisiones tomadas en el diseño de aplicación y que llevan a la consecución los requisitos previamente establecidos, así como de los objetivos explicados anteriormente.

### 4.3.1. Arquitectura de software

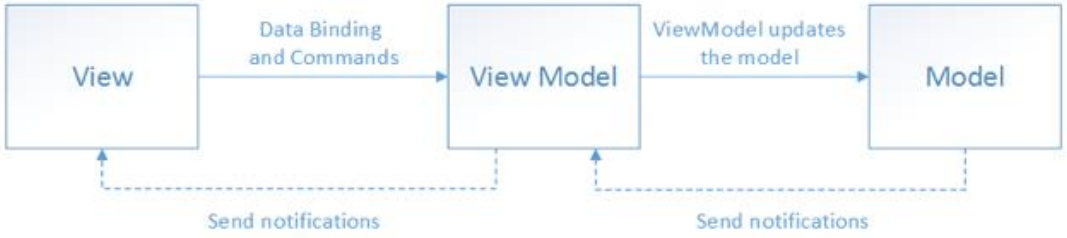
La arquitectura de software utilizada para realizar el proyecto es la misma que desde Proconsi, empresa en la que realicé las prácticas, se me instó a usar. Esta arquitectura es MVVM, que es una de las arquitecturas más recomendadas y, además, Flutter da soporte a la misma. También hay que destacar, que se ha probado que haciendo uso de esta arquitectura se consigue una buena división de tareas en las últimas aplicaciones desarrolladas para Android (Chamorro, 2020).

El significado de MVVM es Modelo-Vista-VistaModelo, que procede de Model-View-ViewModel en inglés. A continuación, procederé a explicar en detalle cada uno de los 3 componentes:

- **Model o modelo:** representa la capa de datos. Todos los datos de la aplicación se van a exponer a los VM (ViewModels) a través de Repositories. Los Repositories a su vez podrán acceder a DAOs, Storage, Services... cualquier origen de datos. En definitiva, no puede haber datos específicos de la vista en ninguna de las capas del Model, las cuales contienen los datos, pero las acciones o servicios que los manipulan.

- **View o vista:** representa la interfaz de usuario, es decir, las pantallas o Widgets. Todas las vistas no podrán contener ningún tipo de lógica de datos, todos los datos a los que acceden serán exclusivamente a través de los VM.
- **ViewModel o VistaModelo:** representa al intermediario entre los datos, el Model, y la interfaz, la vista. Los ViewModels van a tener las variables necesarias que representan los modelos de datos de la Vista (pantalla/widgets), a su vez manejaran toda la lógica necesaria para transformar y obtener los datos en la forma concreta que necesite la Vista a la que pertenezca el VM. El VM solo accederá a los datos a través de Repositories. También son los encargados de manejar los distintos estados de la pantalla. La relación de VM-Pantalla normalmente será de 1:1, una pantalla, un VM, puede darse casos en los que una pantalla tenga por ejemplo pestañas y dependiendo de si lo que se está mostrando en las pantallas tiene identidad propia o no, o una complejidad de datos a mostrar podría haber 1 ViewModel para la pantalla y 1 ViewModel por pestaña.

*Figura 4.5: Patrón MVVM*




### 4.3.2. Arquitectura del sistema

La arquitectura utilizada en el sistema es muy sencilla, se basa en 2 secciones principales y claramente divididas.

Por un lado, el FrontEnd de la aplicación, que es la parte del sistema que se encarga de interactuar directamente con el usuario. Recuerdo que esta parte se ha desarrollado haciendo uso de Flutter y Dart.

Por otro lado, el BackEnd que es la parte del sistema que se encarga de almacenar y proporcionar datos al usuario por medio del FrontEnd. El BackEnd ha sido desarrollado haciendo uso de NodeJS. Estos datos que proporciona el BackEnd, los obtiene de la base de datos, que ha sido desarrollada en MongoDB.

La interacción entre ambas partes es muy sencilla, el BackEnd se encarga de proporcionar los datos extraídos de la basa de datos a través de archivos JSON, que el FrontEnd se encarga de procesar y mostrar de la forma adecuada al usuario, a través de las diferentes pantallas o Widgets.

Por último, cabe destacar que el FrontEnd de nuestra aplicación está conectado con 2 servidores de BackEnd. Por un lado, el ya mencionado en el texto anterior. Por otro lado, nuestra aplicación tiene todo el sistema de registro y login de usuario y contraseña gestionado a través del servicio de autenticación de Firebase. Esto se ha realizado de este modo, para evitar posibles pérdidas de usuario y/o contraseña por un mal almacenaje de los mismos en nuestra propia base de datos, así como evitar el robo de las cuentas asociadas a dichos datos. Esto facilita mucho la comprobación de registro e inicio de sesión, ya que Firebase está muy optimizado para su uso junto con Flutter y Dart, algo muy conveniente y que nos facilita el desarrollo.

### 4.3.3. Diseño de la base de datos

La base de datos es uno de los puntos más importantes de la aplicación desarrollada. Se encarga de almacenar y proporcionar, cuando es necesario, toda la información solicitada.

Para el correcto funcionamiento de la aplicación, se ha optado por el uso de una base de datos NoSQL, en concreto de una base de datos proporcionada por el sistema de bases de datos de MongoDB

#### 4.3.3.1. MongoDB

Es un sistema de bases de datos NoSQL, es decir, no relacional, orientado a objetos y de código abierto. El sistema de funcionamiento es muy sencillo, ya que MongoDB proporciona un sistema de almacenaje datos en estructuras de datos de tipo BSON, muy similares al JSON, lo que facilita usarlo en el BackEnd de una aplicación (Robledano, 2019).

MongoDB presenta un gran número de ventajas como su gran velocidad o su potente rendimiento. Pero las características principales son las siguientes:

- **Consultas ad hoc:** MongoDB permite realizar todo tipo de consultas, ya sea por rangos, campos o expresiones regulares.

- **Indexación:** el concepto de índices es muy similar al usado en bases de datos SQL, por tanto, fácilmente comprensible.
- **Replicación:** soporta el tipo de replicación primario-secundario. Esto permite realizar consultas con el primario mientras que el secundario actúa como réplica de datos en solo lectura a modo copia de seguridad
- **Balanceo de carga:** permite ejecutarse de manera simultánea en múltiples servidores.
- **Almacenamiento de archivos:** permite manipular archivos y contenido gracias a GridFS incluido en la distribución oficial.
- **Ejecución de JavaScript del lado del servidor:** permite consultas escritas en este lenguaje, las cuales son enviadas directamente a la BBDD.

#### 4.3.3.2. Como funciona MongoDB

El funcionamiento de MongoDB es muy sencillo. A lo ya explicado anteriormente, hay que añadir lo siguiente, los términos empleados en MongoDB:

- **Colecciones:** equivalen o se corresponde con las tablas en las bases de datos relacionales.
- **Documentos:** la unidad fundamental que se encarga de almacenar la información. Equivalente a una fila de tabla de las bases de datos relacionales. Su estructura de almacenaje es en BSON, muy similar a JSON.
- **Drivers:** conjunto de librerías empleadas para comunicar la base de datos con la aplicación o servidor correspondiente.

#### 4.3.3.3. Modelado de la base de datos.

A continuación, se mostrará un diagrama que representa la estructura general que sigue la base de datos, es decir, las colecciones con sus respectivos documentos. Cabe recordar, que MongoDB es un sistema de base de datos NoSQL, por tanto, es un sistema no relacional. Esto implica que no se podrá observar una relación visual entre las diferentes colecciones, es decir, entre las diferentes tablas si habláramos de en SQL.

*Figura 4.6: Esquema de la base de datos*


4.3.3.4. Colecciones de la base de datos

A continuación, voy a describir cada una de las 3 colecciones, más conocidas como tablas en bases de datos relacionales, utilizadas en la base de datos de la aplicación.

Para describirlas hablaré de cada uno de los documentos, con una pequeña descripción que indique que información va a almacenar dicho documento, junto con el tipo del mismo, es decir, si es de tipo Int, String, etc.

La primera colección que vamos a comentar es la de **Users**. Esta colección tiene como objetivo almacenar información acerca de los usuarios registrados en la aplicación. Esta información servirá para posteriormente mostrarla en diversas pantallas, así como ofrecer al usuario la posibilidad de modificarla en caso de haber introducido datos erróneos.

*Tabla 4.61: Colección de usuarios*

| Documento | Descripción | Tipo |
|---|---|---|
| **Id** | Identificador único de cada usuario | String |
| **Name** | Nombre del usuario | String |
| **Surname** | Apellidos del usuario | String |
| **Email** | Email del usuario, se podrán mandar notificaciones | String |
| **Birthday** | Fecha de nacimiento del usuario. Debe ser mayor de 12 años para hacer uso de la app | Date |
| **Phone number** | Número de teléfono del usuario | Number |
| **Rol** | Rol del usuario, puede ser: usuario o administrador | String |
| **Token** | Identificador que se corresponde con el otorgado por Firebase, para establecer una relación entre las dos bases de datos | String |
| **Trips** | Array que contiene los viajes del usuario, tanto los realizados como los reservados. | Schema.Trip |

La siguiente colección que vamos a comentar es la de **Trips**, es decir viajes o trayectos. Esta colección tiene como objetivo almacenar la información sobre los viajes a realizar para mostrársela al usuario, permitiendo al mismo reservar uno de estos. Cuando el usuario reserva un trayecto, este asociará a dicho usuario con dicho trayecto.

*Tabla 4.62: Colección de viajes*

| Documento | Descripción | Tipo |
|---|---|---|
| **Id** | Identificador único de cada viaje | String |
| **Early Start** | Hora y fecha inicial del viaje | Date |
| **Late Start** | Hora y fecha tope para la salida del viaje | Date |
| **Start Place** | Ubicación de origen del trayecto | String |
| **End Place** | Ubicación de destino del trayecto | String |
| **Duration** | Duración aproximada del trayecto | Number |

| | | |
|---|---|---|
| **Driver** | Referencia al conductor que se encarga del viaje | Schema.Driver |
| **Users** | Referencia a los usuarios que reservan el viaje | Schema.User |

Por último, la última colección que es necesario describir es la colección de **Drivers**. El objetivo de esta colección es almacenar todos los datos necesarios sobre los conductores. No todos los datos que se almacenan sobre estos conductores se muestran a los usuarios, es más, solo se mostrarán datos como el modelo del coche o la matrícula para facilitar al usuario encontrar el taxi que realizará su transporte.

*Tabla 4.63: Colección de conductores*

| Documento | Descripción | Tipo |
|---|---|---|
| **Id** | Identificador único de cada conductor | String |
| **Name** | Nombre del conductor | String |
| **Surname** | Apellido del conductor | String |
| **License** | Licencia de taxi del conductor, es otorgada por la autoridad competente | String |
| **Car brand** | Marca del coche usado como taxi, sirve para ser identificado por el usuario | Date |
| **Car model** | Modelo del coche usado como taxi, sirve para ser identificado por el usuario | String |
| **Phone number** | Número de teléfono del conductor, puede ser usado para comunicaciones especiales | Number |
| **Max passangers** | Número máximo de viajeros que pueden ir en el taxi del conductor | Number |
| **Max bagggage** | Número de maletas (de tamaño promedio) que el taxi puede transportar | Number |

### 4.3.4. Diseño del logo

En esta sección se procede a describir cual ha sido la idea del diseño que se ha seguido para elaborar el logo de la aplicación. A continuación, se mostrará el logo diseñado.

*Figura 4.7: Logo de BlablaTaxi*

Lo primero que se quiso reflejar en el diseño del logo, era que tuviera alguna similitud con el mundo de los taxis, y para ello, la forma principal del logo sigue la estructura del mítico letrero situado en el techo de la mayoría de los taxis de nuestro país que indica si está ocupado o libre.

*Figura 4.8: Letrero de indicación del estado del taxi*


Una vez definida la forma principal, se pasó a diseñar lo que se incluiría dentro del mismo. Tras varias ideas erróneas, se procedió a diseñar algo sencillo que de alguna forma indicará el nombre de la aplicación, BlablaTaxi. Por ello se situó, en primer lugar, el emoticono, que hace referencia al Blabla del nombre, y, por último, una T que hace

referencia al Taxi del resto del nombre. Con esto se logra que de un simple vistazo el usuario sepa que está usando BlablaTaxi.

Por último, había que seleccionar los colores que iban a formar parte del logo y que lugares. Después de realizar consultas a algunos compañeros se concluyó que todos ellos asociaban los taxis con el color amarillo, y alguno de ellos con el color negro. Por tanto, se seleccionó estos dos colores para completar el logo. Una vez definido todo esto, el logo desarrollado es el visto en la imagen anterior.

## 4.4. Implementación

Este apartado está dedicado al proceso de implementación de la aplicación. El objetico es especificar las herramientas utilizadas y los problemas observados durante la etapa de desarrollo.

### 4.4.1. Tecnologías empleadas

Durante la implementación de la aplicación BlablaTaxi, se han usado los siguientes lenguajes y bibliotecas:

- FrontEnd: se ha usado el framework Flutter en su versión 2.10.3, junto con el lenguaje de programación Dart en su versión 2.16.1 y las DevTools en su versión 2.9.2.
- BackEnd: se ha usado NodeJS en su versión 16.14.2 junto con Firebase. Además, el servidor de NodeJS se encuentra almacenado en una máquina virtual de Ubuntu en su versión 20.04.4 LTS gestionado a través de AWS.
- Base de datos: se ha usado el sistema de bases de datos NoSQL de MongoDB en su versión 4.4.13.

### 4.4.2. Limitaciones existentes

En esta sección se procede a redactar las diferentes limitaciones que pueden surgir del desarrollo de la aplicación.

Al usar Flutter como framework, que recuerdo que es una tecnología que permite el desarrollo híbrido de aplicaciones móviles, la aplicación estará disponible para los sistemas operativos Android y iOS, es decir, para la mayoría de los sistemas operativos presentes en los dispositivos móviles, smartphones y tablets, de la actualidad.

Para poder hacer uso de la aplicación los usuarios deberán ser mayores de 12 años, en caso contrario de le deberá limitar el uso de la misma.

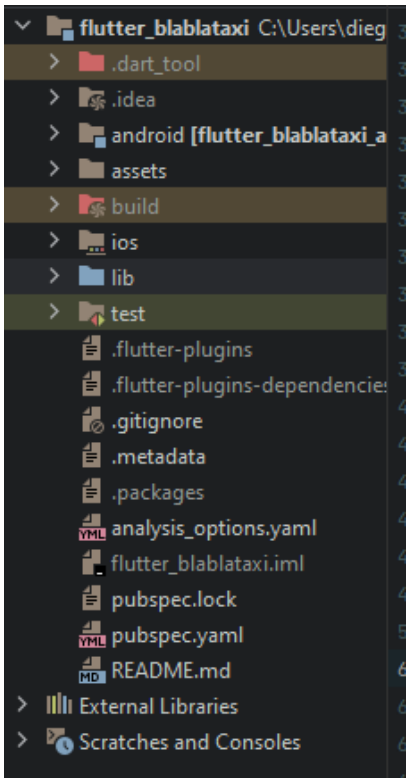
Se tratar de asegurar una concurrencia para la reserva de trayectos, para evitar que varios usuarios a la vez reserven un trayecto provocando un colapso en la aplicación, para lo cual se deberán usar unos sockets que aseguren la concurrencia.

### 4.4.3. Estructura general de la aplicación

En este apartado se procede a explicar la estructura general usada durante el desarrollo de la aplicación.

En primer lugar, vemos la estructura principal. En este caso cuenta con las carpetas correspondientes de Android y iOS donde se especifican las versiones de determinadas librerías que se deben usar para que el proyecto funcione correctamente. Por otro lado, la carpeta "*lib*" que contiene todo lo básico del proyecto y la carpeta "*test*" donde se encuentran los diferentes casos de prueba que se deseen desarrollar. Por último, contamos con unos archivos autogenerados, como son. flutter-plugins y. flutter-plugins-dependencies que no se modifican por parte del desarrollador, y para finalizar el archivo pubspec.yaml que contiene todos los plugins a implementar con sus correspondientes versiones, así como las dependencias necesarias.

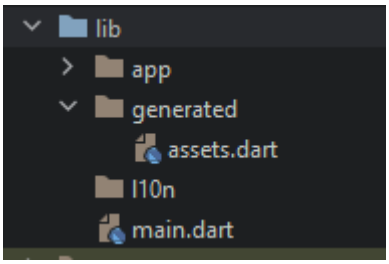
*Figura 4.9: Estructura general del proyecto*




La carpeta "*lib*", donde los desarrolladores proceden a redactar el código, está formada por la carpeta "*app*", la carpeta "*l10n*", que contendrá información sobre

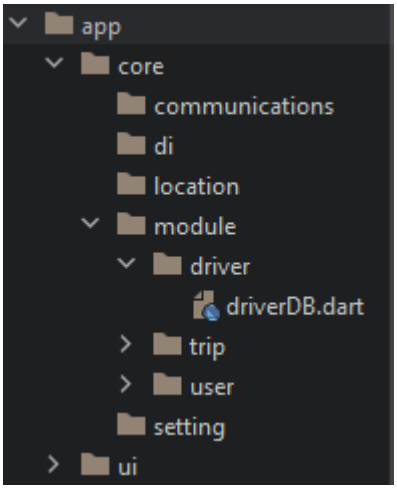
dependencia inyectadas posteriormente, la carpeta "*generated*" que contiene un archivo generado en tiempo de ejecución y por último el *Main.dart*, que es el Main de la aplicación

*Figura 4.10: Estructura carpeta lib*




Dentro del directorio "*app*" encontramos dos subdirectorios, uno que es "*ui*" que se explicará posteriormente y el subdirectorio "*core*". Este último está formado por los directorios de la imagen, donde destacamos el directorio "*module*" que contiene un directorio por cada una de las colecciones de la base de datos y su correspondiente modelo. En el directorio "*setting*" estará disponible información general que se utilizará en diversos lugares de la aplicación.

*Figura 4.11: Estructura de app y subdirectorio core*

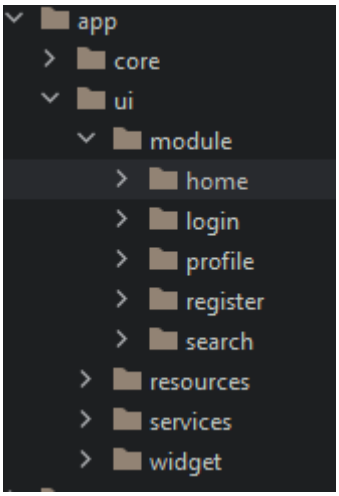



A continuación, en el directorio "*ui*" encontramos los siguientes subdirectorios: "*module*"," resources"," services" y" widget".

- *Module:* en este directorio encontraremos una carpeta por pantalla que se diseña, en esa carpeta además encontraremos el archivo correspondiente con la vista.
- *Resources:* encontraremos archivos que contienen variables comunes a todas las vistas, como son los iconos, los colores o las dimensiones.

- *Services:* se encuentra archivos que dan servicio a las vistas, los ViewModels, como el de Firebase o el de usuario para obtener los datos de los usuarios que en cada momento sea preciso.
- *Widget:* que contiene widgets que van a ser usados en más de una pantalla de la aplicación.

*Figura 4.12: Estructura del directorio ui*

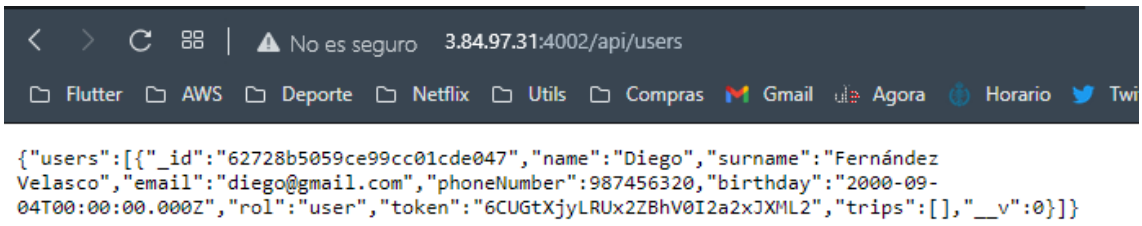



### 4.4.4. Base de datos

La comunicación del FrontEnd con la base de datos es muy sencilla. Hay que recordar que la base de datos esta creada con MongoDB, que es un sistema NoSQL.

El funcionamiento es el siguiente, el servidor BackEnd, que es creado con NodeJS, crea unas API's, que se encuentran almacenadas en un servidor de Ubuntu de AWS. A continuación, se procede a mostrar un ejemplo de API creada por el BackEnd donde se muestran los usuarios que en ese momento se encuentran registrados en la aplicación.

*Figura 4.13: Ejemplo de API - mostrar usuarios*




Por otro lado, en el FrontEnd es muy sencillo pedir los datos, simplemente procede a realizar una petición de tipo GET, en este caso, ya que se pueden realizar otro tipo de peticiones como POST (para el registro) o UPDATE (para actualizar campos). Una vez

realizada esa petición se decodifica la respuesta y se mapea el JSON recibido a una variable del tipo correspondiente, en este caso el modelo de usuario.

*Figura 4.14: Ejemplo de petición para obtener un usuario de la API*

```
Future<UserDB?> getUser(String? token) async {
  if(token!=null){
    var url = Uri.parse('http://3.84.97.31:4002/api/users/'+token);
    var response = await http.get(url, headers: {
      "Content-Type": "application/json",
      "Authorization": "Bearer $token"
    });
    if (response.statusCode == 200) {
      return UserDB.fromJson(jsonDecode(response.body)['user'][0]);
    } else {
      throw Exception('Failed to get user from database');
    }
  }
  return null;
}
```

### 4.4.5. Implementación del FrontEnd

El FrontEnd ha sido creado con el framework Flutter en colaboración con el lenguaje de programación Dart.

Para comprobar que la vista desarrollada se de forma adecuada usando distintas resoluciones en distintos dispositivos, el proyecto se ha ido probando, usando varios emuladores que cuentan con diferentes resoluciones.

El FrontEnd se encarga de realizar las peticiones concretas para obtener los datos necesarios de las API's para poder mostrarlos en las respectivas pantallas cuando sea necesario.

Además, en el caso de que las acciones realizadas por el usuario que conlleven registrar datos en la base de datos, como, por ejemplo, registrar usuarios, modificar datos del usuario o registrar viajes, se realizarán las peticiones adecuadas a la base de datos para agregar o modificar los datos adecuados.

### 4.4.6. Implementación del BackEnd

El BackEnd ha sido creado con el entorno de programación NodeJS, y para su edición se ha usado Visual Studio Code.

Con el entorno citado y haciendo uso de express, Morgan y cors se ha creado una infraestructura que permite crear API's que serán consultadas por el FrontEnd para adquirir los datos necesarios en las pantallas correspondientes.

La conexión con la base de datos es relativamente fácil de lograr, dado que tanto la base de datos MongoDB como la aplicación que actúa como servidor de NodeJS, están almacenadas en el servidor de Ubuntu localizado en AWS, la conexión entre el servidor y la base de datos es una conexión local, lo que facilita las cosas.

*Figura 4.15: Conexión del servidor con la base de datos*

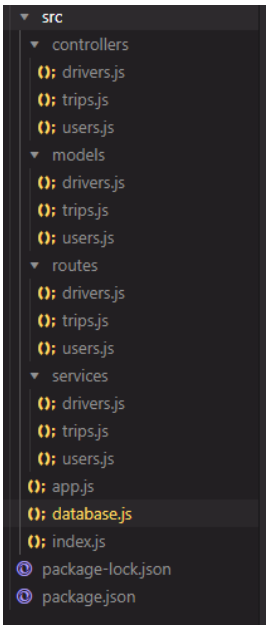
```
const mongoose = require('mongoose');
const dbConection = async() => {
    try{
        console.log('Conectando con la base de datos...');
        await mongoose.connect('mongodb://localhost/BlablaTaxiDB', {
            useNewUrlParser: true,
            useUnifiedTopology: true
        });
        console.log('Conectando...');
    } catch (err) {
        throw new Error(err);
    }
}

module.exports = {
    dbConection
};
```

La creación ha resultado muy sencilla, se ha creado dentro del directorio "*src*" 4 subdirectorios que contienen para cada colección de la base de datos el archivo correspondiente:

- *Models:* están los modelos de las colecciones de la base de datos, con sus respectivos documentos, más conocidos como atributos.
- *Controllers:* se encarga de llamar al servicio y esperar la respuesta recibida, convirtiéndola a JSON.
- *Routes:* se encarga de establecer las rutas que hay que utilizar para la ejecución de las consultas de la base de datos.
- *Services:* es el lugar donde aparecen las consultas a la base de datos.

*Figura 4.16: Estructura del servidor NodeJS*




### 4.4.7. Publicación en las tiendas.

Una vez completado el desarrollo de la aplicación se debería proceder con su publicación y despliegue en las tiendas tanto de Android, Play Store, como en la de iOS, App Store.

Aunque el procedimiento para publicar la aplicación es similar tiene una serie de diferencias que conviene comentar.

#### 4.4.7.1. Publicación en Play Store

En primer lugar, se debe crear una cuenta de desarrollador en Google Play (Yeeply, 2020). El procedimiento es muy similar a la creación de una cuenta de Google normal. Al ser una cuenta de desarrollador se debe pagar una cuota de 25 dólares. En caso, de que las aplicaciones que vayas a publicar vayan a incluir pagos, puedes incluir un perfil de pagos que te proporciona una consola que permite controlar los pagos realizados.

Posteriormente, debes completar una ficha con información sobre la aplicación y que sirve para que Google muestre tu app en función de las búsquedas de los usuarios. Entre otros datos deberás proveer en dicha ficha: nombre, recursos gráficos, idiomas o datos de contacto.

A continuación, debes subir la APK, clasificar su contenido, algo que ayuda a que la aplicación obtenga un mayor número de descargas, y, establecer su precio y distribución.

Por último, se envía la aplicación a revisión y se espera a que sea aprobada y, finalmente, subida a la Play Store estando disponible para su descarga y uso.

4.4.7.2. Publicación en App Store

Los pasos para seguir guardan grandes similitudes.

Inicialmente, deberemos crear una cuenta de desarrollador, la cual tiene un coste de 99 dólares anuales, bastante más caro que en el caso de Google (Antevenic, 2016).

Posteriormente, debes dar de alta la cuenta de desarrollador en Xcode, para lo cual se deberán completar ciertos campos como el objetivo de nuestra app, algo muy similar al procedimiento en Google.

A continuación, se empaqueta y se sube la aplicación para que sea sometida a la revisión por parte del personal de Apple. Para esta revisión hay que enviar un formulario con información adicional sobre la aplicación (Devsdna, 2022).

Por último, hay que esperar al resultado de la revisión de Apple, que suele durar en torno a 2 o 3 días. En caso de que la revisión resulte satisfactoria se procede a publicar la aplicación y, en caso contrario, se te notifica de los fallos cometidos y, en algunas ocasiones, te aportan alguna solución para los mismos.

4.4.8. Posibles mejoras

En esta sección se procede a redactar una serie de requisitos o de funcionalidades que podrían ser implementadas en la aplicación si el cliente deseara o si el equipo de desarrollo dispusiera de más tiempo para la entrega del producto.

- Confirmación de la cuenta de BlablaTaxi por medio del email, es decir, aceptación de un correo de confirmación.
- Si el usuario no accede a la aplicación en un periodo de tiempo preestablecido, por ejemplo 3 meses, enviar un correo a dicho usuario para que vuelva acceder a la misma aplicación.
- Almacenar una foto de perfil del usuario para mejorar la identificación de los clientes por parte del taxista, y de esta forma, evitar fraudes.

## 4.5. Pruebas

En este apartado se describen las diferentes y variadas pruebas que se le han realizado al sistema. A nivel general, las pruebas han sido realizadas la final de cada una

de las iteraciones, pero también se han realizado pruebas Alpha sobre el funcionamiento de la aplicación, así como prueba Beta, que han sido llevadas a cabo por parte del cliente.

### 4.5.1. Pruebas unitarias

Es un tipo de pruebas destinadas a verificar el comportamiento y función de una unidad de código. Estas pruebas se dividen en dos grupos principales: pruebas de caja blanca y pruebas de caja negra.

#### 4.5.1.1. Pruebas de caja blanca

Este tipo de pruebas se realizan conociendo la estructura del módulo que se evalúa, conociendo su funcionamiento.

Para la evaluación y verificación de módulos usando este tipo de prueba el Software Engineering Institute ha creado una tabla que indica el riesgo potencial según el valor de complejidad ciclomática que posea dicho módulo.

| Complejidad ciclomática | Riesgo |
|---|---|
| **1-10** | Programa simple → poco riesgo |
| **11-20** | Programa poco complejo → riesgo medio |
| **21-50** | Programa complejo→ riesgo alto |
| **> 50** | Programa no testeable → riesgo muy alto |

Para comprobar el funcionamiento correcto del sistema se va a ejecutar estas pruebas sobre el módulo encargado de registrar usuarios en la base de datos.

```
Future<UserDB> addUser(UserDB user) async {
  var url = Uri.parse('http://3.84.97.31:4002/api/users/register');
  var json = jsonEncode(mapUser(user));
  var response = await http.post(url, body: json, headers: {
    "Content-Type": "application/json"
  });
  if (response.statusCode == 200) {
    return UserDB.fromJson(jsonDecode(response.body)['user']);
  } else {
    throw Exception('Failed to add user in database');
  }
}
```

El diagrama de flujo muestra 8 nodos y 8 aristas. A continuación, se obtiene la complejidad ciclomática mediante la siguiente formula: **V(G) = nodos – aristas + 2**, por tanto → **V(G) = 8 – 8 +2 = 2.**

Como la complejidad ciclomática es 2, quiere decir que para este módulo existen 2 formas distintas de recorrer todo el código. Acudiendo a la tabla, deducimos que es un programa simple, por tanto, con muy poco riesgo.

| Número | Recorrido | Entrada | Salida | Resultado |
|---|---|---|---|---|
| **1** | 1-2-3-4-5-6-8 | Usuario con todos los campos correcto | Se almacena usuario en base de datos | Ok |
| **2** | 1-2-3-4-5-7-8 | Usuario con algún campo repetido respecto a la base de datos como es el DNI, que tiene que ser único | Error → 'Failed to add user in database' | Ok |

Revisando la tabla anterior observamos que el resultado de las pruebas unitarias es favorable para todos los caminos independientes.

4.5.1.2. Pruebas de caja negra

Este tipo de pruebas son distintas de las anteriores, dado que únicamente se centran en la entrada y salida del código, sin tener en cuenta la estructura interna del módulo a evaluar.

Para simplificar y sintetizar este tipo de pruebas, con el objetivo de evitar el manejo de grandes cantidades de información provenientes de estas pruebas, se han realizado y detallado solo las pruebas para cada uno de los casos de uso definidos anteriormente.

| Caso de uso | Nombre | Resultado esperado | Válido |
|---|---|---|---|
| **CU 01** | Iniciar sesión | El usuario accede correctamente a la app | Sí |
| **CU 02** | Inicio de sesión por medio de cuenta de Google | El usuario accede correctamente a la app | Sí |
| **CU 03** | Cerrar sesión | Se cierra la sesión correctamente | Sí |

| | | | |
|---|---|---|---|
| **CU 04** | Registrarse | Se registra correctamente | Sí |
| **CU 05** | Registrarse con cuenta de Google | Se registra correctamente | Sí |
| **CU 06** | Modificar perfil | Se modifican los datos correctamente | Sí |
| **CU 07** | Eliminar perfil | Se eliminar el perfil de la base de datos | Sí |
| **CU 08** | Reservar trayectos | Se reserva un trayecto correctamente | Sí |
| **CU 09** | Cancelar trayectos | Se cancela un trayecto correctamente | Sí |
| **CU 10** | Consultar trayectos | Se consultan las distintas opciones para realizar trayectos | Sí |
| **CU 12** | Crear trayectos | Se crean nuevos trayectos correctamente | Sí |
| **CU 13** | Confirman trayectos | Se confirman trayectos correctamente | Sí |
| **CU 14** | Cancelar trayectos | Se cancelan trayectos correctamente | Sí |
| **CU 15** | Notificación de confirmación | Se notifica correctamente | Sí |
| **CU 16** | Notificación de cancelación | Se notifica correctamente | Sí |
| **CU 17** | Control de errores registro | Se evitan datos incorrectos | Sí |
| **CU 18** | Control de errores inicio de sesión | Se evitan datos incorrectos | Sí |
| **CU 19** | Creación de conductores | Se crean conductores correctamente | Sí |
| **CU 20** | Crear trayectos administrador | Se crean correctamente | Sí |

| | | | |
|---|---|---|---|
| **CU 21** | Confirman trayectos administrador | Se confirman correctamente | Sí |
| **CU 22** | Cancelar trayectos administrador | Se cancelan correctamente | Sí |
| **CU 23** | Modificar perfil administrado | Se modifican correctamente | Sí |
| **CU 24** | Eliminar perfil administrador | Se eliminan correctamente | Sí |

Teniendo en cuenta que los resultados de las pruebas de cajas de negra son satisfactorios, se concluye que el código implementado funciona lo esperado.

# 5. Normativa vigente

En el siguiente apartado se procede a detallar la normativa vigente actualmente que está relacionada y puede condicionar el proyecto actual.

## 5.1. Ley de protección de datos

En primer lugar, se tiene que definir que es la ley de protección de datos. Este proyecto, se ha desarrollado siguiendo las normas establecidas por Ley Orgánica 15/1999, del 13 de diciembre, de Protección de Datos de Carácter Personal (LOPD), que el 7 de diciembre de 2018 fue derogada con la entrada de la Ley Orgánica 5/2018 de Protección de Datos Personales y garantía de los derechos digitales (Ministerio de cultura, 2018), que adapta la legislación española al Reglamento General de Protección de Datos de la Unión Europea (Wikipedia, 2022).

*Figura 5.1: Cambios de la RPGD*

Para que la aplicación desarrollada cumpla con lo establecido en esta ley se ha procedido a tratar los datos personales de los usuarios siguiendo los principios y derechos de la protección de datos de carácter personal. Entre los derechos más importante y, que se han tenido en cuenta en el desarrollo esta: derecho de acceso, derecho de rectificación, derecho de supresión u olvido, derecho a la limitación del tratamiento, derecho a la portabilidad, derecho de oposición derecho a no ser objeto de decisiones individuales o automatizadas y derecho a retirar el consentimiento (Fundación Ande, 2020).

Unos ejemplos de funciones que se han adaptado debido a esta ley son los siguientes:

- Los usuarios que reservan trayectos saben cuál es el taxi que les va a transportar a través de la marca y matrícula, pero, en ningún momento conocen ni el nombre del taxista ni su número de licencia. Esto se debe a la protección de datos de carácter personal.
- Los usuarios podrán modificar en cualquier momento los datos personales que, voluntariamente, han cedido a la aplicación. Además, también podrán borrar sus perfiles, borrando así, toda la información que previamente ellos mismo habían cedido.

### 5.2. Ley de Propiedad Intelectual

El Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia, considera autor a la persona natural que crea alguna obra literaria, artística o científica (Ministerio de cultura, 1996).

Para corroborar que se realiza una correcta aplicación de esta ley, todas las imágenes que aparecen en la aplicación, son de creación propia, incluido el logo, el icono, el fondo, etc. Además, los otros iconos usados, son aquellos proporcionados por el framework de programación, que son abierto y están sujetos al copyright.

Además, para evitar que algún usuario realice o suba algún contenido que pueda incumplir esta ley, en ninguna de las funciones iniciales se permite la subida de imágenes a la aplicación.

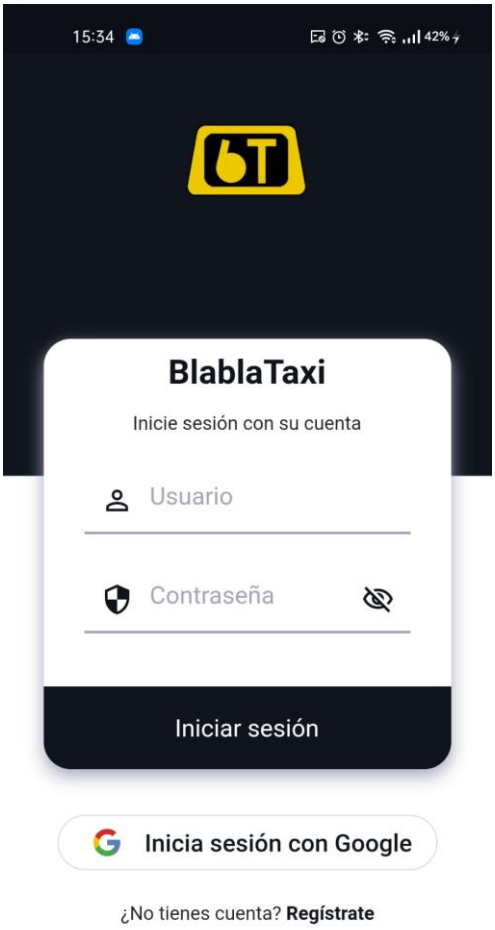
# 6. Producto final

En este apartado se procede a mostrar el resultado final de este proyecto.

## 6.1. Inicio de sesión

La pantalla de iniciar sesión es la pantalla principal de la aplicación. En esta pantalla el usuario podrá acceder a la aplicación. Este acceso lo podrá realizar mediante dos métodos distintos: por un lado, a través de un usuario y contraseña y, por otro lado, mediante una cuenta de Google.

*Figura 6.1: Pantalla de inicio de sesión*



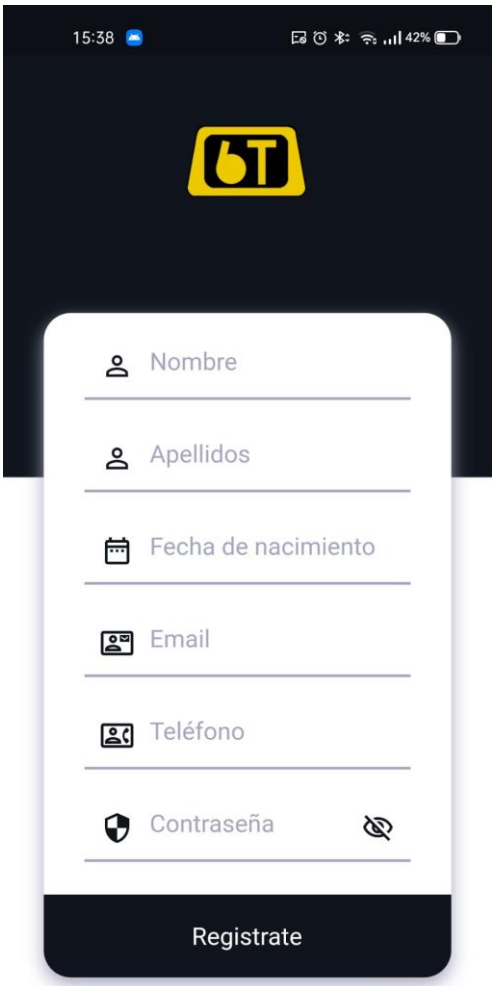

En esta misma pantalla se incluye un botón para acceder a la pantalla de registro.

### 6.2. Registrarse

La pantalla de registro de usuario está destinada a permitir el registro de nuevos usuarios. Para ello, el nuevo usuario deberá incluir los diferentes datos. Para poder registrarse deberá incluir dichos datos, los cuáles, deberán ser validados.

Los usuarios también podrán registrarse haciendo uso de una cuenta de Google.

*Figura 6.2: Pantalla de registro de usuarios*




### 6.3. Cerrar sesión

Una vez loggeado en el sistema, la sesión del usuario permanecerá abierto salvo que el mismo usuario decida cerrar la sesión. Para ello, hará uso del botón de cerrar sesión, que se encuentra situado en la esquina superior derecha.

### 6.4. Buscar viajes

La pantalla de búsqueda de viajes permite a los usuarios buscar nuevos viajes. En dicha pantalla el usuario deberá incluir el origen, destino, fecha y hora del trayecto que desea a realizar. Una vez puestos todos los filtros, el usuario deberá tocar el botón de "Buscar viajes".

*Figura 6.3: Pantalla de búsqueda de viajes*


A continuación, se procede a comprobar que los filtros descritos son correctos, en cuyo caso se procederá a la búsqueda de aquellos viajes que se ajusten a dichos filtros. En caso de que los filtros no sean correctos se mostrará un error que indique el porqué.

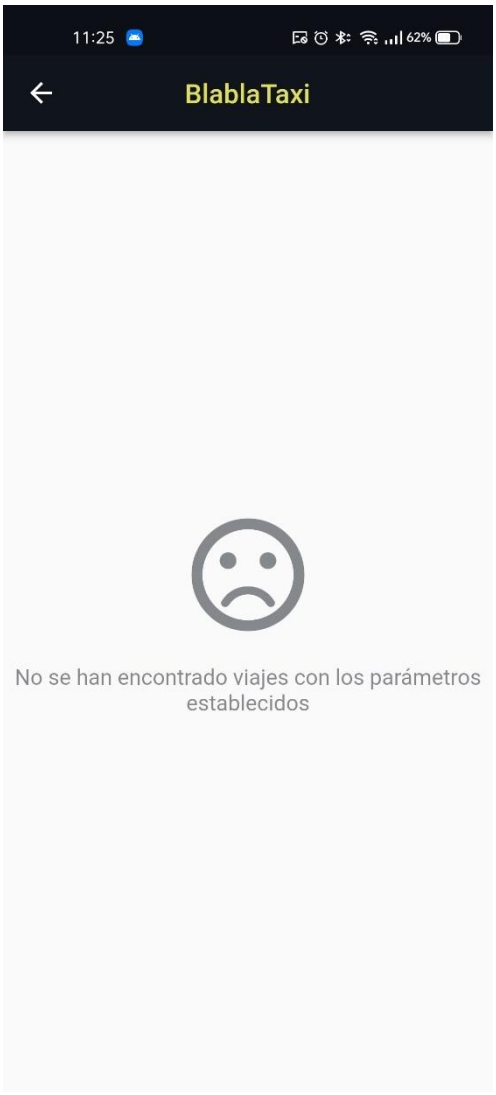
*Figura 6.4: Ejemplo de error en la búsqueda de viajes*


En caso de que el sistema encuentre viajes que se ajusten a los filtros establecidos, se mostrara una lista, con diferentes tarjetas que se corresponden con los posibles viajes a realizar.

*Figura 6.5: Ejemplo de búsqueda de viajes*


En caso de que ningún viaje se ajuste a los parámetros establecidos se mostrará una pantalla indicativa de este hecho.

*Figura 6.6: Ejemplo de viajes no encontrados*




Si el usuario desea obtener información extra sobre el trayecto, simplemente tendrá que tocar sobre la tarjeta correspondiente, y, se mostrará una nueva pantalla que contiene una foto del taxi que realizará el trayecto, información extra sobre el viaje e información sobre el coche y el conductor que se va a encargar de realizar el trayecto.

### 6.5. Consultar mis viajes

En esta pantalla se mostrarán los viajes realizados y reservados por el usuario que ha iniciado sesión. Si desea acceder a información extra sobre el viaje el usuario únicamente deberá tocar sobre la tarjeta correspondiente al viaje.

*Figura 6.7: Pantalla de mis viajes*


### 6.6. Perfil de usuario

En esta pantalla se le mostrará al usuario su información personal. De esta información podrá modificar la situada en la parte inferior, que es el número de teléfono y la fecha de nacimiento. Una vez realizadas las modificaciones debe confirmarlas tocando el botón de guardar información.

En esta misma pantalla el usuario podrá borrar su cuenta en el sistema.

*Figura 6.8: Pantalla de perfil*

# 7. Conclusiones

## 7.1. Aportaciones realizadas

La idea principal sobre la que se apoyó este proyecto era la de crear una aplicación móvil que permitiera a los usuarios compartir taxis con el objetivo de abaratar el coste individual de los trayectos.

Para conseguir este objetivo se procedió al desarrollo de una aplicación móvil haciendo uso de Flutter, que es una tecnología híbrida que permite el desarrollo de una aplicación desplegable tanto en Android como en iOS. Además, se creó una base de datos no relacional haciendo uso de MongoDB. También se creó un proyecto en NodeJS que permitía crear API's con los datos necesarios para ser consumidos en el FrontEnd.

Usando estas tecnologías se ha llegado a consecución de una aplicación móvil que permite reservar taxis con el objetivo de compartirlo entre usuario que vayan a realizar el mismo trayecto en una franja de tiempo similar.

Aunque, en la versión actual, no se han conseguido todos los objetivos establecidos inicialmente, se logrado una versión establece capaz de conseguir el objetivo primordial de la aplicación, compartir taxi.

En conclusión, se ha logrado la consecución de una aplicación móvil que permite a los usuarios compartir taxi. Además, se ha conseguido un diseño *responsive* que se adapta al tamaño de cada uno de los terminales en los que se instala la aplicación.

## 7.2. Trabajos futuros

A pesar de que la aplicación cumple con la mayoría de los requisitos existen algunos requisitos adicionales que en un futuro podrían ser implementados:

- Notificaciones emergentes sobre el estado de los viajes: confirmaciones, cancelaciones, estado, etc.
- Notificaciones en caso de que el usuario no acceda en un determinado tiempo a la aplicación.
- Confirmación de la cuenta de usuario mediante correo electrónico, de forma que no pueda acceder hasta que el usuario haya confirmado la cuenta.
- Permitir a los usuarios colocar una foto de perfil personalizada.
- Crear un modo administrador que permita crear viajes, conductores e incluso modificar los usuarios.

### 7.3. Problemas encontrados

Uno de los principales problemas encontrados durante el desarrollo de la aplicación está derivado con la creación y el hosting de las API's utilizadas para el consumo de datos en la parte de FrontEnd. Para solucionar esto finalmente, se optó por la creación de una cuenta en Amazon Web Services, donde se creó una máquina virtual de Ubuntu a través de la cual se realizó el hosting de las API's.

Otro de los problemas encontrados, se basó en el enlace de las colecciones de la base de datos en MongoDB. El principal problema, viene derivado de sacar los viajes realizados por cada uno de los usuarios de la aplicación.

### 7.4. Opinión personal

En primer lugar, desarrollar una aplicación móvil ha supuesto un gran reto para mí, dado que en los cuatro años de carrera no hemos visto nada que tenga que ver con esta temática.

Por otro lado, tenía que adaptarme a nuevo lenguaje de programación, Dart, junto con el framework correspondiente, algo que no fue fácil, y más teniendo en cuenta que el 95% del conocimiento generado en esta área ha sido autodidacta.

El último gran reto de este trabajo fue el hecho de realizar el cambio de trabajar con bases de datos relacionales a trabajar con bases de datos no relacionales, de forma que tenía que cambiar mi paradigma sobre el cual veía las bases de datos hasta ese momento.

## Agradecimientos

Antes de finalizar el desarrollo de esta memoria acerca del desarrollo de una aplicación móvil para la gestión y reserva de taxis, quería proceder con el apartado de los agradecimientos, para dar las gracias a todas aquellas personas que no solo me han apoyado durante el desarrollo de este trabajo, sino también durante los 4 años en los que he luchado por acabar este grado y por cumplir un sueño.

En primer lugar, agradecer a mis padres, Jose y Deli, y a mi hermano, Sergio, por ser esas personas que me han apoyado en todo momento, en mi día a día, en los momentos bueno y en los malos.

No podía realizar estos agradecimientos sin darle las gracias a Nelia, mi pareja y compañera de vida, mi apoyo fundamental en los momentos más complicados y la responsable de que nunca haya tirado la toalla y siempre haya seguido hacia delante, en especial durante estos 4 años.

Por último, agradecer el trabajo y dedicación de mi tutor de este Trabajo de Fin de Grado, José Alberto, que siempre ha tenido tiempo para atender mis inquietudes, dudas, problemas, los cuales no han sido precisamente pocos, y que siempre ha tenido tiempo para dedicárselo a corregir las numerosas versiones de este documento.

## Bibliografía

Antevenic. (13 de Octubre de 2016). *Subir app a App Store*. Recuperado el 27 de Mayo de 2022, de Subir app a App Store: https://www.antevenio.com/blog/2016/10/como-colgar-una-aplicacion-en-la-app-store/

Aures Tic. (10 de Julio de 2021). *¿Qué es Flutter?* Recuperado el 26 de Febrero de 2022, de AuresTic: https://aurestic.es/que-es-flutter/

Barrera, A. (14 de Agosto de 2020). *¿Qué son las aplicaciones híbridas?* Recuperado el 27 de Febero de 2022, de Aplicaciones híbridas: https://www.nextu.com/blog/aplicaciones-hibridas-que-son-y-como-usarlas/

BlaBlaCar. (2021). *¿Qué es BlaBlaCar?* Recuperado el 3 de Marzo de 2022

Chamorro, A. (14 de Julio de 2020). *Arquitectura MVVM*. Recuperado el 16 de Abril de 2022, de Arquitecturas y patrones de diseño en Flutter - Capítulo 1 - MVVM: https://medium.com/flutter-madrid/arquitecturas-y-patrones-de-diseño-en-flutter-capítulo-1-mvvm-591ad3d63bb

Devsdna. (10 de Febrero de 2022). *Publicar app en App Store*. Recuperado el 27 de Mayo de 2022, de Publicar app en App Store: https://www.devsdna.com/como-publicar-una-app-en-app-store/

Diví, V. (18 de Septiembre de 2019). *¿Qué es Dart?* Recuperado el 26 de Febrero de 2022, de inLabFIB.

Fundación Ande. (04 de Septiembre de 2020). *Derechos RGPD*. Recuperado el 01 de Mayo de 2022, de Derechos RPGD: https://www.fundacion-ande.org/derechos-rgpd/

García, I. J. (30 de Marzo de 2021). *FrontEnd y BackEnd*. Recuperado el 27 de Febrero de 2022, de FrontEnd y BackEnd.

Gimenez, M. (20 de Julio de 2020). *¿Qué es Amazon Web Services?* Recuperado el 28 de Febrero de 2022, de AWS: https://www.hiberus.com/crecemos-contigo/amazon-web-services-aws-que-es-y-que-ofrece/

IEBS. (9 de Diciembre de 2021). *¿Qué son las metodologías ágiles?* Recuperado el 2 de Marzo de 2022, de Metodologías ágiles: https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/

Indeed. (27 de Abril de 2022). *Salario medio desarrollador BackEnd*. Recuperado el 30 de Abril de 2022, de Indeed: https://es.indeed.com/career/backend-developer/salaries

Indeed. (27 de Abril de 2022). *Salario medio desarrollador FrontEnd*. Recuperado el 30 de Abril de 2022, de Indeed: https://es.indeed.com/career/programador-front-end/salaries

Indeed. (25 de Abril de 2022). *Salario medio jefe de proyecto*. Recuperado el 30 de Abril de 2022, de Indeed: https://es.indeed.com/career/jefe-de-proyecto/salaries

Indeed. (25 de Abril de 2022). *Salario medio Técnico de sistemas*. Recuperado el 30 de Abril de 2022, de Indeed: https://es.indeed.com/career/técnico-de-sistemas/salaries

Indeed. (26 de Abril de 2022). *Salario medio tester*. Recuperado el 30 de Abril de 2022, de Indeed: https://es.indeed.com/career/tester/salaries

Jefatura del Estado. (12 de Diciembre de 2018). *Ley de Protección de Datos Personales*. Recuperado el 02 de Mayo de 2022, de Ley de Protección de Datos Personales: https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673

JetBrains. (01 de Enero de 2022). *Licencia IntelliJ empresa*. Recuperado el 30 de Abril de 2022, de Licencia IntelliJ: https://www.jetbrains.com/es-es/idea/buy/#commercial

Kuna, H. D. (28 de Octubre de 2008). *Plan de riesgos*. Recuperado el 27 de Marzo de 2022, de Plan de riesgos: http://sedici.unlp.edu.ar/handle/10915/48365

Microsoft. (01 de Enero de 2022). *Licencia Microsoft Office empresas*. Recuperado el 30 de Abril de 2022, de Licencia Microsoft Office: https://www.microsoft.com/es-es/microsoft-365/p/office-professional-2021/CFQ7TTC0HHJ9?activetab=pivot%3aoverviewtab

Microsoft. (01 de Enero de 2022). *Licencia Windows 10 pro*. Recuperado el 30 de Abril de 2022, de Licencia Windows 10: https://www.microsoft.com/es-es/d/windows-10-pro/df77x4d43rkt?activetab=pivot%3aoverviewtab

Ministerio de cultura. (22 de Abril de 1996). *Ley de Propiedad Intelectual*. Recuperado el 02 de Mayo de 2022, de Ley de Propiedad Intelectual: https://www.boe.es/eli/es/rdlg/1996/04/12/1/con

Ministerio de cultura. (05 de Mayo de 2018). *Ley de Propiedad Intelectual*. Recuperado el 02 de Mayo de 2022, de Ley de Propiedad Intelectual: https://www.culturaydeporte.gob.es/cultura/propiedadintelectual/la-propiedad-intelectual/preguntas-mas-frecuentes/la-propiedad-intelectual.html

Overti. (22 de Marzo de 2016). *Técnica MoSCoW*. Recuperado el 17 de Marzo de 2022, de Técnica MoSCoW: https://www.overti.es/tecnologia/275-priorizacion-de-requisitos-tecnica-moscow

Robledano, A. (28 de Octubre de 2019). *¿Qué es MongoDB?* Recuperado el 26 de Febrero de 2022, de MongoDB: https://openwebinars.net/blog/que-es-mongodb/

Wikipedia. (19 de Marzo de 2022). *Ley Orgánica de Protección de Datos de Carácter Personal*. Recuperado el 01 de Mayo de 2022, de Ley Orgánica de Protección de Datos de Carácter Personal: https://es.wikipedia.org/wiki/Ley_Orgánica_de_Protección_de_Datos_de_Carácter_Personal_(España)

Xataka. (23 de Enero de 2019). *¿Qué son las VTC?* Recuperado el 2 de Marzo de 2022, de VTC: https://www.xataka.com/basics/que-significa-vtc-cuales-diferencias-taxis

Yeeply. (23 de Junio de 2020). *Como subir app a Google Play*. Recuperado el 27 de Mayo de 2022, de Como subir app a Google Play: https://www.yeeply.com/blog/guia-subir-app-google-play-store/

# Anexos

## Anexo I: Controlador de versiones

Para asegurarse del éxito del proyecto se ha hecho uso de un controlador de versiones, el cual es Git, durante todo el periodo de desarrollo del este [referencia].

*Figura 0.1: Logo de Git*

En este proyecto, realmente se han utilizado dos repositorios para la información del proyecto. Por un lado, un repositorio que contiene todo el proyecto del FrontEnd, y, por otro lado, un repositorio que contiene el proyecto que se encarga del BackEnd del mismo.

Teniendo en cuenta que se ha utilizado Git como controlador de versiones, se ha escogido como forja para alojas los dos proyectos GitHub [referencia].

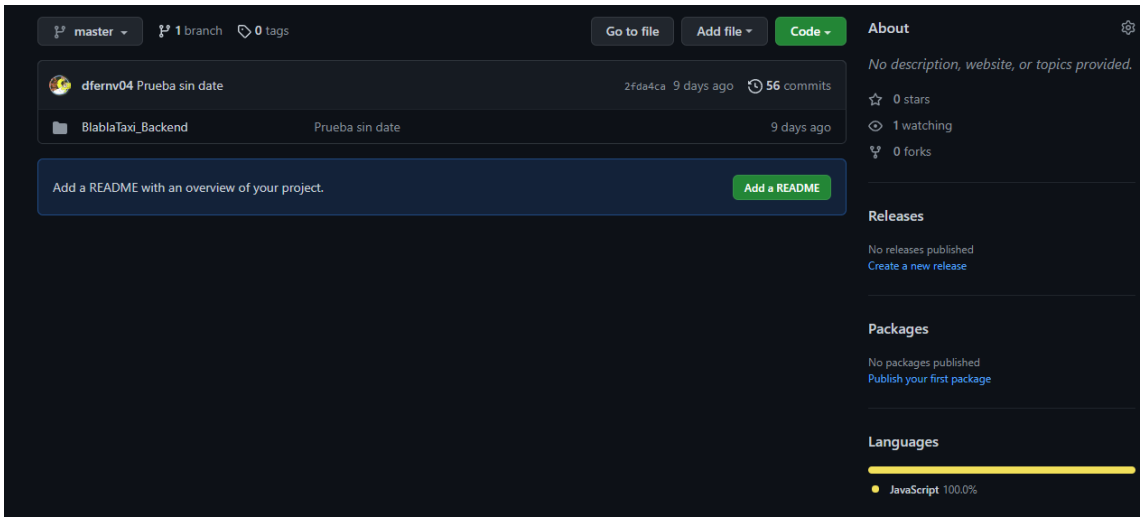
*Figura 0.2: Logo de GitHub*

Ambos repositorios, actualmente, tienen el acceso restringido con el objetivo de evitar copias y plagios.

En cuanto a la forma de trabajar en estos repositorios, ha sido distinta para cada uno:

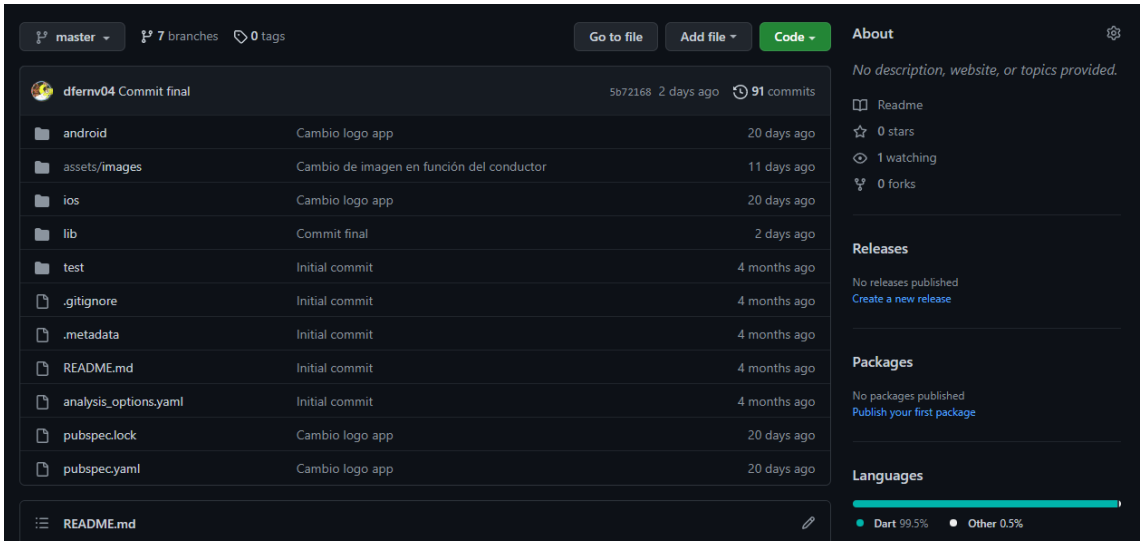
- En el repositorio correspondiente al BackEnd, se optó por trabajar siempre sobre la misma rama, la master, evitando así el uso de diferentes ramas. Se han hecho un total de 56 commits.

*Figura 0.3: Número de commits del repositorio BackEnd*




- En el repositorio correspondiente al FrontEnd, pese a que el trabajo ha sido individual, se ha optado por trabajar haciendo uso de diferentes ramas. Se han hecho un total de 91 commits.

*Figura 0.4: Número de commits del repositorio FrontEnd*

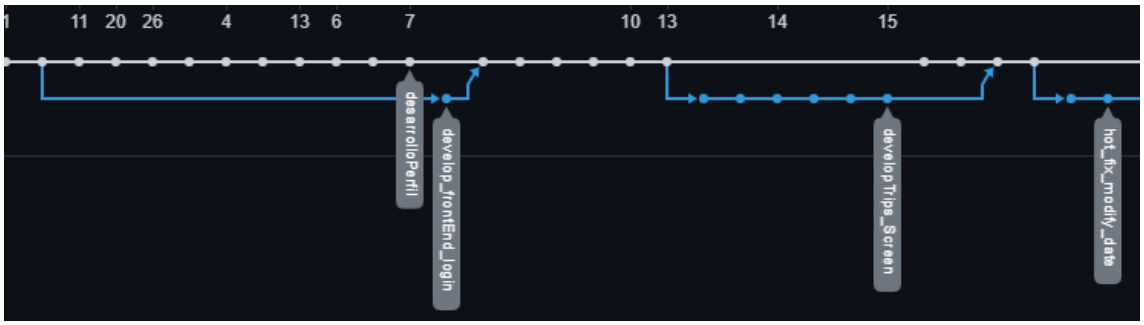



Se han ido creando y utilizando ramas para el desarrollo y pruebas de distintas funcionalidades.

Para desarrollar las diferentes pantallas se han utilizado ramas independientes, como por ejemplo: develop_frontend_login.

Una vez el contenido de la rama era estable, se fusionaba, “merge”, con el contenido de la rama master para continuar con el desarrollo del proyecto.

*Figura 0.5: Ejemplo de las ramas usadas*



## Anexo II: Manual de usuario

En esta sección se incluye un pequeño manual de usuario que explica el funcionamiento básico y normal de la aplicación. Además se incluye un video explicativo donde se muestra de una forma más gráfica el funcionamiento de la misma.

Enlace al video tutorial: https://youtu.be/knw4kYyxaSo

A continuación se expone el manual de usuario:

2022

# BlablaTaxi

Diego Fernández Velasco

BlablaTaxi

6-6-2022

Diego Fernández Velasco

# Índice

1. Introducción ..... 96

2. Inicio de sesión ..... 97

2.1. Inicio de sesión con usuario y contraseña ..... 97

2.2. Inicio de sesión con cuenta de Google ..... 98

3. Registrarse ..... 100

3.1. Registro con datos ..... 100

3.2. Registro con cuenta de Google ..... 101

4. Cerrar sesión ..... 102

5. Buscar viajes ..... 103

6. Reservar viajes ..... 105

7. Consultar viajes ..... 107

8. Información extra de los viajes ..... 108

9. Consultar perfil ..... 110

9.1. Modificar datos de perfil ..... 110

9.2. Eliminar cuenta ..... 111

# 1. Introducción

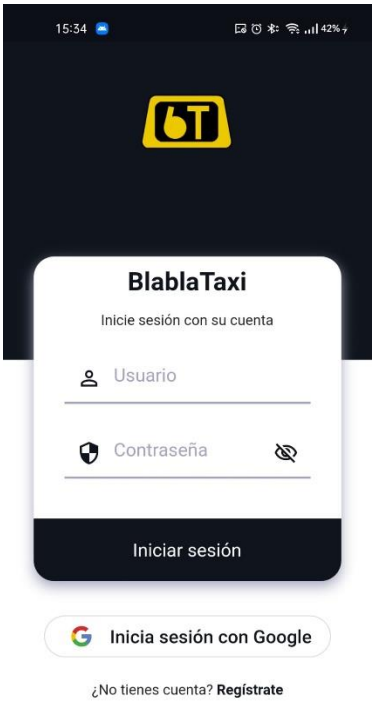
Esta aplicación móvil está diseñada para la gestión y reserva de taxis compartidos. En esta aplicación, que está disponible tanto en Android como en iOS, los usuarios podrán buscar los viajes en función de la fecha y hora elegida, consultar los viajes realizados, y los que realizará en futuras fechas, y podrá consultar su perfil, así como modificar sus datos.

## 2. Inicio de sesión

Para iniciar sesión, el usuario debe acceder a la pantalla principal de la aplicación. Además previamente ha tenido que registrar una cuenta.

Para iniciar sesión existen dos método posibles.

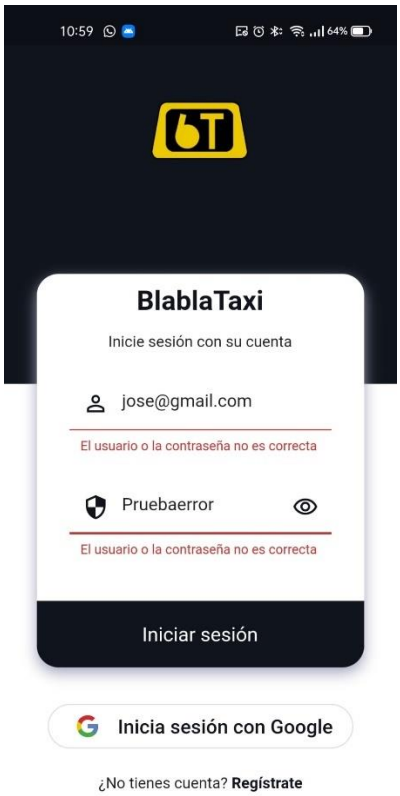
*Figura 2.1: Pantalla principal*




### 2.1. Inicio de sesión con usuario y contraseña

Para iniciar sesión, el usuario deberá introducir su nombre de usuario y contraseña en los campos habilitados para ello.

En caso de error en los datos introducidos, el mensaje mostrado por la aplicación será el siguiente: “El usuario o la contraseña no es correcta”.

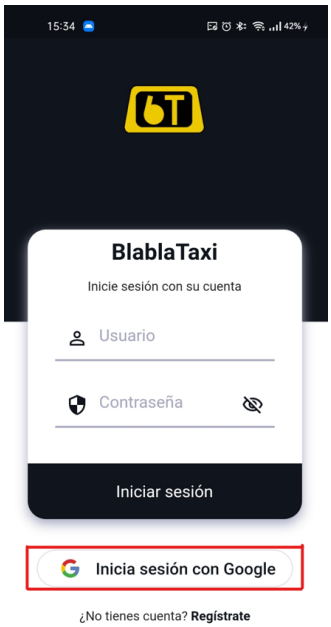



T

### 2.2. Inicio de sesión con cuenta de Google

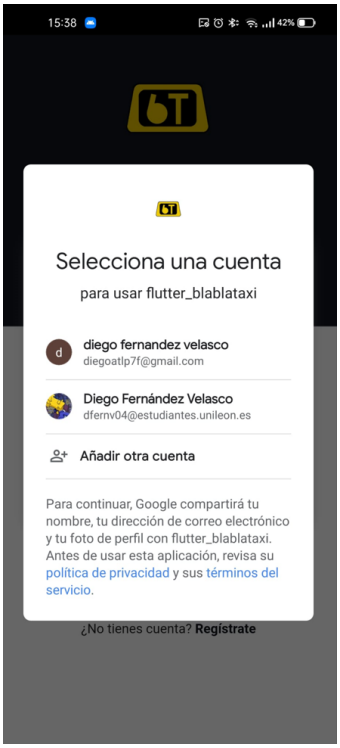
Para iniciar sesión en el sistema por medio de la cuenta de Google, el usuario deberá tocar en el botón de “Inicia sesión con Google”.

*Figura 2.2: Botón de inicio de sesión con Google*




Al usuario el saldrán las cuentas de Google asociadas a dicho usuario y podrá seleccionar aquella que desee.

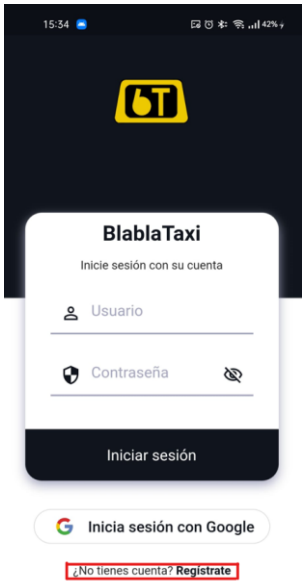
*Figura 2.3: Ejemplo de inicio de sesión con Google*



# 3. Registrarse

Para registrar una cuenta, el usuario deberá proveer una serie de datos al sistema. Para ello previamente, y desde la pantalla principal de aplicación, la de inicio de sesión, deberá tocar sobre “Regístrate”.

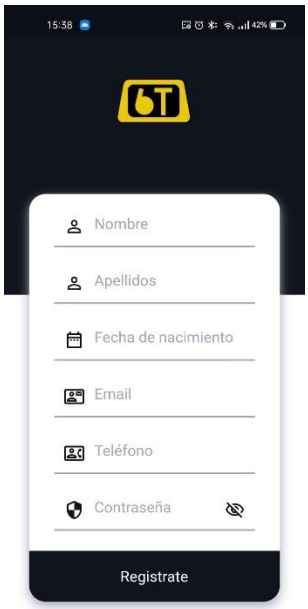
*Figura 3.1: Botón de "Regístrate" de la pantalla principal*




## 3.1. Registro con datos

Para registrarse, el usuario deberá indicar de forma correcta los siguientes datos: nombre, apellido, fecha de nacimiento, email, teléfono y contraseña.

*Figura 3.2: Pantalla de registro con datos*




En caso de que los campos introducidos no cumplan con los requisitos establecidos se mostrarán errores personalizados que indiquen el porqué del error.

*Figura 3.3: Pantalla de registro con errores*


### 3.2. Registro con cuenta de Google

Sigue exactamente el mismo proceso que iniciar sesión con cuenta de Google.

## 4. Cerrar sesión

Por defecto la sesión se mantendrá iniciada, por tanto, si el usuario desea cerrar la sesión, deberá acceder al botón destinado a ello.

*Figura 4.1: Botón de cerrar sesión*


Simplemente tocando este botón el usuario cerrará la sesión y será redirigido a la pantalla principal de la aplicación.

## 5. Buscar viajes

En esta pantalla el usuario deberá introducir los datos correspondientes con el viaje que desea realizar, para ello deberá rellenar los siguientes campos:

- Origen: deberá seleccionar una ubicación de entre todas las posibles.
- Destino: deberá seleccionar una ubicación de entre todas las posibles.
- Fecha: fecha en la que el usuario desea realizar el viaje.
- Hora: hora aproximada en la que el usuario desea realizar el viaje. El sistema mostrará viajes con una margen de tiempo 2 horas superior y 2 horas inferior a la seleccionada por el usuario.

*Figura 5.1: Pantalla de buscar viajes*


Una vez seleccionados todos los campos el usuario deberá tocar el botón de buscar viajes para que el sistema busque los viajes que se correspondan con los filtros establecidos.

En caso de error con la fecha y/u hora seleccionada el sistema mostrará un error personalizado.

*Figura 5.2: Error al buscar viajes*

## 6. Reservar viajes

Una vez establecidos los filtros en la pantalla de anterior de búsqueda, el usuario será redirigido a una nueva pantalla donde existen dos opciones:

- Que se muestren los diferentes viajes que se ajusten a los filtros.

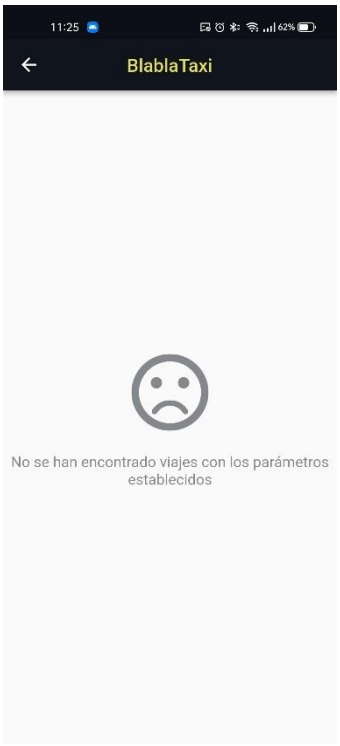
*Figura 6.1: Lista de viajes mostrados al usuario*


Una vez mostrados todos los viajes que se ajustan a los filtros el cliente deberá seleccionar aquel que se ajuste a sus necesidades. Para ellos deberá tocar sobre el botón de “Reservar viaje”.

Si el usuario desea obtener información extra sobre el viaje, deberá tocar sobre la tarjeta del viaje.

- Que se muestre un mensaje que indica que no se ha encontrado ningún viaje que se ajuste a los filtros indicados por el usuario.

*Figura 6.2: Pantalla de viajes no encontrados*



## 7. Consultar viajes

En esta pantalla el usuario puede consultar los viajes que ha realizado, así como aquellos que tiene reservados y pendientes de realizar. A continuación, se muestra un ejemplo de cómo se ve la pantalla de un usuario con varios trayectos realizados.

*Figura 7.1: Pantalla de mis viajes*

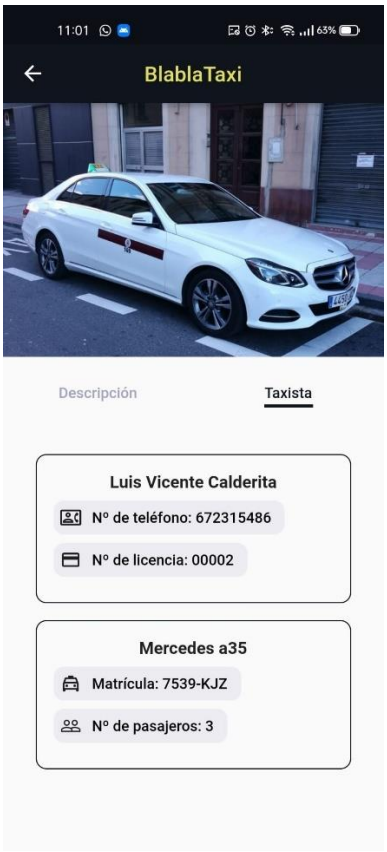

Si el usuario desea obtener información extra del viaje deberá tocar sobre la tarjeta correspondiente al viaje.

## 8. Información extra de los viajes

En esta pantalla el usuario podrá consultar información extra sobre los viajes. En esta pantalla encontrará un foto del taxi que se encarga de realizar el trayecto así como dos pestañas. La primera pestaña contiene información sobre el viaje como origen, destino, hora, etc. La segunda pestaña contiene información sobre el taxista, para poder reconocer el coche que se encarga del trayecto y el conductor encargado de realizarlo.

*Figura 8.1: Pantalla de información extra del viaje*

*Figura 8.2: Pantalla de información extra sobre el conductor*



## 9. Consultar perfil

En esta pantalla el usuario podrá consultar datos sobre su información introducida en el sistema: nombre, email y rol. Este último, vienen determinado por el número de viajes realizados, siendo 3 los posibles roles:

- Bronce: si ha realizado menos de 10 viajes.
- Plata: si ha realizado más de 10 viajes y menos 50.
- Oro: si ha realizado más de 50 viajes.

En la otra tarjeta aparece el número de teléfono y la fecha de nacimiento.

*Figura 9.1: Pantalla de perfil*


### 9.1. Modificar datos de perfil

Para modificar el número de teléfono y la fecha de nacimiento el usuario deberá tocar sobre el icono del lápiz, después deberá modificar los datos deseados y, por último, deberá tocar sobre el icono de guardar, para confirmar los cambios.

*Figura 9.2: Pantalla modificar datos de perfil*


Si al introducir los nuevos datos se comprobará que cumple con los requisitos.

### 9.2. Eliminar cuenta

El usuario podrá eliminar su cuenta en cualquier momento, para ello deberá tocar sobre el botón de “Eliminar cuenta”, una vez eliminada, será redirigido a la pantalla de login de la aplicación.