

## MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Multiplataforma

# Vehicle Maintainer

**Autor:** David Buisan Berrocal

**Tutor:** Isabel Duarte Tosso

**Fecha de entrega:** 01/12/2023

**Convocatoria:** 1s2324

**Documentos del proyecto:**

<https://drive.google.com/drive/folders/1JucdTU1RriwwY3CDNAjDlbzxI4XhEGo6?usp=sharing>



## Índice de contenidos

1. Introducción (3-4 páginas) .....	3
1.1. Motivación.....	4
1.2. Abstract.....	5
1.3. Objetivos propuestos (generales y específicos).....	6
2. Estado del Arte (4-5 páginas) .....	7
3. Metodología usada (2-4 páginas).....	13
4. Tecnologías y herramientas utilizadas en el proyecto (2-3 páginas).....	15
5. Planificación, Diagnóstico y Contexto Laboral (3-4 páginas) .....	18
6. Análisis del proyecto (8-16 páginas) .....	21
7. Diseño del proyecto (6-14 páginas).....	39
8. Despliegue y pruebas (4-10 páginas) .....	50
9. Conclusiones (1-2 páginas) .....	59
10. Vías futuras (1-2 páginas).....	60
11. Bibliografía/Webgrafía (1-2 páginas).....	61
12. Anexos.....	63
1.1. Manual de Usuario .....	64

## 1. Introducción (3-4 páginas)

### Justificación del tema:

La gestión de vehículos es una necesidad que tienen la mayoría de las personas, ya sea un único usuario, o una empresa que tenga una flota de vehículos a su cargo. Una mala gestión de los mantenimientos y seguimiento de los vehículos puede llevar a problemas costosos, e incluso, irreversibles. Además, llevar los vehículos al día, en cuanto a mantenimientos se refiere, puede aumentar la fiabilidad y seguridad para los usuarios.

Por ese motivo, el desarrollo de una aplicación para la gestión de vehículos aborda una necesidad fundamental para el mantenimiento, la seguridad y la eficiencia de los mismos.

### Resumen del contenido del trabajo

El trabajo de desarrollo de la aplicación “Vehicle Maintainer” recoge los siguientes aspectos:

1. Introducción: Proporciona una visión general del proyecto, se justifica la elección de la aplicación desarrollada, así como, la motivación de elección del tema, un breve resumen de la aplicación en inglés (Abstract) y los objetivos, generales y específicos, que se pretenden alcanzar con el desarrollo.
2. Estado del arte: En este apartado, se presenta un análisis de otras aplicaciones del mercado que están relacionadas con la gestión de vehículos, así como sus principales funcionalidades y tendencias.
3. Metodología usada: En esta sección se recogen la metodología utilizada para el desarrollo del proyecto, el ciclo de vida y las fases de desarrollo.
4. Tecnologías y herramientas utilizadas en el proyecto: Se proporcionan las herramientas y tecnologías que se han utilizado para realizar el proyecto y las causas principales de la elección de las mismas.
5. Planificación, diagnóstico y contexto laboral: En esta sección se recoge la planificación del proyecto que incluye los diagramas de Gantt y el análisis DAFO.
6. Análisis del proyecto: En este apartado, se realiza un análisis detallado de la aplicación donde se incluyen los requisitos funcionales y los no funcionales, también, se presentan los diagramas, de casos de uso, de clases y de entidad relación.

7. Diseño del proyecto: En este punto, se crean los distintos layouts de la aplicación a partir de Wireframes y Mockups, además se explican los detalles que tienen más relevancia para el desarrollo.
8. Despliegue y pruebas: Se explica cómo ha sido la implementación de la aplicación y se describen las pruebas realizadas para comprobar su correcto funcionamiento y comprobar si se han alcanzado los requisitos de las funcionalidades necesarias.
9. Conclusiones: En esta parte, se resume los objetivos alcanzados, los problemas que han surgido y la experiencia adquirida tras el desarrollo de la aplicación.
10. Vías futuras: En esta sección, se sugieren posibles las vías futuras de la aplicación recogiendo posibles mejoras y nuevas funcionalidades.
11. Bibliografía/Webgrafía: Se describen las fuentes de información consultadas para el desarrollo de la aplicación.
12. Anexos: En esta parte se incluye documentación adicional para complementar el proyecto, así como el manual de instalación para el usuario final de la aplicación donde se explican todas las funcionalidades y características.

En resumen, el trabajo recoge el desarrollo y la implementación de una aplicación que se encarga de la gestión de vehículos y mantenimientos, que brinda a los usuarios una manera fácil de poder registrar, mantener y gestionar vehículos eficientemente.

### 1.1. Motivación

La idea principal nace de una necesidad personal que me surgió mientras pensaba en un tema para el proyecto, ya que en ese momento tenía que pasar la revisión de mi vehículo y no lograba recordar que fue lo revisado con anterioridad. Además, tuve la mala fortuna de perder la tarjeta de la última revisión.

Dado que en la familia disponemos de dos vehículos y uno de ellos no dispone de libro de revisiones, pensé en la posibilidad de poder gestionar de manera más eficiente cada vehículo y tener toda la información relacionada con los mantenimientos en un mismo lugar almacenada.

Así mismo, sería de gran utilidad si pudiera recibir notificaciones con anterioridad y no tener que estar pendiente de kilometrajes y demás.

A continuación, analicé el problema en cuestión y, pensando en realizar una solución a un problema en general, anoté una serie de antecedentes que detallo a continuación.

**Dificultades en la gestión del mantenimiento:** Si la gestión del mantenimiento se realiza de manera manual, puede llevar a olvidos, pérdidas de documentos, ya sea libro de mantenimiento, la típica tarjeta colgada o la inexistencia de los mismos.

**Dependencia de vehículos:** En la sociedad que vivimos, dependemos en gran medida de los vehículos, ya sea para trabajar, viajar y diversas actividades. Por esa razón, el mantenimiento adecuado de los vehículos es de gran importancia.

**Importancia del mantenimiento:** Realizar los mantenimientos adecuadamente alarga la vida útil de los vehículos, aumentando su fiabilidad y seguridad. Sin embargo, en ocasiones no se lleva un correcto mantenimiento de los vehículos debido a olvidos o faltas de recordatorios.

**Necesidad de notificaciones automatizadas:** Poder tener notificaciones automatizadas es una muy buena herramienta, ya que, podrá avisar al usuario cuando hay que realizar el mantenimiento y evitará el olvido.

En resumen, los antecedentes que han surgido requieren una solución más eficiente y los dispositivos móviles ofrecen la oportunidad de poder desarrollar una aplicación que resuelva dichos antecedentes y ofrezca al usuario mayor facilidad para poder llevar a cabo el mantenimiento de los vehículos de una manera óptima y automatizada.

## 1.2. Abstract

Due to the importance of doing periodic maintenance on your vehicles because vehicle maintenance prolongs its useful life and can avoid unnecessary money expenses, Vehicle Maintainer comes to help you in the management and maintenance of your vehicles. Vehicle Maintainer is an app intended for the management and maintenance of your vehicles. With this app you will be able to manage and track your vehicles and all the maintenance that you have carried out on each vehicle, having everything related to your vehicles in one place. Additionally, you will be able to schedule maintenance, either schedule maintenance by date or schedule maintenance by mileage, so you don't forget when you need to service your vehicle. With Vehicle Maintainer you can register vehicles, you can record maintenance done and you can also schedule new maintenance. In addition, you can consult a list of maintenance done, consult a list of scheduled maintenance and consult a list of the vehicles that you have registered in the application. The app will use notifications to always keep you informed when new maintenance is required.

### 1.3. Objetivos propuestos (generales y específicos)

En este apartado se recogen el conjunto de objetivos los cuales se pretenden alcanzar al finalizar el desarrollo de la aplicación.

#### Objetivos generales

Desarrollar una aplicación para la gestión de vehículos: El objetivo principal es desarrollar una aplicación en la que el usuario pueda gestionar sus vehículos de manera efectiva.

Facilitar el mantenimiento de vehículos: La aplicación tiene que facilitar al usuario el mantenimiento de vehículos enviándole notificaciones automáticas a modo de recordatorio de cuando hay que realizar un mantenimiento.

Centralizar la información en un único aplicativo: La aplicación tiene que poder almacenar los datos más relevantes de cada vehículo, así como los mantenimientos realizados.

#### Objetivos específicos

Registro de vehículos: El usuario tiene que poder registrar vehículos en la aplicación.

Consultar datos de vehículo: El usuario debe poder consultar los datos pertenecientes a cada vehículo.

Programación de mantenimientos: La aplicación ha de ser capaz de permitir a los usuarios programar mantenimientos para sus vehículos, pudiendo hacerlo por fecha o por kilometraje.

Notificaciones automáticas: La aplicación debe enviar notificaciones al usuario cuando se acerque la fecha o el kilometraje programado en el mantenimiento.

Registro de mantenimientos: El usuario debe poder registrar los mantenimientos realizados a los vehículos.

Consultar mantenimientos: El usuario a de poder consultar una lista de mantenimientos programados y realizados de cada vehículo, incluyendo detalles de los mismos.

Consultar mensajes: El usuario debe poder consultar y gestionar los mensajes recibidos pudiendo eliminarlos.

Interfaz intuitiva: La aplicación ha de ser fácil de usar e intuitiva, incluso sin la necesidad de leer el manual de usuario.

Compatibilidad con múltiples dispositivos: La aplicación ha de ser compatible con el mayor número de dispositivos Android.

Rendimiento: La aplicación debe ser eficiente en cuanto a la velocidad de consulta y manipulación de los datos para garantizar una buena experiencia al usuario.

## 2. Estado del Arte (4-5 páginas)

En la actualidad, hay un aumento considerable de vehículos en España, según un artículo publicado por la UNESPA (Unión española de entidades aseguradoras y reaseguradoras) [\[1\]](#) el número de unidades ha crecido un 1,44% durante el segundo trimestre de 2023. Este aumento demuestra la dependencia que tenemos de nuestros vehículos y, de ahí, la necesidad de realizar mantenimientos periódicos y preventivos para el correcto funcionamiento y la eficiencia de los mismos.

Con la llegada de las nuevas tecnologías y los dispositivos móviles se abre un abanico de posibilidades para ofrecer soluciones a los usuarios con una larga lista de aplicaciones.

En el proceso de hacer un estudio sobre el tema y las nuevas tecnologías que se pueden incorporar a la aplicación, se procede a realizar una búsqueda de información por varias páginas web y foros de automoción. También, se analizan las aplicaciones que nos ofrece la tienda de aplicaciones de Android, la Play Store [\[2\]](#).

En primera instancia, se ha realizado la búsqueda de aplicaciones para el mantenimiento de vehículos en la Play Store para saber cuáles son las aplicaciones mejor valoradas por los usuarios.

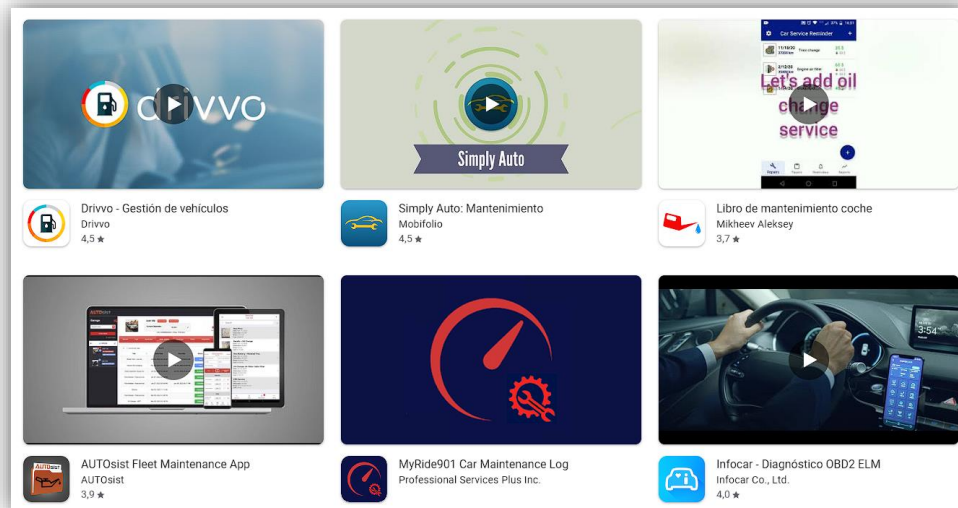


Imagen 1. Búsqueda de aplicaciones para el mantenimiento de vehículos. Google Play

El resultado de la búsqueda realizada se utilizó para hacer una comparación con la posterior investigación, donde se efectuó la lectura de foros de automoción y comparativas de las mejores aplicaciones en el campo.

De toda esa búsqueda de información cabe resaltar la página web de Fiact Seguros [3] donde posee un foro que efectúa una recopilación de aplicaciones recomendadas, que de todas las páginas web consultadas, esta ha sido la que tiene la información más actualizada.

Después del análisis realizado y comparando los resultados devueltos en la investigación, se detecta que en la actualidad existen varias aplicaciones que han sido desarrolladas para ayudar a los usuarios en la gestión y el mantenimiento de los vehículos, algunas de ellas muy bien valoradas por la comunidad y con múltiples soluciones distintas a nuestra disposición, desde lectura de errores, gestores de gastos, consumo de combustibles y demás.

A continuación, se muestra un análisis más al detalle algunas de las aplicaciones líderes en este campo.

Una de las aplicaciones, que ha estado presente más veces y mejor valoradas por los usuarios ha sido Drivvo [4].

Tras el descubrimiento de esta aplicación y la posterior instalación de la misma, se procede a comprobar y analizar todas las opciones que ofrece.





## Drivvo

Drivvo se presenta como una solución muy completa, en cuanto al mantenimiento de vehículos se refiere, contando con una interfaz muy atractiva a la vista y con una paleta de colores de tonalidades frías.

La primera impresión que transmite la interfaz de usuario es de una imagen sobria y profesional. Después de introducir los datos del vehículo, muestra un histórico de acciones realizadas que puedes ir gestionando, tales como gastos, repostaje, servicios, etc.

Consta de un menú principal que se encuentra situado en una barra inferior y permanece ahí a lo largo de todos sus distintos layouts. El menú está muy sintetizado, pero al pulsar en alguna de sus cinco opciones se abre un sinfín de funcionalidades disponibles.

Indagando al detalle en las muchas opciones que ofrece, puedo decir que Drivvo es una aplicación que nace mayormente para hacer una gestión de los gastos del vehículo y aunque posee un apartado para recordatorios noto que no es muy intuitiva a la hora de realizar la gestión o programación de servicios.

Otra cosa negativa, a mi parecer, es la gran variedad de alternativas que presenta, cosa que en principio puede parecer una solución muy completa, al navegar por sus distintas pantallas y menús disponibles puede llegar a abrumar dada la complejidad de la aplicación.

Tras el análisis realizado de la aplicación, recojo un conjunto de ventajas e inconvenientes detectados.

Ventajas	Inconvenientes
Interfaz amigable.	Aplicación poco intuitiva.
Aplicación muy completa.	Demasiada complejidad.
Acabado profesional.	

Tabla 1. Ventajas e inconvenientes de Drivvo

Otra de las aplicaciones mejor valoradas y con notable participación en la investigación ha sido Simply Auto [\[5\]](#).



## Simply Auto

Esta app lo primero que llama la atención es que informa al usuario, mediante el uso de notas informativas a modo de tutorial, que describen el funcionamiento de la aplicación, cosa que se agradece dado la variedad de funcionalidades que presenta.

La interfaz de usuario consta de tonalidades frías y un menú situado en la parte superior de la pantalla en el que dispone de un título, una barra de búsqueda y un icono con tres líneas que al pulsar aparece un menú desplegable con varias funcionalidades disponibles.

Siempre, al iniciar una nueva función por primera vez, la aplicación te acompaña y te guía explicándote cómo has de llevar a cabo la funcionalidad que ofrece.

Esta aplicación hace uso de geolocalización para poder guardar rutas, almacenar lugares como gasolineras y mapas de viajes, aunque algunas de sus funcionalidades descritas solo están disponibles en su versión de pago.

Otra cosa a destacar es que puedes hacer copias de seguridad mediante Google Drive [\[6\]](#) haciendo uso de una cuenta Gmail [\[7\]](#) y ofrece la gestión del vehículo desde múltiples dispositivos.

Además, ofrece la posibilidad de visualizar los gastos referentes al vehículo programando un reporte de gastos pudiendo elegir que sea semanal o mensualmente.

También, proporciona la opción de programar recordatorios que pueden ser cada x kilómetros por fecha o por lo que suceda primero. Aunque la opción de por kilometraje no funciona correctamente, ya que al actualizar el número de este no ocurre absolutamente nada y en este caso tampoco existe un feedback o nota explicativa.

Como en el caso de la aplicación anteriormente analizada, Simply Auto es una aplicación que está más orientada a la gestión de gastos de vehículos, pero añadiendo aún más funcionalidades, siendo una aplicación que, en cuanto a gestión de gastos y visualización de los mismos, es muy completa y grafica.

Algo a resaltar negativamente es que, al instalar la aplicación, se ha detectado que algunos de gran variedad de sus campos o títulos descriptivos están en inglés, cosa que dificulta a la hora de introducir datos a personas que no tengan conocimiento de dicho idioma y descoloca un poco la falta de cohesión de la aplicación.

A continuación, recojo el conjunto de ventajas e inconvenientes detectados.

Ventajas	Inconvenientes
Interfaz amigable.	Falta de cohesión.
Aplicación muy completa.	Funcionalidad cuestionable en algunos aspectos.
Aplicación muy gráfica.	

Tabla 2. Ventajas e inconvenientes de Simply Auto

Por otro lado, deseo hacer mención de Torque Lite [\[8\]](#) que, aunque no es una aplicación destinada 100% a la gestión de vehículos, se utiliza para obtener datos del vehículo en tiempo real.



### Torque Lite

Torque es una aplicación que se utiliza para recoger datos en tiempo real pudiendo hacer lectura de errores, datos de rendimiento, códigos de error, etc.

Lo llamativo de la aplicación es que hace uso de la tecnología OBD (On Board Diagnostic) para conectarse al vehículo y pudiendo conectarse vía bluetooth. Esto hace que pueda monitorear en todo momento el estado del vehículo.

Como aspecto negativo, es el hecho de tener que comprar un dispositivo OBD adaptado a tu vehículo para poder hacer uso de la aplicación, cosa que ha hecho que no pueda realizar un análisis más detallado de la misma al no poseer este dispositivo.

Seguidamente se muestran las ventajas e inconvenientes detectados.

Ventajas	Inconvenientes
Conexión al vehículo mediante bluetooth.	Necesidad de adquirir un dispositivo OBD.
Diagnostico en tiempo real.	Compatibilidad con dispositivos reducida.

Tabla 3. Ventajas e inconvenientes de Torque Lite

Tras el estudio realizado en el campo de las aplicaciones relacionadas con el mantenimiento de vehículos y después de haber analizado las aplicaciones líderes en el campo se han detectado una serie de tendencias y tecnologías utilizadas. Por otra parte, se han podido detectar vacíos que podrían ser mejorados.

Las aplicaciones para la gestión de vehículos tienden a poder gestionar principalmente lo que a gastos se refiere, ya sea de mantenimientos, combustible y demás. También, ofrecen en su mayoría la posibilidad de programar recordatorios para los mantenimientos o servicios.

A pesar de todas las aplicaciones disponibles, la posibilidad de integrar el teléfono móvil al vehículo para poder hacer lectura del kilometraje y, de tal manera, gestionar de manera más eficiente los recordatorios, sería una buena solución en el campo de la gestión de vehículos.

En conclusión, en la actualidad existen una gran variedad de aplicaciones de gestión y mantenimiento de vehículos con un gran número de funcionalidades disponibles. Por el contrario, no he hallado una aplicación que sea exclusivamente enfocada en cubrir la gestión de mantenimientos de vehículos de manera eficiente e intuitiva para el usuario final. Además, queda por resolver cuestiones tales como la integración de las aplicaciones móviles con los vehículos de los usuarios.

### 3. Metodología usada (2-4 páginas)

En este punto se describen la metodología utilizada para el desarrollo de la aplicación Vehicle Maintainer, tales como el ciclo de vida y las fases de desarrollo.

#### Ciclo de vida del proyecto

Al iniciar el desarrollo de la aplicación, en primer lugar, se ha analizado el proyecto teniendo en cuenta los objetivos propuestos y el conjunto de los requisitos necesarios para llevar a cabo el desarrollo con éxito. Tras el análisis, se decidió elegir un ciclo de vida en cascada debido a que los requisitos eran estables, estaban bien definidos y no se necesitan versiones intermedias de la aplicación.

También, se eligió este modelo debido a la facilidad de planificación y comprensión del modelo y al ser una buena opción para afrontar el desarrollo de la aplicación siendo una única persona.

Otro punto importante que favoreció la elección de este modelo en particular es por la opción que ofrece de poder aplicar la variante de modelo en cascada con realimentación, ya que me permitirá volver a atrás, a una etapa anterior del desarrollo, si fuera necesario.

A continuación, se muestran las fases que componen el ciclo de vida del proyecto.

- Planificación: Para la planificación del proyecto y que pudiera entregarse a tiempo, se ha realizado un diagrama de Gantt donde se recogen las fases del desarrollo de la aplicación y la estimación de los tiempos iniciales. También, para poder gestionar las fases del desarrollo se hizo uso de Kanban.
- Análisis: En la fase de análisis se han recogido el conjunto de requisitos funcionales y no funcionales y se han llevado a cabo la definición de los Casos de Uso junto con la realización del diagrama de Casos de Uso, de Entidad-Relación y de clases.
- Diseño: Se realizan el conjunto de wireframes y mockups pertenecientes a la aplicación, así como la elección de la paleta de colores utilizada para la interfaz de usuario.
- Despliegue y pruebas: En esta fase se realiza la codificación del sistema, llevando a cabo la implementación de la BD, del front-end y del back-end. Además, se realizan las pruebas del sistema para comprobar que cumple con su cometido.

- Documentación: En esta fase se finaliza la memoria escrita del proyecto que se ha ido desarrollando a lo largo de este, se realiza la presentación del documento que se apoyará la presentación del video de defensa y la grabación del mismo.

## Metodología ágil

Para la correcta gestión del tiempo y la planificación del desarrollo, se ha utilizado la metodología ágil Kanban.

Para ello hice uso de Trello [\[9\]](#), aquí cree tres listas una llamada por hacer, otra en proceso y por último hecho. Después, me dedique a crear varias tarjetas, cada una con una definición de tarea a realizar y con un color específico para cada fase del desarrollo de la aplicación, azul celeste para la fase de análisis, azul para la fase de diseño, lila para la fase de despliegue y pruebas y gris para la fase de documentación.

La utilización de Kanban me favoreció a tener una visión general del desarrollo en todo momento gracias a sus tarjetas visuales y la división de sus bloques, pudiendo ver en todo momento que tareas se han realizado, las que quedan por hacer y las que se están llevando a cabo en ese momento, cosa que permitió tener una manera ideal de gestionar el tiempo.

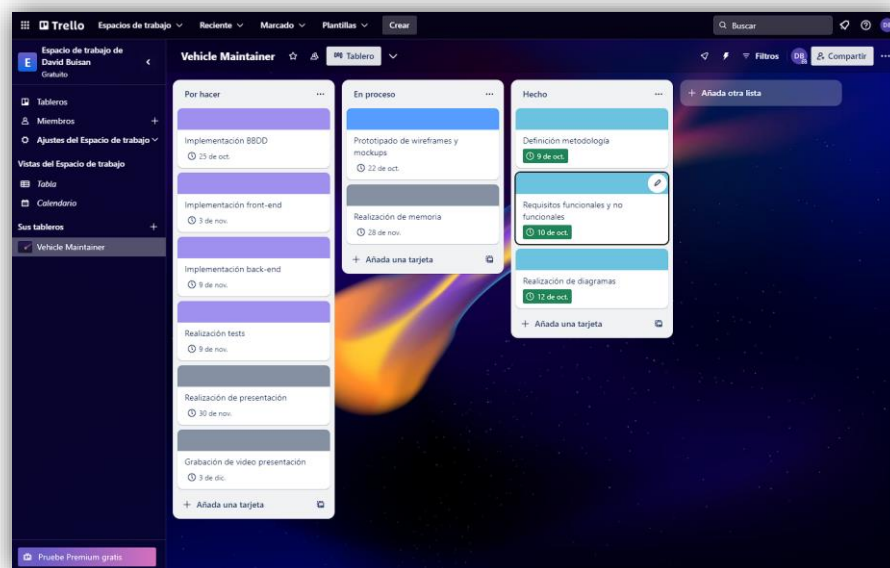


Imagen 2. Tablero de la aplicación Vehicle Maintainer. Trello

## 4. Tecnologías y herramientas utilizadas en el proyecto (2-3 páginas)

En este apartado se describen las herramientas y tecnologías utilizadas en el desarrollo de la aplicación, así como las causas principales de la elección de las mismas.

### Trello

Como ya se detalló en el apartado anterior, elegí la aplicación Trello para la gestión del tiempo y tareas para el desarrollo de la aplicación utilizando la metodología ágil Kanban.

La elección de esta herramienta fue debido a la popularidad de la misma y a la posibilidad de tener unificado tanto en mi equipo de sobremesa como en mi dispositivo móvil la gestión del proyecto en todo momento.

Además de ser una aplicación gratuita, es muy personalizable y fácil de utilizar.

### Canva

Para la realización del diagrama de Gantt, el diseño de los Mockups de la aplicación, la realización de los iconos, la elaboración del logo y el diseño final la interfaz gráfica de usuario se ha utilizado Canva [\[10\]](#).

Canva es una plataforma online para el diseño gráfico que descubrí en el módulo de programación multimedia y dispositivos móviles y la decantación por su elección fue gracias a que es muy intuitiva y potente a la hora de trabajar con ella, además permite obtener unos muy buenos resultados sin una experiencia amplia en el desarrollo gráfico.

También, decidí seleccionar Canva porque desde que supe de esta herramienta y he ido utilizándola en algunas ocasiones me siento muy cómodo trabajando con ella.

Esta aplicación es gratuita y solo requiere de un registro, además proporciona almacenamiento online de los proyectos realizados.

### Draw.io

La herramienta seleccionada para el diseño y realización del diagrama Entidad-Relación, el diagrama de Casos de Uso y el diagrama de clases ha sido Draw.io [\[11\]](#).

Draw.io es un software online de dibujo gráfico destinado a la realización de todo tipo de diagramas que descubrí mientras buscaba una herramienta para la realización de la PAC de desarrollo de la asignatura de Bases de Datos A en la cual teníamos que realizar un diagrama Entidad-Relación.

Después de probar distintos softwares para la implementación de diagramas, esta aplicación ofrecía una gran variedad de herramientas para el modelado de UML siendo una herramienta muy completa. Además, es una herramienta muy gráfica e intuitiva y resulta fácil trabajar con ella.

Como la aplicación anterior, Canva es gratuita con registro previo y también proporciona almacenamiento online de los proyectos realizados.

## Android Studio

Entre todos los IDE disponibles para el desarrollo de software me decanté por Android Studio [\[12\]](#) debido a que la aplicación a desarrollar tenía como objetivo ser para dispositivos móviles con sistema operativo Android siendo Android Studio la mejor opción para el desarrollo de una aplicación para Android.

También me decante por este IDE al haber podido practicar con él en el módulo de Programación multimedia y dispositivos móviles y por querer aprender más en profundidad sobre este entorno de desarrollo.

## Kotlin

Como lenguaje de programación, habiendo elegido Android Studio como IDE, tenía la opción de elegir entre Java [\[13\]](#) o Kotlin [\[14\]](#), decantándome por esta última debido a que Kotlin está bien integrado con Android Studio siendo este el lenguaje que recomienda. Además, tenía ganas de aprender más en profundidad sobre este lenguaje ya que solo pudimos ver unas pinceladas en el módulo de Programación multimedia y dispositivos móviles y me parece muy potente y fácil de aprender.



## SQLite

Para la persistencia de los datos gestionados por la aplicación, también tenía varias opciones como son SQLite [\[15\]](#) o Firebase [\[16\]](#).

Me decanté por SQLite ya que este tipo de base de datos está integrada en Android Studio. Además, quería una base de datos relacional que fuera sencilla de configurar y tampoco necesitaba una base de datos en la nube como en el caso Firebase

Al ser una base de datos relacional embebida, fácil de implementar y habiendo podido trabajar con ella con anterioridad, fueron estos aspectos que hicieron decantarme por esta base de datos.

## Git

Para el control de versiones se ha utilizado Git [\[17\]](#).

Me decidí utilizar Git básicamente por ser el control de versiones con mayor popularidad.

Anteriormente había indagado sobre Git y su manera de funcionar y me sorprendió lo bien integrado que está con Android Studio siendo muy fácil de trabajar con él.

Además, ofrece la posibilidad de subir tu proyecto a GitHub y así poder tener un respaldo en la nube, por si fuera necesario, o para poder trabajar en el proyecto desde distintos dispositivos.

## 5. Planificación, Diagnóstico y Contexto Laboral (3-4 páginas)

En este apartado se describe cómo ha sido la planificación para el desarrollo de la aplicación, haciendo una estimación inicial del tiempo y reparto de tareas mediante la realización del diagrama de Gantt. También recoge la comparativa con el tiempo real que ha sido empleado y la realización del análisis DAFO de nuestra aplicación

### Planificación inicial

En primer lugar, para hacer un reparto de tareas y poder organizar el tiempo empleado para el desarrollo del proyecto, a principios del mes de octubre de 2023, se realizó un diagrama de Gantt con el reparto de tareas en el tiempo estimado para el desarrollo.

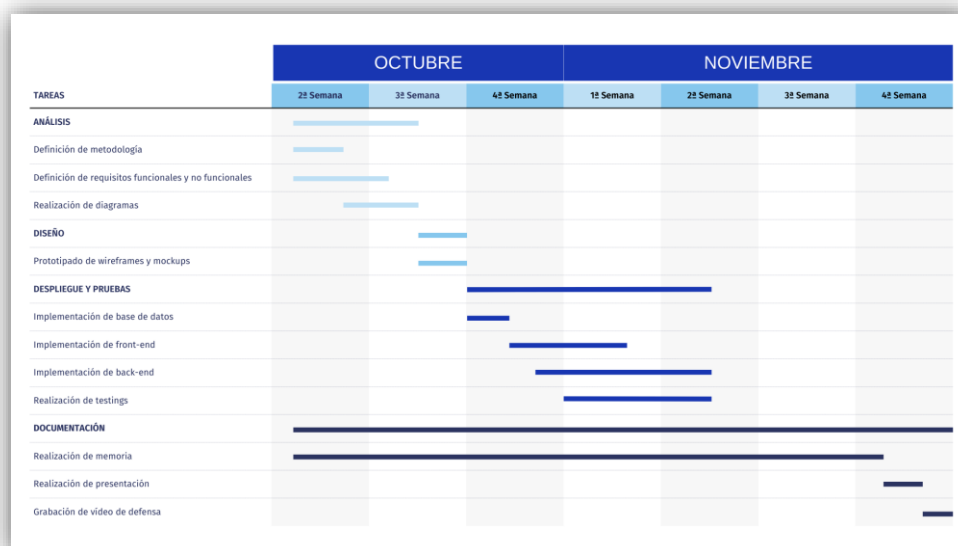


Imagen 3. Diagrama de Ganntt inicial. Canva

Como se puede apreciar en la imagen anterior, el diagrama muestra una fecha inicial estimada que empieza en la segunda semana de octubre, fecha en la que fue prevista la aceptación de la propuesta de proyecto.

Desde la fecha de inicio hasta finales de la tercera semana de octubre están repartidas las fases de análisis y diseño del proyecto.

Al inicio de la cuarta semana de octubre, se prevé que inicie la fase de despliegue y pruebas de la aplicación, finalizando dicha fase hacia mediados de la segunda semana de noviembre.

Por último, desde la fecha de inicio hasta finales de la cuarta semana de noviembre, se realiza la fase de documentación, fase que se solapa entre las demás etapas del desarrollo puesto que la elaboración de la memoria escrita será realizada en paralelo a todas las fases que componen el desarrollo de la aplicación.

## Planificación final

Tras la finalización del proyecto, modifiqué el diagrama de Gantt realizado anteriormente y cambié el color de las barras de progresión de las tareas realizadas. Apliqué el color verde a las tareas que se habían llevado a cabo en el tiempo estimado o antes del tiempo estimado y apliqué el color rojo a las tareas que me llevaron más tiempo del estimado.

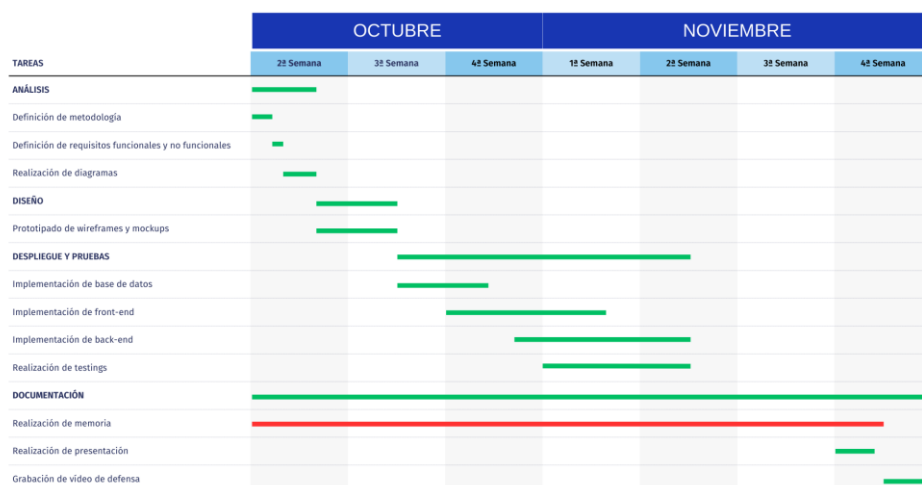


Imagen 4. Diagrama de Gantt final. Canva

Como se puede apreciar en el diagrama, únicamente la tarea que me llevó más tiempo del estimado fue la realización de la memoria escrita.

## Análisis DAFO

A continuación, se evalúa el proyecto realizando un análisis DAFO de la aplicación Vehicle Maintainer.

Este análisis recoge las debilidades, amenazas, fortalezas y oportunidades para ver la viabilidad del proyecto desarrollado.

Debilidades	Amenazas
<ul style="list-style-type: none"> <li>• Funcionalidad limitada.</li> <li>• Recursos limitados.</li> <li>• Poca experiencia en el sector.</li> </ul>	<ul style="list-style-type: none"> <li>• Existencia de aplicaciones bien valoradas dedicadas a la gestión de vehículos.</li> <li>• Mercado en constante evolución.</li> <li>• Gran abanico de soluciones presentes en el mercado.</li> </ul>
Fortalezas	Oportunidades
<ul style="list-style-type: none"> <li>• Aplicación muy intuitiva.</li> <li>• Facilidad de uso.</li> <li>• Se centra en una funcionalidad concreta.</li> </ul>	<ul style="list-style-type: none"> <li>• La dependencia de vehículos de multitud de personas.</li> <li>• Necesidad de realizar el mantenimiento de vehículos.</li> <li>• La mayoría de las personas posee un dispositivo móvil.</li> </ul>

## 6. Análisis del proyecto (8-16 páginas)

En este apartado se recogen los requisitos funcionales y no funcionales que se han detectado como indispensables a la hora de desarrollar la aplicación Vehicle Maintainer. También, se exponen el conjunto de los diagramas realizados junto con sus especificaciones.

Para dicho cometido, se hace uso de UML para documentar y modelar gráficamente el sistema, así como la interacción del usuario con el mismo y viceversa.

Este análisis recoge las necesidades de los usuarios y los requisitos del sistema. Esto garantizara un punto de partida para el correcto diseño y desarrollo de la aplicación.

### Requisitos funcionales y no funcionales

A continuación, se presenta el conjunto de requisitos funcionales y no funcionales recogidos con una descripción de los mismos.

### Requisitos Funcionales

**Registrar vehículo:** Los usuarios han de poder registrar vehículos indicando la información de los mismos como marca, modelo, número de matrícula, kilometraje actual y demás datos de la ficha técnica.

**Consultar listado de vehículos:** Los usuarios deben poder consultar un listado de todos los vehículos que tenga registrados.

**Consultar ficha técnica:** Los usuarios han de poder consultar los datos de la ficha técnica de los vehículos que tenga registrados.

**Actualizar kilometraje:** Los usuarios deben poder actualizar el kilometraje de los vehículos que tenga registrados.

**Programar mantenimiento:** Los usuarios deben poder programar el mantenimiento de sus vehículos, ya sea por fecha como por kilometraje.

**Notificaciones automáticas:** La aplicación ha de enviar notificaciones automáticas a los usuarios cuando se acerque la fecha o el kilometraje en referencia a los mantenimientos programados.

Recibir, consultar y manipular mensajes: Los usuarios deben poder recibir, consultar y manipular los mensajes, pudiendo marcar como leído o borrarlos.

Consultar listado de mantenimientos programados: Los usuarios han de poder consultar un listado de todos los mantenimientos programados.

Registrar mantenimiento de vehículo: Los usuarios deben poder registrar los mantenimientos realizados a los vehículos indicando la información como descripción, fecha, kilometraje, importe y tipo.

Consultar listado de mantenimientos realizados: Los usuarios han de poder consultar un listado de todos los mantenimientos realizados.

## Requisitos no funcionales

Usabilidad: La aplicación ha de ofrecer una interfaz intuitiva y fácil de usar, sin importar la experiencia previa del usuario.

Rendimiento: La aplicación ha de ser eficiente en cuanto a velocidad de consulta e inserción de datos se refiere para que el usuario tenga una buena experiencia.

Gestión de excepciones: La aplicación ha de tener una buena gestión de excepciones que puedan surgir de los datos introducidos por el usuario para así evitar cierres inoportunos. Además, ha de mantener informado al usuario.

Escalabilidad: La aplicación debe poder crecer en cuanto a volumen de vehículos como de mantenimientos realizados sin que afecte significativamente su rendimiento. También, ha de poder permitir la incorporación de funcionalidades futuras.

Compatibilidad: La aplicación ha de ser compatible con el mayor número de dispositivos y versiones de Android.

## Modelo Entidad-Relación

Continuando con el análisis del sistema, se presenta el diagrama Entidad-Relación realizado para la representación de la BD que utilizara el sistema.

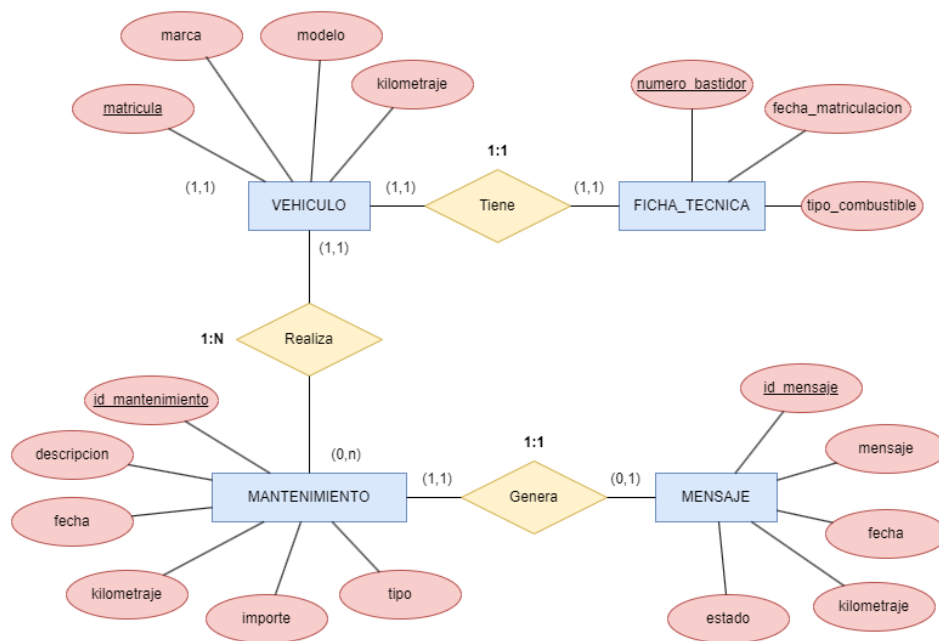


Imagen 5. Diagrama Entidad-Relación

Tras la realización del diagrama Entidad-Relación se construye el modelo relacional resultante del análisis del diagrama, dando como resultado la totalidad de tres tablas a construir en la BD.

La primera tabla resultante es VEHICULO, que es el resultado de la unificación de la entidad VEHICULO y FICHA\_TECNICA, los atributos de FICHA\_TECNICA pasaran a ser atributos de la tabla VEHICULO ya que su cardinalidad es de 1:1 y sus participaciones de (1,1) y (1,1).

- VEHICULO (matricula, numero\_bastidor, marca, modelo, fecha\_matriculacion, kilometraje, tipo\_combustible)
  - **matricula**: PRIMARY KEY de la tabla VEHICULO

La segunda tabla resultante es MANTENIMIENTO, que fruto de la relación que tienen las entidades VEHICULO y MANTENIMIENTO, dado que su cardinalidad es de 1:N con participaciones de (1,1) y (0, n), hereda como clave ajena la clave primaria de la entidad VEHICULO.

- MANTENIMIENTO (id\_mantenimiento, descripcion, fecha, kilometraje, importe, tipo, matricula)
  - **id\_mantenimiento**: PRIMARY KEY de la tabla MANTENIMIENTO
  - **matricula**: FOREIGN KEY de la tabla VEHICULO (matricula)

La última tabla restante es MENSAJE, ya que la relación que tienen las entidades MANTENIMIENTO y MENSAJE, dando como resultado una cardinalidad 1:1 y participaciones de (1,1) y (0,1), hereda como clave ajena la clave primaria de la entidad MANTENIMIENTO.

- MENSAJE (id\_mensaje, mensaje, fecha, kilometraje, estado, id\_mantenimiento)
  - **id\_mensaje:** PRIMARY KEY de la tabla NOTIFICACION
  - **id\_mantenimiento:** FOREIGN KEY de la tabla MANTENIMIENTO (id\_mantenimiento)

Llevada a cabo la realización del modelo relacional se procede a comprobar que la normalización del modelo sea la adecuada.

Tras el análisis del modelo y de los requisitos que debe cumplir, se llega a la conclusión que el modelo relacional resultante es adecuado, pudiendo asegurar que se ha llegado a la normalización en 3ª forma normal, normalización óptima y suficiente de las tablas.

## Casos de uso

En este apartado toca analizar cómo el usuario interactúa con el sistema, para ello se realiza el diagrama de Casos de Uso acompañado del conjunto de sus especificaciones para cada caso.

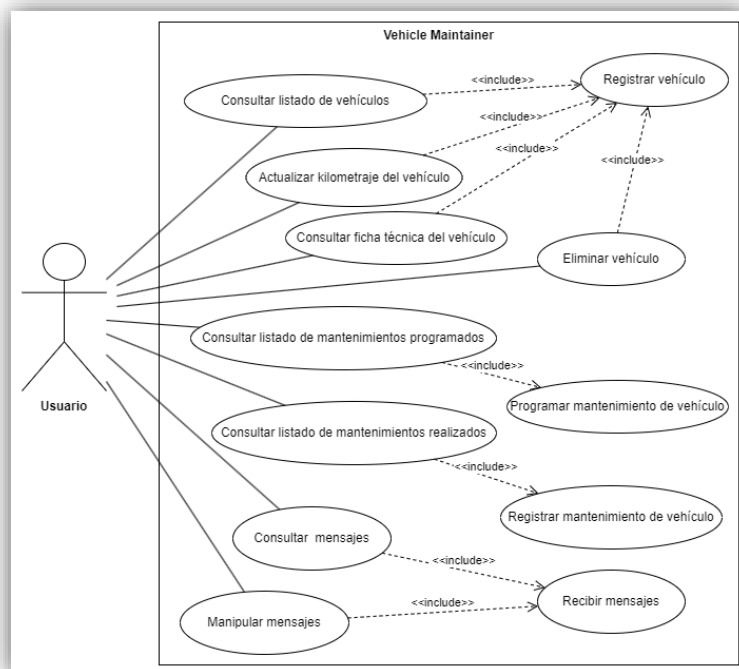




Imagen 6. Diagrama de Casos de Uso

Identificador de Caso Uso	CU_001
Nombre	Registrar vehículo
Descripción	Permite al usuario registrar vehículo
Actores	Usuario
Secuencia normal	
Actor	Software
1. El usuario pulsa el botón de crear vehículo.	
	2. El sistema muestra una pantalla para ingresar los datos.
3. El usuario inserta los datos.	
4. El usuario pulsa el botón de guardar.	
	5. El sistema comprueba los datos.
	6. Si los datos son correctos, el sistema los almacena en la BD.
	7. El sistema muestra mensaje al usuario.
	8. El sistema retorna a la pantalla principal.
Excepciones	Software
Datos introducidos incorrectos.	6. Si los datos no son correctos, el sistema muestra un mensaje para informar al usuario.
<b>CU relacionados</b>	N/A
Precondición	N/A
Postcondición	Si los datos introducidos son correctos, quedan guardados los datos del vehículo.
Secuencia alternativa	
Actor	Software

4. El usuario pulsa el botón de cancelar.	
	5. El sistema retorna a la pantalla de inicio.

Tabla 4. Caso de Uso registrar vehículo

Identificador de Caso Uso	CU_002
Nombre	Eliminar vehículo
Descripción	Permite eliminar el registro de un vehículo.
Actores	Usuario
Secuencia normal	
Actor	Software
1. El usuario pulsa el botón de eliminar vehículo.	
	2. El sistema muestra un mensaje de confirmación.
3. El usuario pulsa aceptar.	
	4. El sistema borra el registro del vehículo.
	5. El sistema retorna a la pantalla de inicio.
Excepciones	Software
N/A	
CU relacionados	CU_001
Precondición	Tienen que haber registrado como mínimo un vehículo.
Postcondición	Si existen mensajes, el sistema muestra un indicador para advertir de la existencia de mensajes nuevos.
Secuencia alternativa	
Actor	Software
3. El usuario pulsa el botón de cancelar.	

	4. El sistema retorna a la pantalla de inicio.
--	--

Tabla 5. Caso de Uso eliminar vehículo

Identificador de Caso Uso	CU_003
Nombre	Consultar listado de vehículos
Descripción	Permite ver un listado de todos los vehículos creados.
Actores	Usuario
Secuencia normal	
Actor	Software
1. El usuario entra en la aplicación.	
	2. El sistema comprueba si existen registros de vehículos.
	3. El sistema muestra un listado con todos los vehículos registrados.
Excepciones	Software
No existen datos de vehículos	3. El sistema no muestra ningún listado.
<b>CU relacionados</b>	CU_001
Precondición	Tiene que haber como mínimo un vehículo registrado.
Postcondición	Si existen vehículos registrados, el sistema muestra una lista de todos los vehículos registrados.

Tabla 6. Caso de Uso consultar listado de vehículo

Identificador de Caso Uso	CU_004
Nombre	Consultar ficha técnica del vehículo
Descripción	Permite ver los datos de la ficha técnica del vehículo
Actores	Usuario

Secuencia normal	
Actor	Software
1. El usuario pulsa el botón de ver ficha técnica.	
	2. El sistema muestra una pantalla con todos los datos del vehículo.
	3. El sistema muestra un listado con todos los vehículos registrados.
Excepciones	Software
N/A	
<b>CU relacionados</b>	CU_001
Precondición	Tiene que haber como mínimo un vehículo registrado.
Postcondición	El sistema muestra los datos de la ficha técnica del vehículo.

Tabla 7. Caso de Uso consultar ficha técnica

Identificador de Caso Uso	CU_005
Nombre	Actualizar kilometraje del vehículo
Descripción	Permite actualizar el kilometraje del vehículo.
Actores	Usuario
Secuencia normal	
Actor	Software
1. El usuario pulsa el botón de actualizar kilometraje.	
	2. El sistema muestra una pantalla para actualizar los datos.
3. El usuario introduce los datos.	
4. El usuario presiona el botón de guardar datos.	
	5. El sistema comprueba que los datos introducidos sean correctos.

	6. Si los datos son correctos, el sistema guarda los datos introducidos en la BD.
	7. El sistema muestra mensaje al usuario.
	8. El sistema retorna a la pantalla principal.
Excepciones	Software
Datos introducidos incorrectos.	6. Si los datos no son correctos, el sistema muestra un mensaje para informar al usuario.
<b>CU relacionados</b>	CU_001
Precondición	Tiene que haber como mínimo un vehículo registrado.
Postcondición	Si los datos introducidos son correctos, el sistema actualiza los datos del kilometraje.
Secuencia alternativa	
Actor	Software
5. El usuario pulsa el botón de cancelar.	
	6. El sistema retorna a la pantalla de inicio.

Tabla 8. Caso de Uso actualizar kilometraje de vehículo

Identificador de Caso Uso	CU_006
Nombre	Programar mantenimiento de vehículo
Descripción	Permite programar el mantenimiento para un vehículo
Actores	Usuario
Secuencia normal	
Actor	Software
1. El usuario presiona el botón de programar mantenimiento.	

	2. El sistema pregunta al usuario cómo desea programar el mantenimiento, por fecha o por kilometraje.
3. El usuario selecciona programar mantenimiento por fecha.	
	4. El sistema muestra una pantalla para que el usuario seleccione la fecha deseada.
5. El usuario selecciona la fecha.	
	6. El sistema guarda los datos del mantenimiento en la BD.
	7. El sistema programa la notificación para recordar la fecha.
	8. El sistema guarda los datos de la notificación en la BD para utilizarla posteriormente.
	9. El sistema retorna a la pantalla principal.
Excepciones	Software
N/A	
<b>CU relacionados</b>	CU_001
Precondición	Tiene que haber como mínimo un vehículo registrado.
Postcondición	El sistema almacena los datos del mantenimiento y queda programada una notificación de recordatorio para el usuario.
<b>Secuencia alternativa 1</b>	
Actor	Software
3. El usuario presiona programar mantenimiento por kilometraje.	
	4. El sistema comprueba los datos introducidos.

	5. Si los datos son correctos, el sistema guarda los datos del mantenimiento en la BD.
	6. El sistema programa la notificación para recordar la fecha.
	7. El sistema guarda los datos de la notificación en la BD para utilizarla posteriormente.
	8. El sistema retorna a la pantalla principal.
Excepciones	Software
Datos introducidos incorrectos.	5. Si los datos no son correctos, el sistema muestra un mensaje para informar al usuario.
Secuencia alternativa 2	
Actor	Software
3. / 5. El usuario pulsa el botón de cancelar.	
	4. / 6. El sistema retorna a la pantalla de inicio.

Tabla 9. Caso de Uso programar mantenimiento de vehículo

Identificador de Caso Uso	CU_007
Nombre	Consultar listado de mantenimientos programados
Descripción	Permite ver un listado de todos los mantenimientos programados.
Actores	Usuario
Secuencia normal	
Actor	Software
1. El usuario pulsa el botón de listado de mantenimientos programados.	

	2. El sistema comprueba si existen registros de mantenimientos programados en la BD.
	3. Si existen datos de mantenimientos programados, el sistema muestra un listado con todos los mantenimientos programados.
<b>Excepciones</b>	<b>Software</b>
No existen datos de mantenimientos programados	3. Si no existen datos de mantenimientos realizados, el sistema no muestra ningún listado y muestra un mensaje informativo al usuario.
<b>CU relacionados</b>	CU_006
<b>Precondición</b>	Tiene que haber como mínimo un mantenimiento programado registrado.
<b>Postcondición</b>	Si existen mantenimientos programados, el sistema muestra una lista de todos los mantenimientos programados registrados.

Tabla 10. Caso de Uso consultar listado de mantenimientos programados

<b>Identificador de Caso Uso</b>	CU_008
<b>Nombre</b>	Registrar mantenimiento de vehículo
<b>Descripción</b>	Permite al usuario registrar un mantenimiento de vehículo realizado.
<b>Actores</b>	Usuario
<b>Secuencia normal</b>	
<b>Actor</b>	<b>Software</b>
1. El usuario pulsa el botón de registrar mantenimiento de vehículo.	
	2. El sistema muestra una pantalla para ingresar los datos.
3. El usuario inserta los datos.	



4. El usuario pulsa el botón de guardar.	
	5. El sistema comprueba los datos.
	6. Si los datos son correctos, el sistema los almacena en la BD.
	7. El sistema muestra mensaje al usuario.
	8. El sistema retorna a la pantalla principal.
<b>Excepciones</b>	<b>Software</b>
Datos introducidos incorrectos.	6. Si los datos no son correctos, el sistema muestra un mensaje para informar al usuario.
<b>CU relacionados</b>	CU_001
<b>Precondición</b>	Tiene que haber como mínimo un vehículo registrado.
<b>Postcondición</b>	Si los datos introducidos son correctos, quedan guardados los datos del mantenimiento de vehículo realizado.
<b>Secuencia alternativa</b>	
<b>Actor</b>	<b>Software</b>
4. El usuario pulsa el botón de cancelar.	
	5. El sistema retorna a la pantalla de inicio.

Tabla 11. Caso de Uso registrar mantenimiento de vehículo

<b>Identificador de Caso Uso</b>	CU_009
<b>Nombre</b>	Consultar listado de mantenimientos realizados
<b>Descripción</b>	Permite ver un listado de todos los mantenimientos realizados.
<b>Actores</b>	Usuario
<b>Secuencia normal</b>	

Actor	Software
1. El usuario pulsa el botón de listado de mantenimientos realizados.	
	2. El sistema comprueba si existen registros de mantenimientos realizados en la BD.
	3. Si existen datos de mantenimientos realizados, el sistema muestra un listado con todos los mantenimientos realizados.
Excepciones	Software
No existen datos de mantenimientos programados	3. Si no existen datos de mantenimientos realizados, el sistema no muestra ningún listado y muestra un mensaje informativo al usuario.
CU relacionados	CU_008
Precondición	Tiene que haber como mínimo un mantenimiento realizado registrado.
Postcondición	Si existen mantenimientos realizados, el sistema muestra una lista de todos los mantenimientos realizados registrados.

Tabla12. Caso de Uso consultar listado de mantenimientos realizados

Identificador de Caso Uso	CU_010
Nombre	Recibir mensajes
Descripción	Permite mostrar los mensajes recibidos.
Actores	Usuario
Secuencia normal	
Actor	Software
6. El usuario entra en la aplicación	
	7. El sistema comprueba si existen mensajes.

	8. Si existen mensajes, el sistema muestra un indicador para advertir de la existencia de mensajes nuevos.
Excepciones	Software
N/A	
<b>CU relacionados</b>	CU_006
Precondición	Tienen que haber registrado como mínimo un mantenimiento de vehículo.
Postcondición	Si existen mensajes, el sistema muestra un indicador para advertir de la existencia de mensajes nuevos.

Tabla 13. Caso de Uso recibir mensajes

Identificador de Caso Uso	CU_011
Nombre	Consultar mensajes
Descripción	Permite mostrar un listado de todos los mensajes.
Actores	Usuario
Secuencia normal	
Actor	Software
1. El usuario pulsa el botón de consultar mensajes.	
	2. El sistema comprueba si existen registros de mensajes.
	3. Si existen registros de mensajes, el sistema muestra una nueva pantalla con el listado de mensajes.
Excepciones	Software
No existen datos de mensajes	4. Si no existen datos de mensajes, el sistema no muestra ningún listado y muestra un mensaje informativo al usuario.

<b>CU relacionados</b>	CU_010
Precondición	Tienen que haber registrado como mínimo un mensaje.
Postcondición	Si existen mensajes, el sistema muestra un indicador para advertir de la existencia de mensajes nuevos.

Tabla 14. Caso de Uso consultar mensajes

Identificador de Caso Uso	CU_012
Nombre	Manipular mensajes
Descripción	Permite al usuario manipular los mensajes pudiendo marcar como leídos y pudiendo borrarlos.
Actores	Usuario
Secuencia normal	
Actor	Software
1. El usuario pulsa el botón de borrar mensaje.	
	2. El sistema muestra un mensaje de confirmación.
3. El usuario pulsa aceptar.	
	4. El sistema borra el registro del mensaje.
	5. El sistema retorna a la pantalla principal.
Excepciones	Software
N/A	
<b>CU relacionados</b>	CU_010
Precondición	Tienen que haber registrado como mínimo un mensaje.
Postcondición	Quedan borrados los datos del mensaje.
Secuencia alternativa 1	
Actor	Software

1. El usuario pulsa el botón de cancelar.	
	2. El sistema retorna a la pantalla de inicio.
Secuencia alternativa 2	
Actor	Software
1. El usuario pulsa el botón de marcar como leído.	
	2. El sistema cambia el estado del mensaje.

Tabla 15. Caso de Uso manipular mensajes

## Diagrama de clases

Para finalizar con la fase de análisis, se realiza el diagrama de clases, parte fundamental para representar la estructura general del sistema. Este diagrama representa la estructura general de las clases y sus relaciones facilitando la comprensión de la arquitectura y el desarrollo de la aplicación.

En primera instancia, el diagrama constaba de 5 clases, la clase Garaje, la clase Vehículo, la clase FichaTecnica, la clase Mantenimiento y la clase Mensaje, pero en la fase de desarrollo se optó por la simplificación de las mismas para facilitar la persistencia de los datos en la BD dando como resultado la unión de Vehículo y FichaTecnica y la eliminación de la clase Garaje como tal pasando a ser una Activity llamada GarajeActivity.

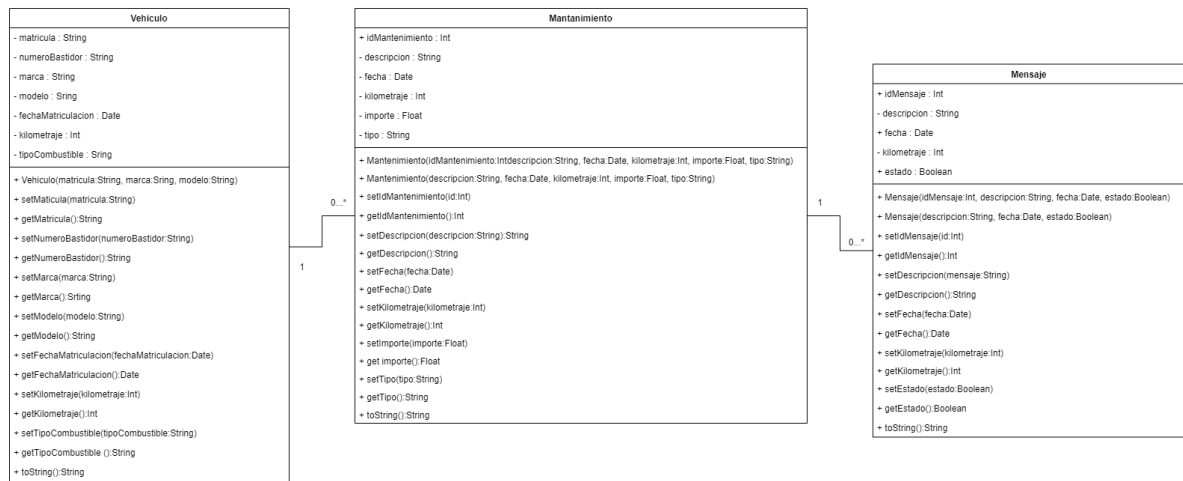


Imagen 7. Diagrama de clases final

## 7. Diseño del proyecto (6-14 páginas)

En este apartado se describe el proceso realizado en cuanto al diseño de la aplicación se refiere desde la elección de la paleta de colores hasta el diseño de los distintos layouts que serán parte de la interfaz de usuario.

### Wireframes

Para la representación inicial del diseño de la interfaz gráfica de usuario, se comenzó por realizar un conjunto de wireframes con la intención de ver de manera sencilla como sería cada uno de los distintos layouts que utilizara la aplicación.

Apoyándome en la fase anterior de análisis, donde fueron elaborados los Casos de Uso y el diagrama de clases entre otros, comencé por diseñar la pantalla principal de la aplicación la cual sería el nexo de unión de las demás pantallas.

La elaboración del conjunto de wireframes será realizada mediante papel y lápiz, dando como resultado wireframes de baja fidelidad, que serán un buen recurso para realizar la disposición de elementos y poder visualizar de manera orientativa las distintas pantallas de la aplicación.

En primera instancia elabore la pantalla que se mostrara al inicio de la aplicación. Esta pantalla contendrá un menú en la barra superior con tres botones y el resto de pantalla mostrará un listado con el conjunto de vehículos creados por el usuario.

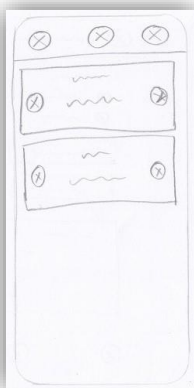


Imagen 8. Wireframe pantalla principal

El primer botón del menú, situado más a la izquierda, abrirá una nueva pantalla que mostrará un listado de todos los mantenimientos programados.

El segundo botón, situado en el medio del menú, abrirá una nueva pantalla para añadir un vehículo.

El tercer botón, situado más a la derecha, abrirá una nueva pantalla que mostrará el listado de mensajes.

En la pantalla aparecerá una tarjeta por cada vehículo que tenga registrado el usuario, en ella se mostraran datos del vehículo como marca, modelo y matrícula, además contendrá dos botones, uno situado a la izquierda de la tarjeta para eliminar el vehículo y otro situado a la derecha para acceder a todos los datos del vehículo.

Seguidamente, realice el wireframe para la pantalla de registrar los datos de vehículo. La intención principal es mantener un aspecto y distribución de elementos que sea parecido en todo el conjunto de layouts de la aplicación intentando dar homogeneidad al diseño de las mismas.

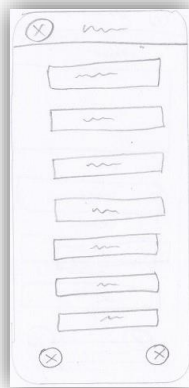


Imagen 9. Wireframe pantalla registrar nuevo vehículo

Esta pantalla estará conformada de un menú en la barra superior que contará con una descripción de la pantalla, en este caso registrar nuevo vehículo, y un botón que regrese a la pantalla anterior.

La pantalla principal de este layout mostrará los campos disponibles para que el usuario pueda registrar el vehículo tales como marca, modelo, matrícula, kilometraje, fecha de matriculación, tipo de combustible y número de bastidor.



También, tendrá en la parte inferior de la pantalla dos botones, uno para rechazar la inserción de los datos y otro para aceptar.

A continuación, fue realizado el wireframe de la pantalla vehículo que será la encargada de gestionar lo relevante a el vehículo seleccionado.

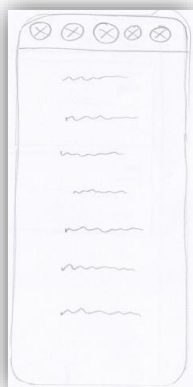


Imagen 10. Wireframe pantalla vehículo

Esta pantalla será la encargada de mostrar todos los datos que contiene el vehículo seleccionado.

Además, tendrá el menú en la parte superior que contendrá cinco botones. El primer botón, el más situado a la izquierda, será para volver a la pantalla principal, el segundo botón será el encargado de programar un mantenimiento para el vehículo, el tercer botón, el central, tendrá como función actualizar el kilometraje del vehículo, el cuarto botón se encarga de mostrar el listado de mantenimientos programados y el quinto botón se encargará de mostrar una pantalla para agregar un mantenimiento realizado.

El paso siguiente fue realizar el wireframe para la pantalla encargada de programar un mantenimiento para el vehículo.

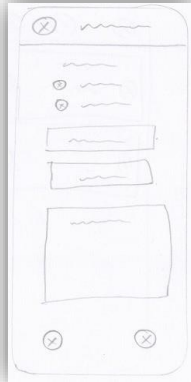


Imagen 11. Wireframe pantalla programar nuevo mantenimiento

Esta pantalla consta de dos RadioButtons para poder elegir si se desea programar el mantenimiento por fecha o por kilometraje.

Contará con tres apartados para insertar datos. Uno para elegir la fecha deseada, otro para indicar kilometraje y el último para añadir una descripción del mantenimiento.

Además, contendrá dos botones en la parte inferior, uno para cancelar la programación del mantenimiento y otro para aceptar.

El siguiente wireframe realizado fue para la pantalla encargada de registrar un mantenimiento realizado.

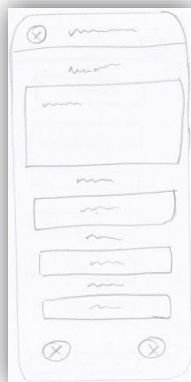


Imagen 12. Wireframe pantalla registrar nuevo mantenimiento

Esta pantalla posee cuatro campos para insertar datos del mantenimiento realizado. El primer campo se encarga de recoger datos de la descripción del mantenimiento, el segundo campo para indicar la fecha, el tercer campo se encarga de recoger el kilometraje y el cuarto campo recoge los datos del importe.

La pantalla contendrá dos botones en la parte inferior, uno para rechazar la inserción de los datos y otro para aceptar.

También, contendrá el menú en la barra superior con un botón para volver a la pantalla principal y una etiqueta descriptiva de la funcionalidad.

Para finalizar con la realización de los wireframes, diseñe una pantalla que sería común a todas las funcionalidades de mostrar listas.

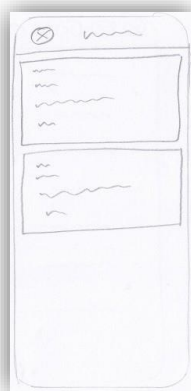


Imagen 13. Wireframe para pantallas que muestren listados

Esta será una pantalla que muestre un listado de elementos según sea conveniente, lista de mensajes, lista de mantenimientos programados y lista de mantenimientos realizados.

También contendrá el menú superior con la etiqueta descriptiva y un botón para volver a la pantalla principal.

## Paleta de colores

Una vez realizado el conjunto de wireframes para cada distinto layout de la aplicación se procede a la elección de la paleta de colores que tendrá la interfaz de usuario.

Para la elección de la paleta de colores me decante por utilizar un conjunto de tonalidades frías.

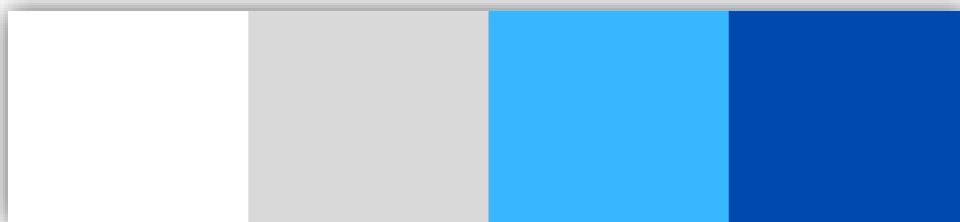


Imagen 14. Paleta de colores utilizada

Los colores que componen el conjunto de elementos de la interfaz de usuario son blanco, gris claro, azul claro y azul oscuro. También se hará uso de rojo y verde para botones de cancelar o aceptar.

## Mockups

Tras la elección de la paleta de colores que utilizará la aplicación y apoyándome en los wireframes elaborados con anterioridad, llegó el momento de realizar el maquetado de los distintos layouts.

En primer lugar, se realizó la pantalla principal de la aplicación. Esta pantalla estará relacionada con la clase Garaje que se realizó en la fase de análisis. Además, está relacionada con el Caso de Uso consultar listado de vehículo y eliminar vehículo.

Esta será la encargada de recuperar los objetos de tipo Vehículo de la base de datos.

Siempre que se inicie esta pantalla, el sistema comprobará si existen vehículos registrados en la base de datos, y si es así, cargara cada vehículo y los mostrará en la pantalla.

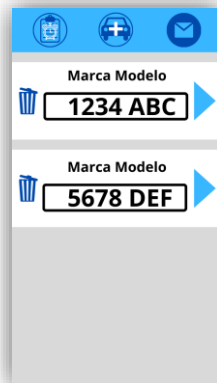


Imagen 15. Mockup pantalla principal

La imagen anterior muestra el diseño final de la pantalla principal de la aplicación.

Seguidamente, se realizó el diseño de la pantalla que aparecerá al pulsar el botón del medio del menú superior, la pantalla de registrar vehículo.

Esta pantalla se relaciona con las clases Vehículo y FichaTecnica de la fase de análisis y es la encargada de registrar vehículos en la base de datos. Además, se relaciona con el Caso de Uso registrar vehículo.

Aquí el usuario deberá introducir los datos pertenecientes al vehículo que desea registrar pudiendo cancelar o aceptar la operación.

Para poder registrar el vehículo será necesario que se rellenen todos los campos y que los tipos de datos sean correctos.

Tras comprobar los datos introducidos, estos se almacenarán en la base en la entidad VEHICULO del diagrama Entidad-Relación especificado en la fase de análisis.

Una vez realizada la operación, el sistema retornará a la pantalla de inicio y ahí se podrá ver el vehículo registrado.

Imagen 16. Mockup pantalla registrar nuevo vehículo

A continuación, se realizó el diseño de la pantalla que aparecerá al pulsar el botón para acceder a vehículo.

Esta pantalla muestra todos los datos del vehículo y, como en el caso de la pantalla anterior está relacionada con las clases Vehículo y FichaTecnica.

Por otra parte, está relacionada con los Casos de Uso consultar ficha técnica de vehículo y actualizar kilometraje del vehículo, todos ellos elaborados en la fase de análisis.

Esta pantalla se encargará de gestionar un vehículo en particular, pudiendo acceder a la pantalla de programar nuevo mantenimiento, actualizar el kilometraje del vehículo, acceder a la pantalla para ver la lista de mantenimientos realizados y también para poder acceder a la pantalla de registrar nuevo mantenimiento realizado.

Imagen 17. Mockup pantalla vehículo

El siguiente diseño realizado fue el de la pantalla referente a programar nuevo mantenimiento.

Esta pantalla está relacionada con la clase Mantenimiento y el Caso de Uso programar mantenimiento de vehículo de la fase de análisis del proyecto.

Será la pantalla encargada de programar un mantenimiento para el vehículo pudiendo elegir entre realizar mantenimiento por fecha o por kilometraje.

Aquí el usuario debe elegir un tipo de mantenimiento e insertar los datos pertinentes, así como una descripción del mismo.

Estos datos se almacenarán en la base de datos y serán relacionados con la entidad MANTENIMIENTO del diagrama Entidad-Relación, con la particularidad de que el tipo de MANTENIMIENTO será almacenado como programado, para distinguir en posteriores búsquedas y poder filtrar por tipos de mantenimientos, ya sean del tipo realizado o programado. Además, está relacionada con la entidad NOTIFICACION ya que el sistema gestionara una notificación automática para cada mantenimiento programado.

Una vez realizada o cancelada la operación, el sistema retornara a la pantalla de inicio de la aplicación.

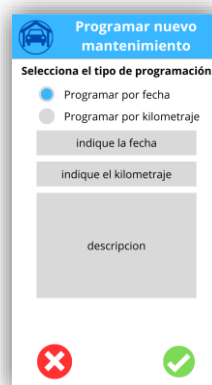


Imagen 18. Mockup pantalla programar nuevo mantenimiento

Continuando con el diseño se realiza el mockup de la pantalla destinada a registrar nuevo mantenimiento de vehículo.

Esta pantalla está relacionada con la clase Mantenimiento y con el Caso de Uso registrar mantenimiento de vehículo. Además, está relacionada con la entidad MANTENIMIENTO del diagrama Entidad-Relación.

Este será el lugar para que el usuario pueda insertar los datos de un mantenimiento realizado tales como descripción, fecha de realización, kilometraje e importe.

Estos datos sean almacenados en la entidad MANTENIMIENTO de la base de datos, siempre que el usuario pulse el botón de aceptar, aquí el tipo de MANTENIMIENTO será almacenado como realizado.

Al concluir o finalizar la tarea, el sistema retornará a la pantalla de inicio.



Imagen 12. Mockup pantalla registrar nuevo mantenimiento

A continuación, se realizó el diseño de la pantalla para mostrar la lista de mantenimientos realizados, la lista de mantenimientos programados y la pantalla para mostrar el listado de mensajes.

Todas estas pantallas son prácticamente iguales con la peculiaridad de que cambian los elementos que muestran.

En el caso de las pantallas para mostrar la lista de mantenimientos realizados y la lista de mantenimientos programados, ambas pantallas están relacionadas con la clase Mantenimiento y la entidad MANTENIMIENTO del diagrama Entidad-Relación.

También, están relacionadas con su Caso de Uso específico, uno con el Caso de Uso consultar listado de mantenimientos programados y el otro con el Caso de Uso consultar listado de mantenimientos realizados.



Por otro lado, la pantalla para mostrar los mensajes está relacionada con la clase Notificación, con la entidad NOTIFICACION y con los Casos de Uso consultar mensajes, manipular mensajes y recibir mensajes.



Imagen 13. Mockups de pantallas que muestren listados

## Logotipo

Para finalizar con la fase de diseño y darle a la aplicación un aspecto más personal, se realizó un diseño para el logotipo a mostrar, intentando que este representara de manera gráfica la funcionalidad de la aplicación.



Imagen 13. Logo Vehicle Maintainer

## 8. Despliegue y pruebas (4-10 páginas)

En el proceso de codificación del proyecto, lo primero que se implementó fue la clase llamada `MiSqlHelper`.

### `MiSqlHelper`

Esta clase hereda de la clase `SQLiteOpenHelper` y es la encargada de todo lo referente a la base de datos. Esta clase es la que será utilizada para la inserción, recuperación y eliminación de los datos de la aplicación.

A continuación, se implementaron las clases detalladas en el diagrama de clases realizado en la fase de análisis.

### `Vehiculo`

Esta clase se utiliza para registrar y recuperar los vehículos del usuario. La clase consta de propiedades `matricula`, `numeroBastidor`, `marca`, `modelo`, `fechaMatriculación`, `kilometraje` y `tipoCombustible`.

Se han generado los `Getters` y `Setters` de todas las propiedades de clase y las propiedades `matricula` y `numeroBastidor` se han programado para que, al insertar un dato, si este no es `null`, eliminan los espacios en blanco y pasan los caracteres a mayúsculas. Este proceso se ha realizado con la intención de evitar incoherencias y errores dado que la propiedad `matricula` es la PK de la entidad `VEHICULO` de la base de datos.

La clase tiene dos constructores, uno que recibe por parámetro todos los campos de la clase y otro que recibe tres campos, `matricula`, `marca` y `modelo`. Este último constructor será utilizado para mostrar el listado de vehículos almacenados en la aplicación.

Además, la clase implementa el método `toString`.

### `Mantenimiento`

Esta clase se utiliza para registrar y recuperar mantenimientos, ya sean de tipo realizado o de tipo programado.

La clase consta de las propiedades `idMantenimiento`, `descripcion`, `fecha`, `kilometraje`, `importe`, `tipo` y `matricula` con sus respectivos Getters y Setters.

Además, posee un companion object con dos variables constantes, `MANTENIMIENTO_PROGRAMADO` y `MANTENIMIENTO_REALIZADO`, que serán utilizadas a la hora de instanciar los objetos y para su posterior filtrado de datos, ya sea para la inserción o recuperación de los mismos.

Esta clase también posee dos constructores, uno que recibe todos los campos por parámetro y otro que recibe todos los campos menos `idMantenimiento`. Este último constructor se utilizará para insertar los datos en la BD, ya que esta será la encargada de asignar un `id_mantenimiento` automáticamente utilizando la cláusula `AUTOINCRAMENT`.

También, implementa el método `toString`.

## Mensaje

Esta clase es la encargada de registrar y recuperar mensajes.

La clase posee las propiedades `idMensaje`, `mensaje`, `fecha`, `kilometraje`, `estado`, `idMantenimiento` y sus respectivos Getters y Setters.

También, consta de un companion object con un par de variables constantes, `NO_LEIDO` Y `LEIDO`, que se utilizarán para controlar el estado de los mensajes.

Esta clase, como ocurre con la clase `Mantenimiento`, tiene dos constructores, uno que recibe todos los campos por parámetro y otro que recibe todos los campos menos `idMensaje`. Este último constructor será utilizado para insertar los datos en la BD, ya que esta será la encargada de asignar un `id_mensaje` automáticamente utilizando la cláusula `AUTOINCRAMENT`.

Como en el caso de las clases anteriormente citadas, también implementa el método `toString`.

## Notificacion

Esta clase es la que se encarga de crear una notificación.

Contiene un companion object con una variable constante `ID_NOTIFICACION` necesaria para lanzar la notificación.

Esta clase será utilizada a la hora de programar un mantenimiento por fecha en la aplicación.

Tras la implementación de las clases principales que serán utilizadas por la aplicación, se procede a la creación de las Activities.

## RegistrarVehiculoActivity

Esta Activity se encarga de registrar los vehículos en la aplicación, para ello el usuario tendrá que rellenar todos los campos facilitados.

Los campos a rellenar están restringidos por tipo de dato y numero de caracteres, en su archivo layout, para evitar errores y excepciones a la hora de introducir los datos en la BD.

La Activity consta de tres botones, uno en el menú superior para ir a la GarajeActivity, Activity que se creará posteriormente, un botón para cancelar y otro botón para aceptar el registro.

Cuando el usuario pulse el botón de aceptar, mediante su función `setOnClickListener`, se ejecutarán varias comprobaciones. Primero se comprobará que todos los campos tengan valor y si hay algún campo vacío entonces se mostrará un mensaje en pantalla para informar al usuario. Si todos los campos tienen valor, se comprueba, haciendo uso de la clase `MiSqlHelper`, que no exista un vehículo registrado con la misma matrícula. Si la matrícula ya existe se lo indico al usuario y si no existe un vehículo registrado con la misma matrícula, en ese caso, haciendo uso de la clase `Vehiculo`, se instancia un objeto y se procede a insertar los datos en la BD.

## GarajeActivity

Esta Activity es la que se mostrará al iniciar la aplicación y es la encargada de mostrar los vehículos registrados por el usuario mediante un `RecyclerView`.

Para poder mostrar los vehículos se han creado un nuevo paquete, llamado `adapter`, que almacenará las clases `Adapter` y `ViewHolder` que necesita `RecyclerView` para su funcionamiento. En él se han creado su clase `Adapter` llamada `VehiculoAdapter` y su `ViewHolder` con el nombre `VehiculoViewHolder`. También, se ha tenido que crear el layout necesario para mostrar los vehículos llamado `item_vehiculo.xml`.

Esta Activity consta de tres botones en la parte superior, uno para ir a la `MantenimientosProgramadosActivity`, otro para lanzar la `RegistrarVehiculoActivity` y el último para ir a `MensajeActivity`.

También se encarga de mostrar, si es que existen, un número de mensajes no leídos haciendo uso de las funciones que implementa, `seleccionarIdMantenimiento` y `seleccionarListaMensajesNoLeidos`.

## VehiculoActivity

Esta Activity se encarga de mostrar los datos de un vehículo seleccionado.

Además, desde esta Activity se podrá volver a `GarajeActivity`, programar un mantenimiento, actualizar el kilometraje, ver un listado de mantenimientos realizados y añadir un mantenimiento realizado, haciendo uso de los botones situados en el menú superior que serán los encargados de lanzar la Activity correspondiente.

## RegistrarMantenimientoActivity

Esta Activity se encarga de registrar un mantenimiento realizado a un vehículo específico.

En ella el usuario ha de rellenar los datos en los campos que ofrece y, como en el caso de `RegistrarVehiculoActivity`, posee mecanismos para restringir el tipo de dato y el número de caracteres en su archivo layout.

La Activity consta de tres botones, uno en el menú superior para ir a la `GarajeActivity`, un botón para cancelar y otro botón para aceptar el registro.

Una vez el usuario pulse el botón de aceptar, se comprueban que todos los campos tengan valor. Si existe algún campo vacío, se muestra un mensaje al usuario. En el caso que todos los campos tengan valor, se instancia un objeto `Mantenimiento` y se procede a su inserción en la BD.

## MantenimientosRealizadosActivity

Esta Activity se encarga de mostrar un listado de mantenimientos realizados haciendo uso de un `RecyclerView`.

Para tal fin se han creado las clases necesarias, su Adapter llamada `MantenimientosRealizadosAdapter` y su ViewHolder `MantenimientosRealizadosViewHolder`. También, se ha creado el layout necesario para mostrar los mantenimientos realizados llamado `item_mantenimiento_realizado.xml`.

Esta Activity consta de un botón en el menú superior para volver a `GarajeActivity` y una descripción de la misma.

### ActualizarKilometrajeActivity

Esta Activity se encarga de actualizar el kilometraje de un vehículo específico.

La Activity es lanzada a modo de dialogo, para ello se modificó el archivo `AndroidManifest`, y consta de un campo a rellenar y dos botones, aceptar y cancelar.

Para poder modificar un kilometraje el campo ha de tener un valor, si no es así se informa al usuario. Si en campo tiene valor, el número introducido no puede ser igual o menor al kilometraje existente. Si se cumplen las condiciones para actualizar el kilometraje, el nuevo valor se almacena en la BD y se comprueba la existencia de mensajes y así determinar si se debe mostrar una alerta al usuario haciendo uso de la clase `AlertDialog`.

### ProgramarMantenimientoActivity

Esta Activity es la encargada de programar los mantenimientos pudiendo elegir entre programar por fecha o por kilometraje.

Al iniciar la Activity se crea el canal necesario para las notificaciones dependiendo de la versión de Android haciendo uso de la función `crearCanal`.

La Activity consta de tres botones, uno en el menú superior para ir a la `GarajeActivity`, un botón para cancelar y otro botón para aceptar el registro.

Si el usuario presiona el botón para aceptar la programación, se comprueba que los campos necesarios tengan valor dependiendo de la opción elegida, por fecha o por kilometraje.

Si la opción elegida es por fecha y los campos tienen valor, se instancian los objetos `Mantenimiento` y `Mensaje` para insertarlos en la BD. Además, haciendo uso de la función `notificacion`, se programa la notificación en la fecha y hora seleccionada por el usuario.

Si la opción elegida es por kilometraje y los campos tienen valor, se instancian los objetos Mantenimiento y Mensaje para insertarlos en la BD.

La Activity también está dotada de mecanismos para restringir el tipo de dato y el número de caracteres en su archivo layout.

## MantenimientosProgramadosActivity

Esta Activity se encarga de mostrar un listado de todos mantenimientos programados haciendo uso de un RecyclerView.

Para tal fin se han creado las clases necesarias, su Adapter llamada MantenimientosProgramadosAdapter y su respectivo ViewHolder llamado MantenimientosProgramadosViewHolder. También, se ha creado el layout necesario para mostrar los mantenimientos programados llamado item\_mantenimiento\_programado.xml.

Esta Activity consta de un botón en el menú superior para volver a GarajeActivity y una descripción de la misma.

Los mantenimientos programados poseen la capacidad de ser eliminados pulsando el botón con el icono de una papelera, eliminando conjuntamente el mensaje correspondiente.

## MensajeActivity

Esta Activity se encarga de mostrar un listado de los mensajes recibidos mediante un RecyclerView.

Como en los casos anteriores, se han creado las clases necesarias, su Adapter llamada MensajeAdapter y su ViewHolder MensajeViewHolder. También, se ha creado el layout para mostrar los mensajes llamado item\_mensaje.xml.

Esta Activity consta de un botón en el menú superior para volver a GarajeActivity y una descripción de la misma.

Cada mensaje tiene la funcionalidad de poder marcarse como leído y de poder ser eliminado, eliminando conjuntamente el correspondiente mantenimiento programado.

## Pruebas

A continuación, se detallan el conjunto de pruebas realizadas en el despliegue de la aplicación. Para tal fin, se ha tomado como referencia el diagrama de Casos de Uso y los requisitos funcionales detallados con anterioridad.

### Registrar vehículo

Para esta prueba se realizó el registro de un vehículo utilizando la Activity RegistrarVehiculoActivity y posteriormente se comprobó su correcta inserción haciendo uso de la Activity GarajeActivity ya que esta es la encargada de seleccionar y mostrar los vehículos insertados en la BD. También, se ha intentado insertar dos vehículos con la misma matrícula para comprobar el correcto funcionamiento de la aplicación.

### Consultar listado de vehículos

En esta prueba se realizó la inserción de varios vehículos y se comprobó su correcta visualización en el RecyclerView de GarajeActivity. Además, se realizó la eliminación de un vehículo existente y se comprobó la correcta eliminación de todos los objetos relacionados con el mismo.

### Consultar ficha técnica

Dado que finalmente se optó por unificar las clases Vehiculo y FichaTecnica esta prueba ha pasado a ser consultar vehículo. Para realizar la prueba se ha hecho uso de la Activity VehiculoActivity que es la encargada de mostrar todos los datos pertenecientes a un vehículo en particular.

### Actualizar kilometraje

Esta prueba se ha realizado haciendo uso de la Activity ActualizarKilometrajeActivity. Para su realización se ha insertado un valor valido para comprobar su correcta inserción y también se ha intentado introducir un valor no valido para comprobar que los mecanismos de restricción funcionan correctamente.



## Programar mantenimiento

Para esta prueba se ha utilizado la Activity ProgramarMantenimientoActivity. Aquí se han programado varios mantenimientos. Uno de ellos en la fecha y hora de las pruebas, otro en una fecha y hora alejada a la actual y otro en un kilometraje específico.

## Notificaciones automáticas

Tras las pruebas realizadas en programar mantenimiento, se ha comprobado que se reciben las notificaciones en el día y hora programado. Además, haciendo uso de ActualizarKilometrajeActivity se ha comprobado que funcione correctamente el aviso en el caso de que exista un mantenimiento a realizar tras la actualización del kilometraje.

## Recibir, consultar y manipular mensajes

Una vez programados los mantenimientos, se comprueba que desde la Activity GarajeActivity se visualice si existe un mensaje nuevo a mostrar, cosa que se realiza mostrando un número con el número de mensajes sin leer al lado del botón de mensajes. Al acceder a MensajeActiviti se comprueba la correcta visualización de los mensajes que deben mostrarse. También, se comprueba el funcionamiento de marcar como leído un mensaje y comprobar que ya no se visualice en la pantalla principal GarajeActivity. Además, se realiza la eliminación de un mensaje y se comprueba que, conjuntamente, se elimine el mantenimiento programado en la Activity MantenimientosProgramadosActivity.

## Consultar listado de mantenimientos programados

Tras la programación de mantenimientos realizada anteriormente, se comprueba en la Activity MantenimientosProgramadosActivity la correcta visualización de todos los mantenimientos programados. Además, se comprueba el borrado de un mantenimiento y la eliminación pertinente del Mensaje relacionado con el mismo.

## Registrar mantenimiento de vehículo

Para esta prueba se realizó el registro de un mantenimiento realizado utilizando la Activity RegistrarMantenimientoActivity y posteriormente se comprobó su correcta inserción haciendo uso de la Activity MantenimientosRealizadosActivity ya que esta es la encargada de seleccionar y mostrar los mantenimientos en referencia a un vehículo insertados en la BD.

## Consultar listado de mantenimientos realizados

En esta prueba se realizó la inserción de varios mantenimientos realizados y se comprobó la correcta selección y visualización de los mismos en el RecyclerView de la Activity MantenimientosRealizadosActivity.

Además de todas las pruebas realizadas con anterioridad, se ha creado una clase llamada MiSqlHelperInstrumentedTest, en el paquete correspondiente a las pruebas instrumentales, donde se realizan un conjunto de pruebas relacionadas con las sentencias CRUD de la base de datos.

## 9. Conclusiones (1-2 páginas)

En el proceso de realización del proyecto ha sido todo un reto ya que no tenía mucha experiencia acerca de Android Studio y Kotlin, cosa que hizo que tuviera que estar consultando casi continuamente la documentación de Android Studio.

En primera instancia, viendo la carga de trabajo que tenía la realización del proyecto, me sentí algo abrumado y no confiaba en poder acabar con éxito el mismo, cosa que fue desapareciendo con el progreso realizado con el paso del tiempo.

En el proceso de desarrollo han surgido varias complicaciones que me han dado algún que otro quebradero de cabeza. Uno de los que más trabajo me dio e hizo que no pudiera avanzar con fluidez fue la nulidad de un valor recogido de la base de datos, ya que yo esperaba un valor NULL y por algún motivo me estaba asignando un 0, cosa que hizo que tuviera que rehacer el código varias veces hasta que, gracias a los mensajes de Log, me di cuenta de lo que pasaba, pude adaptar el código y seguí avanzando con normalidad en el proyecto.

Otra dificultad que apareció fue debida a la eliminación de los datos en cascada, cosa que en SQLite no está habilitada por defecto, por suerte, con un poco de investigación sobre el tema, pude solventar el problema.

Básicamente, el conjunto de inconvenientes surgidos en el desarrollo ha sido debido a la poca experiencia personal con el IDE y el lenguaje de programación Kotlin. Debido a estos inconvenientes y a la consulta de información casi continua en el transcurso del desarrollo del proyecto, me siento más cómodo con el uso de estas tecnologías.

Tras la realización del proyecto y analizando la aplicación final resultante, puedo afirmar que se han logrado alcanzar todos los objetivos que fueron propuestos en el inicio.

Finalmente, siento satisfacción personal por el trabajo realizado y por poder desarrollar yo solo una aplicación móvil funcional de principio a fin.

## 10. Vías futuras (1-2 páginas)

Una vez finalizado el proyecto, pese que la aplicación realizada cumple con toda la funcionalidad y se ha logrado alcanzar todos los objetivos propuestos, se hace mención de algunas mejoras que podrían ser aplicadas en el futuro para aportar mayor valor a la aplicación.

Sería interesante de añadir más detalles relacionados con el vehículo como podría ser la posibilidad de poder registrar la póliza de seguro e incluso algunos aspectos técnicos como la presión recomendada de los neumáticos dependiendo de la carga del mismo.

Otra funcionalidad que podría adquirir la aplicación sería la capacidad de tener un contador total de los gastos derivados de los mantenimientos realizados a los vehículos, para poder tener una visión global de gastos.

Con el fin de proteger los datos del usuario, sería bueno almacenar los datos del usuario en la nube, a modo de copia de seguridad, y poder tener en todo momento un respaldo de los mismos.

Para finalizar, sería interesante la posibilidad de poder implementar tecnologías como OBD para conectar el dispositivo móvil al vehículo mediante bluetooth para actualizar el kilometraje automáticamente cuando estes cerca del vehículo y con ello el usuario no tenga que depender de insertarlo manualmente, cosa que mejoraría bastante la usabilidad de la aplicación.

## 11. Bibliografía/Webgrafía (1-2 páginas)

- [1] - Unión española de entidades aseguradoras y reaseguradoras. (04 de julio 2023). *España supera los 33 millones de vehículos a motor en circulación*. <https://www.unespa.es/notasdeprensa/vehiculos-asegurados-junio-2023/#:~:text=El%20aumento%20interanual%20del%201,el%20tercer%20trimestre%20de%202021.>
- [2] - Google. (2023). *Google Play* (Versión 37.7.22-21). [Aplicación móvil] Google Play. [https://play.google.com/store/games?hl=es\\_419&gl=US](https://play.google.com/store/games?hl=es_419&gl=US)
- [3] - Fiact Seguros. (21 de junio de 2022). *5 Apps para el mantenimiento de coches: ¡descubre las más útiles!* <https://www.fiatc.es/blog/post/app-mantenimiento-coche>
- [4] - Drivvo. (2023). *Drivvo - Gestión de vehículos* (Versión 8.4.8). [Aplicación móvil] Google Play. [https://play.google.com/store/apps/details?id=br.com.ctncardoso.ctncar&hl=es\\_419&gl=US](https://play.google.com/store/apps/details?id=br.com.ctncardoso.ctncar&hl=es_419&gl=US)
- [5] - Mobifolio. (2023) *Simply Auto: Mantenimiento* (Versión 53.2). [Aplicación móvil] Google Play. [https://play.google.com/store/apps/details?id=mrigapps.andriod.fuelcons&hl=es\\_419&gl=US](https://play.google.com/store/apps/details?id=mrigapps.andriod.fuelcons&hl=es_419&gl=US)
- [6] - Google LLC. (2023). *Google Drive* (Versión 2.23.397.0.all.alldpi). [Aplicación móvil] Google Play. [https://play.google.com/store/apps/details?id=com.google.android.apps.docs&hl=es\\_419&gl=US](https://play.google.com/store/apps/details?id=com.google.android.apps.docs&hl=es_419&gl=US)
- [7] - Google LLC. (2023). *Gmail* (Versión 2023.09.24.570825368.Release). [Aplicación móvil] Google Play. [https://play.google.com/store/apps/details?id=com.google.android.gm&hl=es\\_419&gl=US](https://play.google.com/store/apps/details?id=com.google.android.gm&hl=es_419&gl=US)
- [8] - Ian Hawkins. (2018). *Torque Lite (ODB2 & Car)* (Versión 1.2.22). [Aplicación móvil] Google Play. [https://play.google.com/store/apps/details?id=org.prowl.torquefree&hl=es\\_419&gl=US](https://play.google.com/store/apps/details?id=org.prowl.torquefree&hl=es_419&gl=US)
- [9] - Trello, Inc. (2023). *Trello* (Versión 2023.13.1.7275). [Aplicación móvil] Google Play. [https://play.google.com/store/apps/details?id=com.trello&hl=es\\_419&gl=US](https://play.google.com/store/apps/details?id=com.trello&hl=es_419&gl=US)
- [10] - Canva. (2023) *Canva: diseño, foto y video* (Versión 2.238.0). [Aplicación web] Canva. <https://www.canva.com/>

- [11] - Draw.io. (2023). *Diagrams.net* (Versión 22.1.0). [Aplicación web] <https://app.diagrams.net/>
- [12] - Google. (2023). *Android Studio* (Versión 2022.3.1). Google. <https://developer.android.com/studio?hl=es-419>
- [13] - Oracle Corporation. (2022). *Java* (Versión 18). Oracle Corporation. <https://www.oracle.com/es/java/>
- [14] - JetBrains. (2023). *Kotlin* (Versión 1.9.20). JetBrains. <https://kotlinlang.org/>
- [15] - SQLite Consortium. (2023). *SQLite* (Versión 3.44.0). SQLite Consortium. <https://www.sqlite.org/>
- [16] - Google. (2023). *Firebase* (Versión 10.6.0). Google. <https://firebase.google.com/>
- [17] - Software Freedom Conservancy. (2023). *Git* (Versión 2.42.1) Software Freedom Conservancy. <https://git-scm.com/>

## 12. Anexos

### Glosario

BD. Base de Datos

CRUD. Create, Read, Update and Delete (Crear, Leer, Actualizar y Borrar)

DAFO. Debilidades Amenazas Fortalezas y Oportunidades

IDE. Integrated Development Environment (Entorno de desarrollo integrado)

OBD. On Board Diagnostic (Diagnostico a bordo)

PAC. Prova d'avaluació continua (Prueba de evaluación continua)

PK. Primary key (Clave principal)

UML. Unified Modeling Language (Lenguaje unificado de modelado)

## 1.1. Manual de Usuario



# Vehicle Maintainer

Manual de Usuario



## Índice

[Introducción](#)

[Funcionamiento](#)

[Iniciar la aplicación](#)

[Registrar vehículo](#)

[Consultar vehículo](#)

[Registrar mantenimiento realizado](#)

[Consultar mantenimientos realizados](#)

[Actualizar kilometraje](#)

[Programar mantenimiento](#)

[Consultar mantenimientos programados](#)

[Mensajes](#)

[FAQs](#)

## Introducción

Vehicle Maintainer es una aplicación destinada a la gestión y mantenimiento de vehículos.

Vehicle Maintainer es una aplicación móvil desarrollada en Android Studio con la capacidad de almacenar vehículos, almacenar y programar mantenimientos y recibir notificaciones automáticas para informar, cuando llegue el momento, de realizar un mantenimiento.

Este documento recoge el conjunto de funcionalidades disponibles para poder registrar y gestionar los vehículos y los mantenimientos de los mismos de manera efectiva.

Este manual explica detalladamente como realizar todas las funciones que ofrece la aplicación.

## Funcionamiento

### Iniciar la aplicación

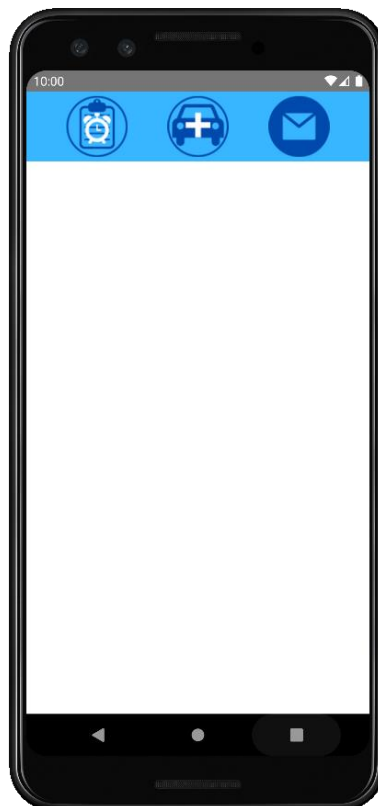
Para iniciar la aplicación Vehicle Maintainer primero se debe instalar.

Para instalar la aplicación se debe incorporar el archivo VehicleMaintainer.apk a tu dispositivo móvil. Una vez incorporado el archivo en el dispositivo móvil, pulsa sobre el archivo y selecciona la opción instalar.

Para iniciar la aplicación, se debe pulsar sobre el icono Vehicle Maintainer.

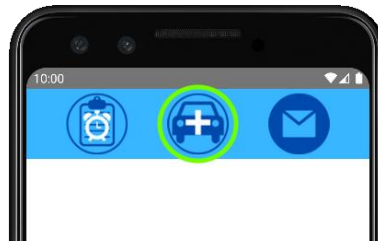


Una vez iniciada la aplicación, se mostrará la pantalla principal de la aplicación.



### Registrar Vehículo

Una vez iniciada la aplicación, para añadir un nuevo vehículo, se debe presionar el botón añadir nuevo vehículo, situado en el medio del menú superior, que muestra la imagen de un vehículo y un símbolo más.



Se abrirá una nueva pantalla donde se deben rellenar todos los campos ofrecidos en referencia al vehículo a registrar.

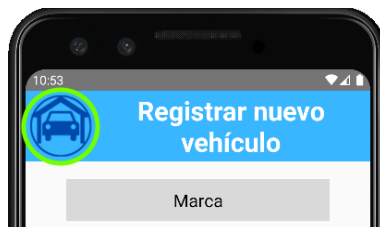
Una vez rellenados todos los campos, se debe pulsar el botón aceptar de color verde.



En el caso de no desear insertar el vehículo, se debe presionar el botón cancelar de color rojo.

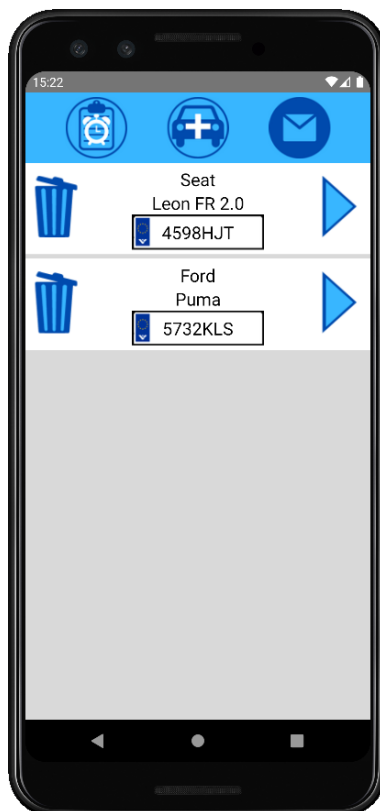


Para volver al menú principal, se debe pulsar en el botón del menú superior con la imagen de un garaje con un vehículo en su interior.

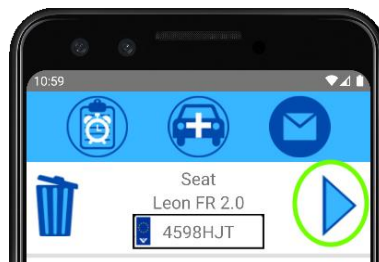


### Consultar vehículo

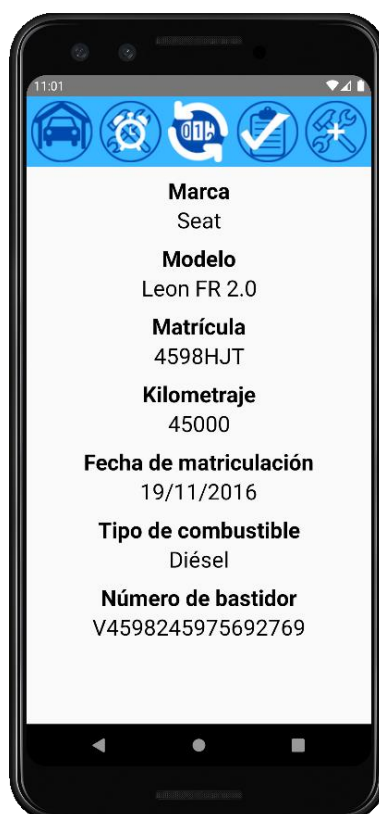
Una vez hecho el registro de un vehículo, este se podrá visualizar en la pantalla principal de la aplicación.



Para consultar al detalle todos los datos en referencia al vehículo se debe presionar el botón con la flecha azul que se muestra a continuación.



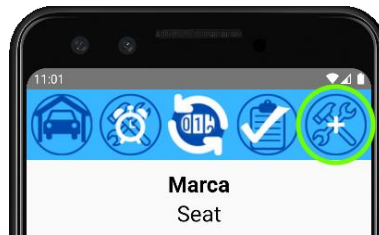
A continuación, se mostrará una pantalla con los datos del vehículo seleccionado en la que se podrá realizar el registro y programación de mantenimientos, así como la actualización del kilometraje.



## Registrar mantenimiento realizado

Para registrar un mantenimiento realizado, se debe seleccionar un vehículo como en el caso de consultar vehículo.

En la pantalla donde se muestran los datos del vehículo se debe presionar el botón situado en el menú superior que muestra la imagen de herramientas y un símbolo más que se muestra a continuación.



A continuación, se mostrará una pantalla donde se deben insertar los datos en referencia al mantenimiento realizado.

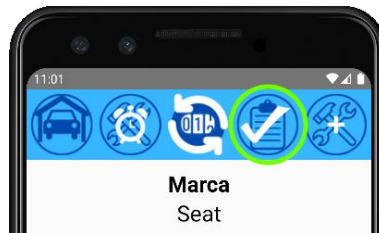


Una vez rellenados todos los campos, se debe pulsar el botón aceptar de color verde.

En el caso de no desear insertar el mantenimiento realizado, se debe presionar el botón cancelar de color rojo.

### **Consultar mantenimientos realizados**

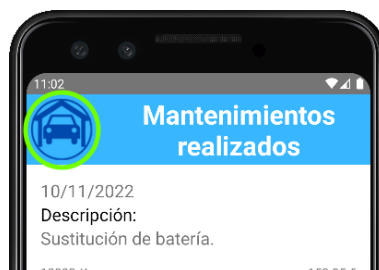
Desde la pantalla donde se muestran los datos del vehículo, se debe presionar el botón del menú superior con la imagen de una lista con un check que se muestra a continuación.



A continuación, se mostrará una pantalla donde se podrán visualizar todos los mantenimientos realizados al vehículo seleccionado.



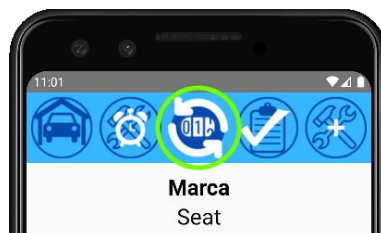
Para volver al menú principal, se debe pulsar en el botón del menú superior que tiene la imagen de un garaje con un vehículo en su interior.



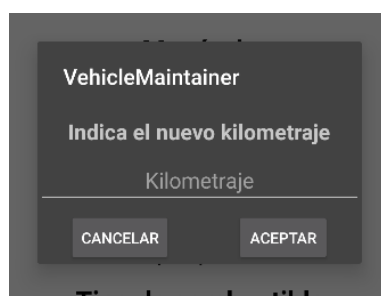
## Actualizar kilometraje



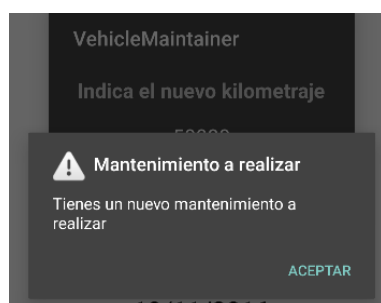
Desde la pantalla donde se muestran los datos del vehículo, se debe presionar el botón central del menú superior que tiene la imagen de un cuentakilómetros con dos flechas que se muestra a continuación.



A continuación, se mostrará una pantalla donde se debe introducir el nuevo kilometraje del vehículo.



Para actualizar el kilometraje se debe introducir un nuevo kilometraje, superior al existente, y presionar el botón aceptar.



Una vez actualizado el nuevo kilometraje podrá verse en la pantalla de los datos del vehículo.

Siempre que se actualice el kilometraje, la aplicación comprobará si existe un mantenimiento programado con el kilometraje indicado o inferior, y si es así, se notificará al usuario mediante una alerta.

Imagen

De lo contrario, al pulsar cancelar, los datos no serán actualizados.

## Programar mantenimiento

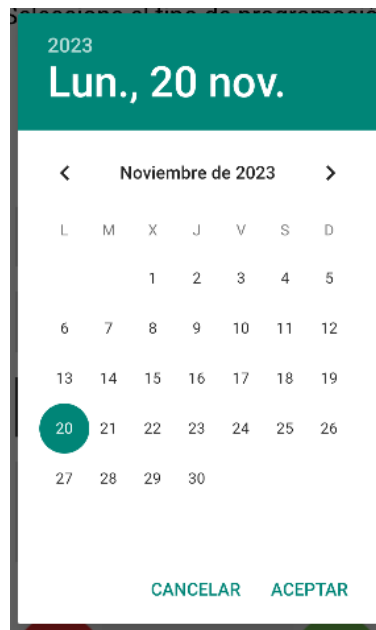
Desde la pantalla donde se muestran los datos del vehículo, se debe presionar el botón del menú superior con la imagen de herramientas con un reloj que se muestra a continuación.

Imagen

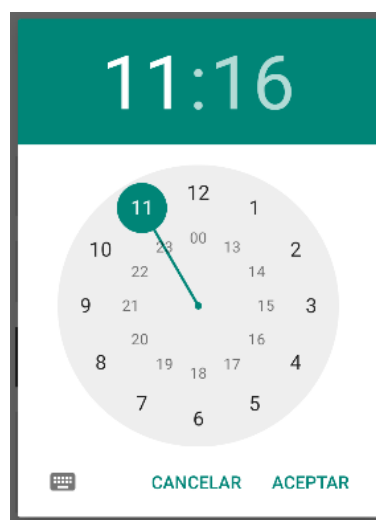
A continuación, se mostrará una pantalla donde se debe rellenar los datos para programar el mantenimiento. Aquí, se debe elegir entre programar por fecha o por kilometraje, seleccionando la opción elegida.

Si se elige la opción de programar el mantenimiento por fecha, el campo de kilometraje será deshabilitado y se deberá elegir fecha y hora.

Al pulsar sobre el campo fecha se mostrará un calendario donde se debe seleccionar la fecha deseada.



Al pulsar sobre el campo hora se mostrará un reloj donde se deberá elegir la hora deseada.



A continuación, se debe indicar una descripción del mantenimiento a realizar.

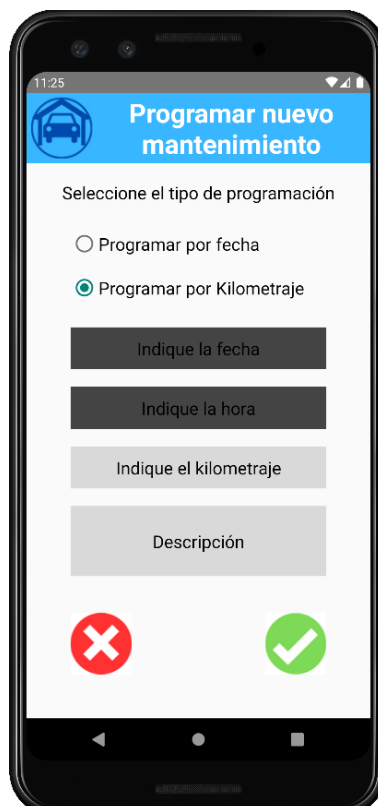
Una vez finalizada la inserción de los datos, se debe pulsar el botón aceptar de color verde.

Tras aceptar la programación de mantenimiento por fecha, la aplicación enviara una notificación en la fecha y hora seleccionada a modo de recordatorio.



De lo contrario, si se pulsa el botón cancelar, la programación será cancelada.

Por otro lado, si se elige programar el mantenimiento por kilometraje, el campo fecha y hora se deshabilitarán y se deberán rellenar el campo kilometraje y la descripción del mantenimiento a realizar.



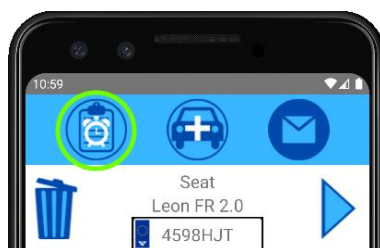
Una vez finalizada la inserción de los datos, se debe pulsar el botón aceptar de color verde y el mantenimiento quedará registrado.

En este caso, como ya se detalló en el apartado actualizar kilometraje, al actualizar el kilometraje del vehículo, si existe un mantenimiento programado con el kilometraje indicado o inferior, se notificará al usuario mediante una alerta.

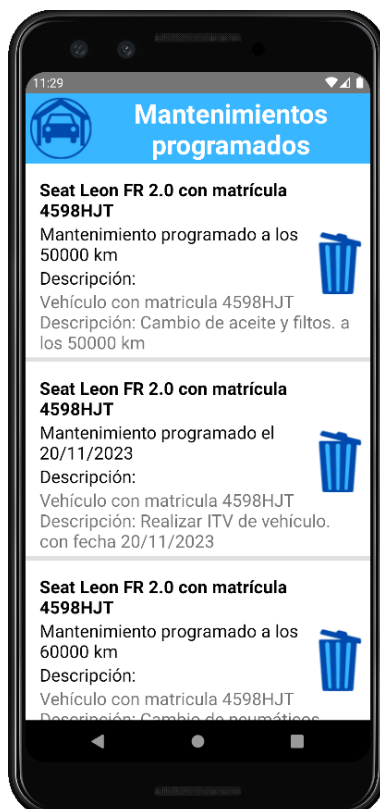
De lo contrario, si se pulsa el botón cancelar, la programación será cancelada.

## Consultar mantenimientos programados

Desde la pantalla principal de la aplicación se debe pulsar en el botón situado a la izquierda del menú superior que tiene la imagen de una lista con un reloj.

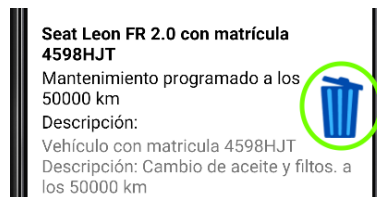


A continuación, se mostrará una pantalla con todos los mantenimientos programados.



Aquí, se podrá consultar y eliminar los mantenimientos programados.

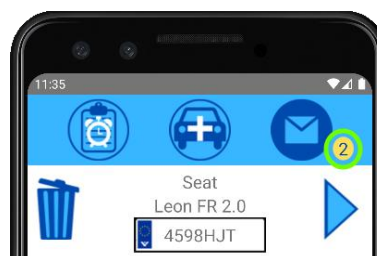
Para eliminar un mantenimiento, se debe pulsar en el botón con la imagen de una papelera.



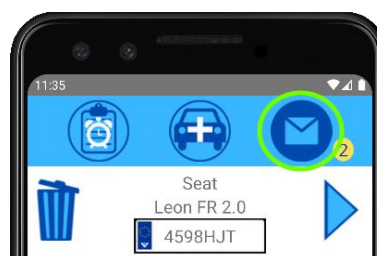
Para volver al menú principal, se debe pulsar en el botón del menú superior.

## Mensajes

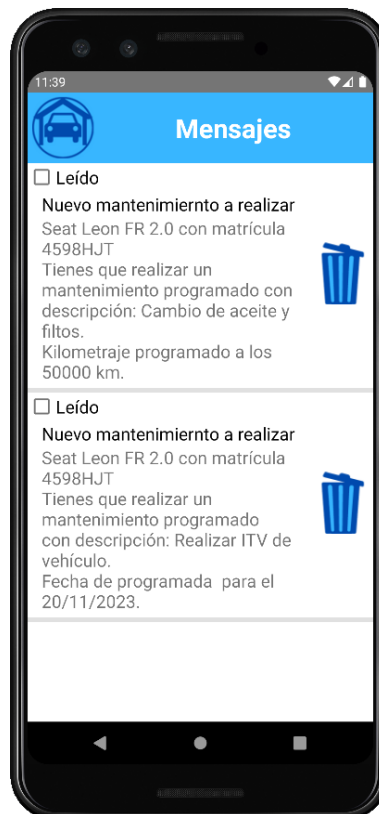
Siempre que existan mensajes nuevos o mensajes no leídos, en la pantalla principal, aparecerá el número de los mismos.



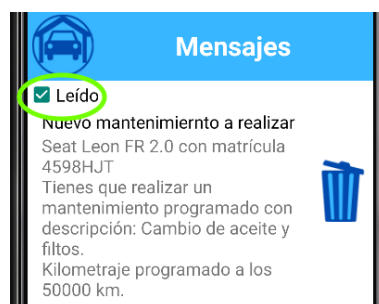
Para acceder a los mensajes se debe pulsar sobre el botón situado a la derecha del menú superior con la imagen de un sobre.



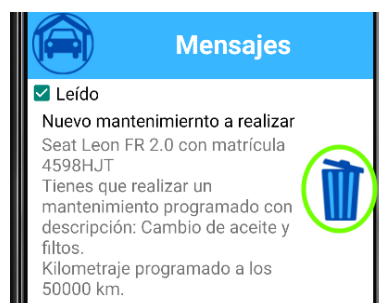
A continuación, se mostrará una pantalla con todos los mensajes existentes.



Los mensajes podrán ser marcados como leídos seleccionando la casilla Leído.



También, se podrán eliminar los mensajes pulsando el botón con la imagen de una papelera.



Para volver al menú principal, se debe pulsar en el botón del menú superior.

## FAQs

### **¿Cómo puedo eliminar un mantenimiento realizado?**

No es posible eliminar un mantenimiento realizado. Estos solo se eliminarán al eliminar el vehículo que los contiene.

### **¿Cómo cancelo la programación de un mantenimiento?**

Para cancelar la programación de un mantenimiento se tendrá que eliminar el mantenimiento programado como se detalla en el apartado de consultar mantenimientos programados.

### **¿Cómo puedo modificar los datos de un vehículo?**

No es posible modificar los datos de un vehículo. Solo es posible actualizar el kilometraje. Si se desea, se puede eliminar el vehículo existente y crear uno nuevo, aunque si se opta por esta opción, todos los datos almacenados en el vehículo eliminado también se eliminarán.

Para poder recibir notificaciones, has de asegurarte de tener activadas las notificaciones en la aplicación. Para ello, accede a Ajustes, Aplicaciones, Gestionar Aplicaciones, Vehicle Maintainer, Notificaciones y activa Mostrar notificaciones.

### **¿Cómo puedo recibir notificaciones?**

Para poder recibir notificaciones, has de asegurarte de tener activadas las notificaciones en la aplicación. Para ello, accede a Ajustes, Aplicaciones, Gestionar Aplicaciones, Vehicle Maintainer, Notificaciones y activa Mostrar notificaciones.