# TRABAJO DE FIN DE GRADO 2º DAM DE ADRIÁN GARCÍA JIMÉNEZ

***MyCarCheck***

8 DE DICIEMBRE DE 2025

IES VLLE INCLÁN
Tutor: Andrés Maroto

Contenido

Introducción: ..... 2

Requisitos Funcionales: ..... 4

Requisitos no Funcionales: ..... 5

Diagrama de clases – Explicación técnica: ..... 6

Modelos de datos: ..... 6

Persistencia: ..... 6

Actividades visuales: ..... 6

Adaptadores: ..... 7

Relaciones clave: ..... 7

Diagrama UML: ..... 8

Diagrama de Gantt: ..... 9

Tabla de presupuesto estimado: ..... 10

**Introducción:**

El presente proyecto, titulado ***MyCarCheck***, surge como solución a una necesidad cotidiana: disponer de una herramienta digital que permita llevar un control organizado y accesible del mantenimiento y repostajes de un vehículo personal. Como usuario habitual de coche, he detectado que muchas personas no registran sus gastos de combustible ni el estado de su vehículo, lo que dificulta la planificación de revisiones, el control del consumo o la detección temprana de anomalías. Esta aplicación permite llevar un registro diario de todos los gastos derivados del uso habitual del vehículo, facilitando una gestión más eficiente y responsable.

En un contexto donde la digitalización de procesos cotidianos es cada vez más común, disponer de una herramienta que centralice el historial de mantenimiento y consumo resulta especialmente útil. Este tipo de soluciones no solo mejoran la organización personal, sino que también contribuyen a una conducción más consciente y sostenible.

Este proyecto se ha desarrollado como parte del módulo de Proyecto Final del Grado Superior en Desarrollo de Aplicaciones Multiplataforma. He elegido esta temática porque me permite aplicar conocimientos clave adquiridos durante el ciclo y otros que he ido aprendiendo según avanzaba la aplicación, adentrándome en nuevos lenguajes de programación como Kotlin, los cuales nunca había utilizado previamente. Migrar todo el código anterior a este lenguaje me ha permitido mejorar la escalabilidad del proyecto y trabajar con una tecnología más versátil y potente orientada al desarrollo móvil.

Además, he profundizado en el uso de Android Studio como entorno principal, entorno que me gustaría que fuese el punto de partida para mi especialización como desarrollador móvil. A lo largo del desarrollo, he aprendido a crear interfaces funcionales, gestionar bases de datos SQLite, implementar control de versiones con Git y GitHub, y documentar técnicamente el software de forma estructurada.

El objetivo principal del proyecto es desarrollar una aplicación funcional, intuitiva y técnicamente sólida que permita al usuario registrar repostajes, gastos y revisiones de su vehículo, ofreciendo además métricas como el consumo medio y el coste por kilómetro. Todo ello con una interfaz clara y una estructura de datos bien definida.

Para llevar a cabo el proyecto, he combinado el uso de herramientas como GitHub para el control de versiones y Android Studio como entorno principal de desarrollo. Esta forma de trabajo me ha permitido detectar errores, documentarlos y corregirlos de forma progresiva.

Este proyecto no solo me ha permitido consolidar conocimientos técnicos, sino también desarrollar una actitud crítica frente a los errores, mejorar mi capacidad de documentación y reforzar mi autonomía como desarrollador. Ha sido una experiencia que me ha motivado a seguir explorando el desarrollo móvil como posible especialización profesional.

**Requisitos Funcionales:**

Los requisitos funcionales definen las acciones que el sistema debe ser capaz de realizar. Para *MyCarCheck*, he identificado los siguientes:

- Registrar nuevos repostajes, incluyendo litros, precio por litro, fecha y kilómetros actuales.
- Calcular automáticamente el precio total del repostaje y el consumo medio.
- Consultar el historial de repostajes realizados.
- Registrar vehículos y asociar cada repostaje a un vehículo concreto.
- Mostrar una lista de repostajes ordenada por fecha.
- Editar o eliminar registros de repostaje.
- Editar o eliminar vehículos
- Almacenar los datos localmente mediante una base de datos SQLite.
- Validar los campos obligatorios antes de guardar un registro.
- Mostrar mensajes de confirmación o error tras cada acción del usuario.
- Permitir la navegación entre pantallas mediante botones o menú.

**Requisitos no Funcionales:**

Los requisitos no funcionales definen las características de calidad del sistema. Para este proyecto se han considerado:

- Interfaz clara, intuitiva y adaptada a dispositivos móviles Android.
- Respuesta rápida del sistema y sin bloqueos perceptibles para el usuario.
- Desarrollada en Kotlin, siguiendo buenas prácticas de programación.
- Control de versiones mediante Git y alojado en un repositorio en GitHub.
- Estructura del proyecto orientada a la escalabilidad y el mantenimiento futuro.
- Documentación técnica actualizada y reflejando la estructura del código y la base de datos.
- Compatibilidad con versiones recientes de Android (API 33 o superior).
- Instalación mediante APK o desde el entorno de desarrollo.
- Preparación para futuras mejoras como sincronización en la nube o exportación de datos.

**Diagrama de clases – Explicación técnica:**

El diagrama de clases representa la estructura lógica del sistema *MyCarCheck*, desarrollado en Android con Kotlin. Se han modelado las entidades principales, las actividades visuales, los adaptadores y la clase de persistencia, reflejando cómo se relacionan los componentes y qué función cumplen dentro del sistema.

***Modelos de datos:***

- **Usuario**: representa al usuario registrado en el sistema. Cada usuario, puede vincular vehículos a un perfil.
- **Vehículo**: contiene atributos como matricula, marca, modelo, tipoCombustible. Cada usuario puede tener varios vehículos.
- **Repostaje**: registra datos como litros, precio, fecha, kilometraje. Cada vehículo puede tener múltiples repostajes asociados.

**Relación**: ***Usuario*** → ***Vehículo*** → ***Repostaje*** (1:N en ambos casos)

***Persistencia:***

- **BaseDeDatos**: clase central que gestiona la persistencia local mediante SQLite. Incluye métodos CRUD para **Usuario**, **Vehículo** y **Repostaje**, así como funciones específicas como `**editarVehiculo**`, `**editarRepostaje**`, `**getVehiculoPorId**` y `**getRepostajePorId**` para facilitar la edición y recuperación individual de registros.
- Todas las actividades acceden a esta clase para guardar o recuperar datos.

***Actividades visuales:***

Cada Activity representa una pantalla funcional del sistema:

- ***MainActivity***: punto de entrada de la app.
- ***LoginActivity*** / ***RegistroActivity***: gestión de acceso básico.

- ***MenuPrincipalActivity***: panel de navegación principal.
- ***AgregarVehiculoActivity***: formulario para registrar vehículos.
- ***ListaVehiculosActivity***: muestra los vehículos registrados.
- ***AgregarRepostajeActivity***: permite registrar un repostaje.
- ***ListaRepostajesActivity***: muestra los repostajes de un vehículo.
- ***EstadisticasActivity***: calcula y muestra estadísticas de consumo.
- ***HistorialRepostajesActivity***: vista alternativa para consultar repostajes.
- ***EditarVehiculoActivity***: permite modificar los datos de un vehículo ya registrado.
- ***EditarRepostajeActivity***: permite modificar los datos de un repostaje existente.

Todas estas actividades interactúan con ***BaseDeDatos*** y con los modelos mediante Intent y validación previa.

***Adaptadores:***

- ***VehiculoAdapter*** y ***RepostajeAdapter***: gestionan la visualización de listas en RecyclerView, conectando los modelos con sus respectivos layouts (***item_vehiculo.xml***, ***item_repostaje.xml***).

***Relaciones clave:***

- Las actividades acceden a ***BaseDeDatos*** para realizar operaciones sobre los modelos.
- Los adaptadores reciben listas de objetos (***Vehículo***, ***Repostaje***) y los renderizan en pantalla.
- Las relaciones entre entidades están claramente definidas para garantizar trazabilidad y validación.

## Diagrama UML:

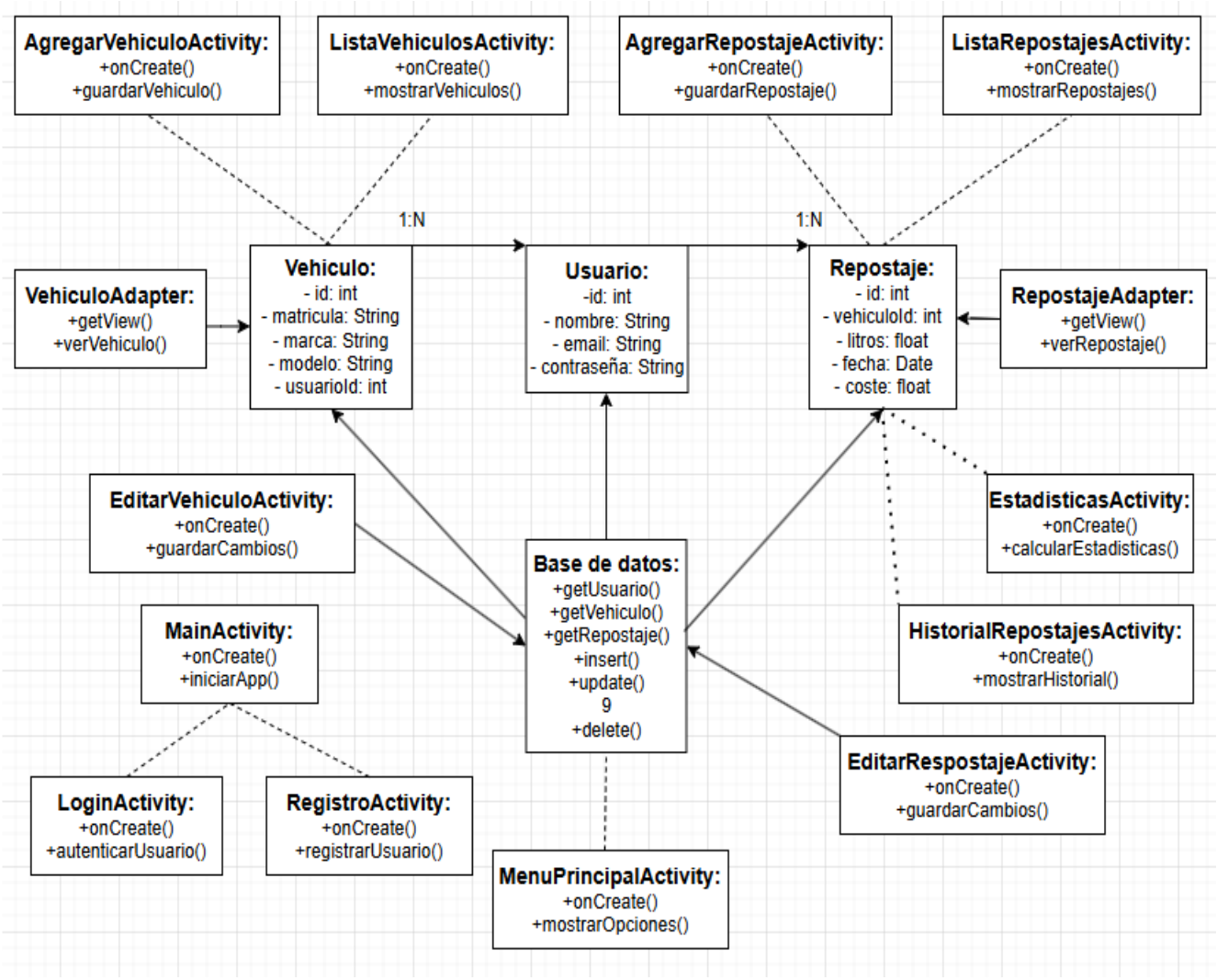

**Diagrama de Gantt:**

El diagrama de Gantt del proyecto ***MyCarCheck*** refleja la planificación temporal del desarrollo, distribuyendo las tareas clave desde el análisis inicial hasta la entrega final. Cada fase se representa con su duración estimada y dependencias, permitiendo visualizar el progreso y controlar los tiempos de implementación, pruebas y documentación. Esta herramienta facilita la organización del trabajo y garantiza una ejecución ordenada del TFG.

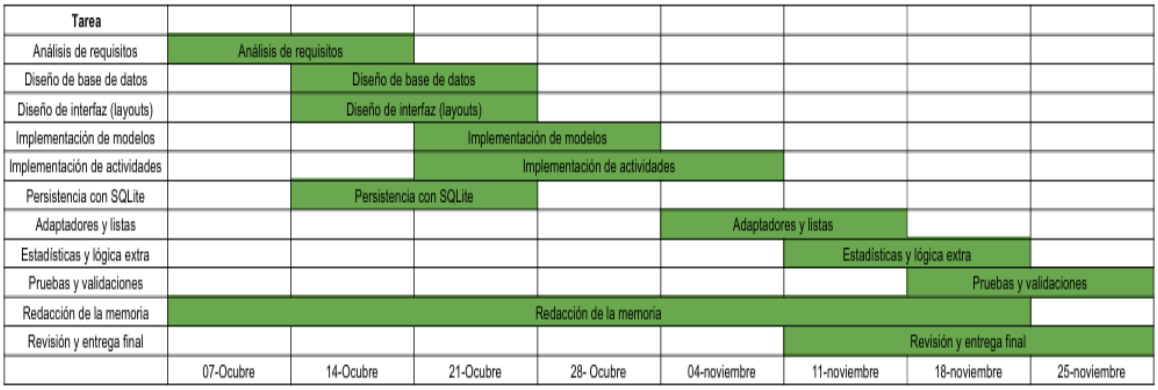

**Tabla de presupuesto estimado:**

| Concepto | Detalle | Coste estimado |
|---|---|---|
| Horas de desarrollo | 150 h × 15 €/h | 2.250 € |
| Redacción de la memoria | 30 h × 12 €/h | 360€ |
| Hardware | Ordenador y móvil durante 3 meses | 150 € |
| Electricidad e internet | Consumo estimado durante el desarrollo | 60€ |
| Licencias de software | Android Studio, SQLite (gratuito) | 0€ |
| Impresión de la memoria | 1 copia física | 20 € |
| Transporte (tutorías) | Desplazamientos puntuales | 30€ |
| **Total estimado** | | **2.870 €** |

El presupuesto del proyecto ***MyCarCheck*** se ha elaborado como una estimación teórica, considerando el coste de desarrollo, documentación, uso de hardware, consumo energético y otros recursos implicados. Se han utilizado herramientas gratuitas, por lo que no se han requerido licencias de pago. El coste total estimado asciende a 2.870 €, reflejando el valor profesional del trabajo realizado.