UNIVERSIDAD DE VALLADOLID

E.U. DE INFORMÁTICA (SEGOVIA)

Grado en Ingeniería Informática de Servicios y Aplicaciones

Android OBD-II

Autor: *Ángel Martín Velázquez*

Tutor: *Ignacio Aparicio Morgado*

ÍNDICE

SECCIÓN I: MEMORIA DEL PROYECTO

SECCIÓN II. MANUAL DE USUARIO

SECCIÓN I

MEMORIA DEL PROYECTO

ESTRUCTURA DE LOS ENTREGABLES

Identificación del TFG

Título: Android OBD-II
Autor: Ángel Martín Velázquez
Tutor: Ignacio Aparicio Morgado

Organización de la Memoria

La memoria de este Trabajo Fin de Grado constará de la siguiente estructura:

- SECCIÓN I: MEMORIA DEL PROYECTO, que contendrá la información de la memoria propiamente dicha y del Manual Técnico. Recoge los siguientes capítulos:
  - CAPITULO 1. La aplicación Android OBD-II
  - CAPÍTULO 2. La planificación del proyecto
  - CAPÍTULO 3. Análisis del sistema
  - CAPÍTULO 4. Diseño del sistema
  - CAPÍTULO 5. Pruebas del sistema
  - CAPÍTULO 6. Conclusiones personales
  - CAPÍTULO 7. Futuras líneas de trabajo
  - CAPÍTULO 8. Glosario
  - CAPÍTULO 9. Bibliografía
  - CAPÍTULO 10. Índice de tablas
  - CAPÍTULO 11. Índice de figuras
- SECCIÓN II. MANUAL DE USUARIO, donde se facilitará la información de ayuda al usuario para la correcta instalación, configuración y uso de la aplicación.
  - CAPÍTULO 1. Requisitos del sistema
  - CAPÍTULO 2. Instalación del sistema
  - CAPÍTULO 3. Configuración
  - CAPÍTULO 4. Manual de uso

Estructura del CD

Dentro de los entregables de este Trabajo Fin de Grado, se incluye un CD con la documentación en formato electrónico, así como los ficheros ejecutables y el código fuente al completo. Para facilitar la navegación por este CD, a continuación se detalla su estructura:

- **CD_TFG_AndroidOBDII**:
  - ***1.-Documentacion***

    En este directorio se incluirá la documentación del proyecto en formato pdf.

  - ***2.-Software***

    En este directorio es donde se encuentra el Software de la aplicación, y se divide en los siguientes subdirectorios:

    - ***2.1-Instalable***

      Aquí se encuentra el archivo de instalación de la aplicación "*AndroidODBII.apk*"

    - ***2.2-Codigo_Fuente***

      Aquí se encuentra el código fuente de la aplicación, incluido dentro de la carpeta "*AndroidODBII*", que contiene todo el proyecto Android.

    - ***2.3-Librerias_Externas***

      En este subdirectorio se han incluido las dos librerías externas que se han añadido al proyecto, por si fueran de utilidad. Estas librerías son:

      "*api-1.3.jar*", librería java que proporciona la lectura de comandos del conector OBDII.

      "*androidplot-core-0.6.0.jar*", librería que facilita la generación de graficas en aplicaciones Android.

CAPÍTULO 1

LA APLICACIÓN Android OBD-II

CAPÍTULO 1
LA APLICACIÓN "Android OBD-II"

1.1 DESCRIPCIÓN GENERAL

Este Trabajo Fin de Grado surge de la curiosidad y del interés por conocer un poco más en profundidad el funcionamiento de la electrónica que actualmente incorporan los vehículos y el increíble hecho de poder disponer de toda esa información en tu propio teléfono móvil. Fusionar estos dos conceptos tan cotidianos y poder disponer y explotar una información, a veces muy valiosa y normalmente poco conocida, que permite al conductor, entre muchas otras cosas, anticiparse a posibles problemas, rectificar sus conductas a la hora de conducir y permitir obtener una mayor eficiencia en sus desplazamientos.

El nexo de unión, para poder vincular estas cosas tan cotidianas y aparentemente tan poco relacionadas, es el puerto de diagnosis del vehículo, que se apoya en protocolo OBD-II (On-Board Diagnosis). Este puerto es el utilizado por los profesionales en los talleres y hasta hace relativamente poco requería de equipos muy sofisticados y caros para poder visualizar la información que ofrece.

El término OBD se refiere a la capacidad de auto diagnóstico y monitorización de la que disponen los coches para mantener bajo control todos los parámetros que pueden afectar a las emisiones, rendimiento y salud del automóvil. Éste sistema recoge la información generada por los distintos sensores para mantener bajo control y disparar los distintos sistemas de los que el vehículo dispone, tanto de seguridad como para regular el funcionamiento del motor, etc. Como por ejemplo, puede realizar funciones tan dispares como poner en funcionamiento el ABS o ESP, controlar la velocidad del ventilador o regular la apertura de la mariposa de admisión.

Estos sistemas electrónicos surgieron de la necesidad de controlar y auto-regular las emisiones, en los sistemas de inyección electrónica y el primer fabricante en introducir un sistema de este tipo, con capacidad para monitorizarlo fue Volkswagen en el año 1969. A partir de ese momento, y con la creciente expansión de la electrónica en los coches cada fabricante fue desarrollando su propio sistema de diagnosis, hasta que en 1988 la "*Society of AutomotiveEngineers*" (SAE) recomendó estandarizar el conector de diagnosis y un conjunto de señales. A partir de ese momento y gracias a ciertas restricciones legislativas se fue imponiendo un sistema estandarizado, que comenzó con el OBD-I, y evolucionó hasta el actual OBD-II. Este protocolo, realmente no es un estándar completo y cerrado, sino que establece una serie de parámetros comunes, y después cada fabricante puede extender éstos e incluir parámetros propios. De hecho, hoy en día conviven varios estándares equivalentes a este protocolo, que se han ido implementando por restricciones legales de los distintos países, como puede ser el *EOBD (EuropeanOnBoardDiagnostics)* en Europa o el *ISO 15765-4* en Estados Unidos.

A continuación mostramos un diagrama de un puerto OBD II, que se encuentra disponible en el interior del vehículo, que es un conector de 16 pines:

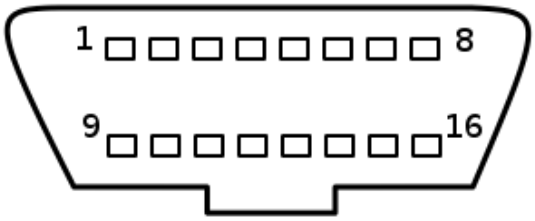


Figura 1 - Puerto OBD-II

Ejemplo de localización del puerto OBD2 (coche propio, modelo Opel Astra):

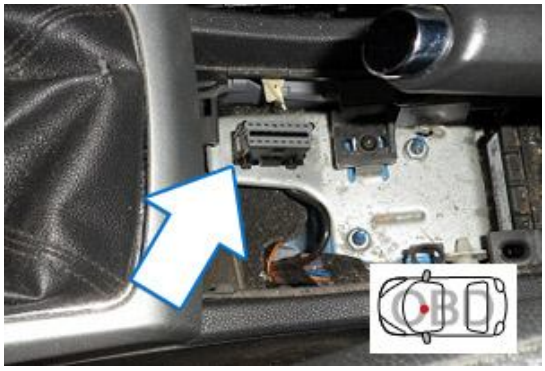


Figura 2 - Puerto USB en Opel Astra

Los distintos tipos de dispositivos que se pueden conectar a este puerto son:

- Máquinas de diagnosis. Son los dispositivos utilizados por los profesionales y talleres, son dispositivos de diagnosis especializados (incluso específicos de cada fabricante), y habitualmente muy caros:

Figura 3 - Dispositivo especializado

- Desde hace pocos años, han surgido conectores que permiten la lectura que ofrece el protocolo OBD-II y que son capaces de transmitir la información del vehículo para ser procesada por ordenadores a través de USB o puerto serie, pero requieren de programas especializados, normalmente también bastante caros, pero no tanto como las máquinas comentadas en el punto anterior:

Figura 4 - Conector USB y puerto serie

- Los últimos en aparecer han sido conectores OBD capaces de obtener la misma información y transmitirla por Bluetooth. Estos conectores han sido una verdadera revolución, pues por muy poco dinero (disponibles desde unos 20 euros), se puede visualizar dicha información en multitud de programas de ordenador, pero además ha abierto la puerta a la utilización de smartphones y tablets (Android e iOS), con lo que el precio se reduce drásticamente al no tener que disponer de un dispositivo especializado para solo este cometido. Además los programas disponibles para estos dispositivos, si bien tienen sus limitaciones en cuanto a funcionalidad, son muy baratos o incluso gratuitos.

Figura 5 - Conector OD-II Bluetooth

En concreto, este último conector, el ELM 327, ha sido el conector utilizado para el desarrollo y pruebas del presente Trabajo Fin de Grado.

Una vez determinado el conector que va a proporcionar la información, ésta debe ser procesada y mostrada, y el dispositivo elegido para realizar esta funcionalidad, ha sido un terminal móvil con sistema operativo Android con conexión Bluetooth, debido a su gran expansión, y las facilidades que ofrece sus sistema operativo para su programación.

El objetivo del Trabajo Fin de Grado es el desarrollo de un programa en Android, que permita obtener la información extraída del puerto OBD-II para su visualización "en vivo" durante la conducción, de la información más relevante, así como el enriquecimiento de esta información a partir de los sensores del teléfono Android.

Las partes en las que se estructura el proyecto son:

- Conexión con OBD2.
- Visualización de Parámetros.
- Información adicional.
- Almacenamiento en Base de datos.
- Datos globales y Estadísticas.

Y ahora pasamos a explicar brevemente cada una de ellas:

1.2 CONEXIÓN CON OBD-II

Es el objetivo principal del proyecto, tiene una complejidad doble, pues por un lado, tiene que proporcionar el establecimiento y gestión de la conexión bluetooth con el conector OBD-II, y por otro también tiene que cubrir la funcionalidad necesaria para entenderse con el conector y poder consultar a éste de forma periódica para obtener los parámetros del sistema (OBD-II PIDs, o "*On-boarddiagnosticsParameterIDs*"), tales como la velocidad, las revoluciones por minuto del motor o la temperatura del refrigerante.

La solución de esta conexión viene desarrollada en el software base del que se ha partido, tan solo ha sido necesario realizar pequeños cambios para hacer el software un poco más robusto y más eficiente.

1.3 VISUALIZACIÓN DE PARÁMETROS

Una vez fijada la periodicidad de las llamadas, cada vez que se obtiene el valor de un parámetro, veremos la representación de este valor por pantalla, para que el conductor pueda observar en tiempo real el estado y condiciones del vehículo y representar esta información de forma clara, concisa y rápida.

1.4 INFORMACIÓN ADICIONAL

Además de la información obtenida por el OBD-II, el propio terminal móvil permite calcular información importante como la velocidad media, también permite generar

información con sus propios sensores y funciones, como calcular la distancia usando el posicionamiento GPS, o mostrar una brújula, utilizando los sensores internos de los que se dispone.

1.5 ALMACENAMIENTO EN BASE DE DATOS

Para un mayor aprovechamiento de la información, se ha dotado a la aplicación de una base de datos para poder recuperar la información en cualquier momento. Además, se ha desarrollado la funcionalidad de exportación, que genera la información almacenada en ficheros "csv" para su posible tratamiento con otras herramientas. Por ejemplo, se podría pasar a un ordenador y abrir con Excel para una visualización rápida y generación de gráficas básicas, hasta cargar esta información en herramientas matemáticas, como ***MATLAB***..., para análisis más complejos.

1.6 DATOS GLOBALES Y ESTADÍSTICAS

Por último, para poder aprovechar toda la información almacenada desde el propio terminal móvil y obtener la información más representativa, se ha desarrollado una interfaz para mostrar los datos globales del último viaje, así como unas gráficas estadísticas para ver la evolución a lo largo de los últimos viajes. Haciendo uso de estas estadísticas, el conductor podrá conocer de forma empírica cual ha sido su comportamiento al volante y poder mejorar la eficiencia de conducción.

CAPÍTULO 2

LA PLANIFICACIÓN DEL PROYECTO

CAPÍTULO 2
LA PLANIFICACIÓN DEL PROYECTO

2.1. INTRODUCCIÓN

Los principales motivos por los que decidí llevar a cabo este proyecto es que se trata de un ámbito totalmente distinto al tipo de proyectos en los que he participado durante mi carrea profesional y a los que estoy habituado (grandes proyectos para grandes empresas de Telecomunicaciones y Banca); por el interés que despierta en mí la tecnología aplicada al mundo del automóvil; y por el reto que suponía trabajar en una tecnología muy poco conocida y en un ámbito funcional totalmente desconocido. Es muy posible que toda esta complejidad en otro momento me hubiera echado para atrás, pero me atrajo mucho aprovechar esta oportunidad para hacer frente a la dificultad, adquirir más conocimientos y distanciarme de mi zona de "confort".

Realmente esto me ha planteado muchas dificultades que he tenido que subsanar, dedicándole más tiempo y esfuerzo del previsto inicialmente.

2.2. ÁMBITO DE LA APLICACIÓN

A priori, desarrollar una aplicación desde cero con toda esta funcionalidad, se escapaba en tiempo del plazo y coste en horas/persona que establece el Trabajo Fin de Grado, así que después de barajar varias posibilidades y estudiar las posibles soluciones, se optó por partir de un proyecto de software libre con licencia Apache 2, que me ha proporcionado la base sobre la cual trabajar. Este proyecto es el "***OBD-II reader for Android devices***" que se encuentra en la siguiente URL de Google Code: http://code.google.com/p/android-obd-reader/.

Para la programación se ha utilizado la plataforma de desarrollo de Android, que se basa en el lenguaje de programación Java, enriquecido con características propias. Además, se ha hecho uso de la base de datos integrada en este SDK (Software Development Kit) que es SQLite3 que, aunque limitada, es muy ligera y se adapta perfectamente a este tipo de aplicaciones en dispositivos móviles, con una capacidad de procesamiento y espacio un tanto restringido.

2.3. OBJETIVOS DEL PROYECTO

El objetivo principal del proyecto ha sido el poder ofrecer la información interna del automóvil, durante la conducción, a cualquier persona que disponga de un teléfono o tablet con sistema operativo Android. Por ello, para llegar a un mayor número de

personas, se ha desarrollado sobre la versión 2.3.3 de éste sistema operativo, que es compatible con las versiones posteriores. La mayoría de los dispositivos actuales poseen esta versión o una superior, con lo que la disponibilidad está garantizada.

Pero no solo ha sido este el objetivo perseguido, sino también, diferenciarse de las aplicaciones similares que ya están disponibles en el mercado y poder ofrecer información calculada, disponibilidad de la información en cualquier momento y generación de gráficas estadísticas, que permitan al usuario poder aprovechar toda la información obtenida con antelación, pues la mayoría de las aplicaciones similares, una vez que sales del coche, no permiten realizar acción alguna y mucho menos disponer de toda esta información.

2.4. ESTIMACIÓN DE TIEMPOS

En este punto veremos la distribución de los tiempos estimados en las tareas que se han llevado a cabo para la realización del trabajo.

Como podremos ver, una parte muy importante era el estudio de la aplicación base, tarea fundamental para construir el proyecto completo.

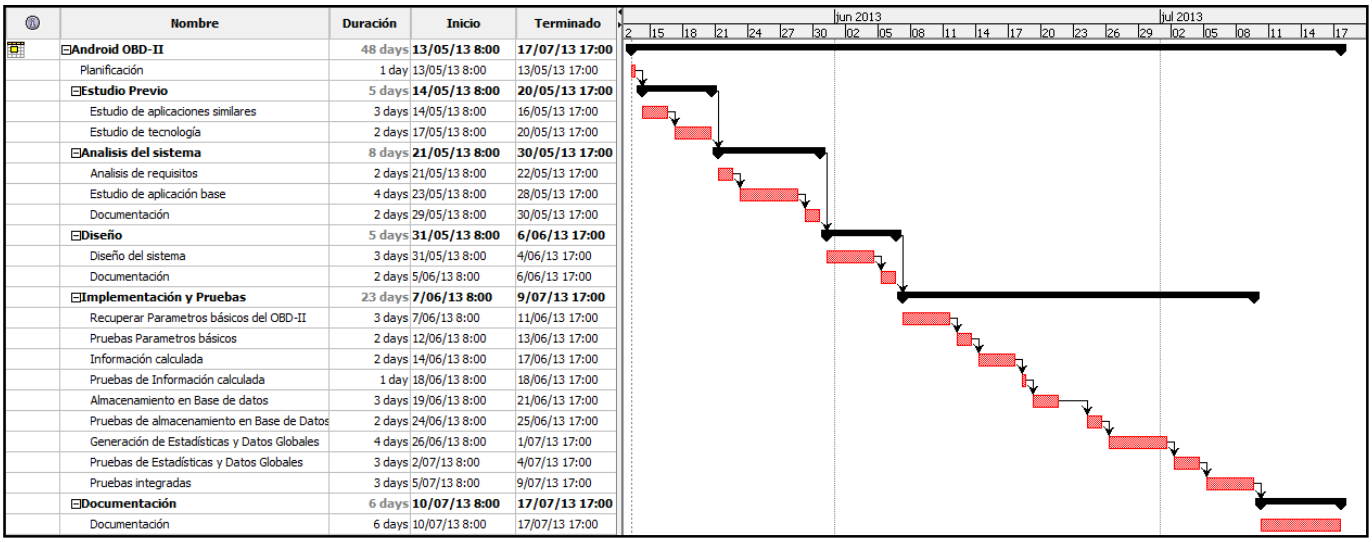
Por tanto, la estimación de tiempos resulta de la siguiente manera:

- **8 horas**: Planificación de las etapas del trabajo.
- **40 horas**: Estudio de la aplicación base.
- **64 horas**: Análisis del sistema
- **40 horas**: Diseño del sistema.
- **184 horas**: Implementación de la aplicación:
    - **96 horas**: Implementación.
    - **88 horas**: Pruebas
- **80 horas**: Documentación y Memoria del Trabajo.

La suma de todas estas tareas hacen un total de 384 horas. Suponiendo que el proyecto lo acometa un Analista-Programador con una jornada laboral de 8 horas diarias, de lunes a viernes, el desarrollo del proyecto tendría una duración de 48 días laborables, que correspondería a unos 58 días naturales, más festivos.

Tras analizar todos estos datos, se estima que la duración total del proyecto será de 65 días naturales, es decir, aproximadamente dos meses y una semana.

A continuación, se muestra un diagrama de Gantt, que permite ver, de una forma gráfica y clara, las tareas a realizar a lo largo del tiempo en el periodo de duración del proyecto. Se puede observar las fechas a cumplir en cada una de las tareas y subtareas.



	Nombre	Duración	Inicio	Terminado
	Android OBD-II	48 days	**13/05/13 8:00**	**17/07/13 17:00**
	Planificación	1 day	13/05/13 8:00	13/05/13 17:00
	Estudio Previo	5 days	**14/05/13 8:00**	**20/05/13 17:00**
	Estudio de aplicaciones similares	3 days	14/05/13 8:00	16/05/13 17:00
	Estudio de tecnología	2 days	17/05/13 8:00	20/05/13 17:00
	Analisis del sistema	8 days	**21/05/13 8:00**	**30/05/13 17:00**
	Analisis de requisitos	2 days	21/05/13 8:00	22/05/13 17:00
	Estudio de aplicación base	4 days	23/05/13 8:00	28/05/13 17:00
	Documentación	2 days	29/05/13 8:00	30/05/13 17:00
	Diseño	5 days	**31/05/13 8:00**	**6/06/13 17:00**
	Diseño del sistema	3 days	31/05/13 8:00	4/06/13 17:00
	Documentación	2 days	5/06/13 8:00	6/06/13 17:00
	Implementación y Pruebas	23 days	**7/06/13 8:00**	**9/07/13 17:00**
	Recuperar Parametros básicos del OBD-II	3 days	7/06/13 8:00	11/06/13 17:00
	Pruebas Parametros básicos	2 days	12/06/13 8:00	13/06/13 17:00
	Información calculada	2 days	14/06/13 8:00	17/06/13 17:00
	Pruebas de Información calculada	1 day	18/06/13 8:00	18/06/13 17:00
	Almacenamiento en Base de datos	3 days	19/06/13 8:00	21/06/13 17:00
	Pruebas de almacenamiento en Base de Datos	2 days	24/06/13 8:00	25/06/13 17:00
	Generación de Estadísticas y Datos Globales	4 days	26/06/13 8:00	1/07/13 17:00
	Pruebas de Estadísticas y Datos Globales	3 days	2/07/13 8:00	4/07/13 17:00
	Pruebas integradas	3 days	5/07/13 8:00	9/07/13 17:00
	Documentación	6 days	**10/07/13 8:00**	**17/07/13 17:00**
	Documentación	6 days	10/07/13 8:00	17/07/13 17:00

Figura 6 - Planificación inicial

2.5. ESTIMACIÓN DE COSTES

Como ya hemos visto en el punto anterior, la duración de este proyecto ha sido estimada en 384 horas, con 8 horas diarias de trabajo. Para simplificar el cálculo del coste y debido a la tipología del proyecto, considero que lo puede llevar a cabo perfectamente un Analista-Programador con experiencia suficiente para realizar las labores de análisis y planificación. Por tanto, teniendo en cuenta estas suposiciones y que el coste por hora de este Analista-Programador es de 30 €, nos sale un total de costes de Personal de 11520 €.

Respecto al material y software empleado, se ha hecho un gran esfuerzo en reducir los costes, recurriendo a la utilización de software libre y aplicaciones gratuitas siempre que ha sido posible (a excepción de la utilización de Microsoft Windows y Office, que por estar acostumbrado a su uso, redunda en un menor tiempo para la realización de las tareas en las cuales se requiere su utilización).

A continuación, detallamos en la siguiente tabla el desglose de los costes del proyecto:

TIPO DE RECURSO	RECURSO	COSTE DE ADQUISICION	% COSTE APLICADO	COSTE DEL PROYECTO
HUMANO	Salario	11.520 €	100%	11.520 €
MATERIAL	Ordenador portátil - SONY VAIO	750 €	20%	150 €
	Teléfono móvil - SAMSUNG GALAXY NEXUS	400 €	20%	80 €
	CONECTOR OBD-II - ELM 327	35 €	100%	35 €
	Aplicación Base - "***android-obd-reader***" (Software Libre, licencia Apache 2)	Gratuito	100%	0 €

	Librería de generación de gráficas - "***Androidplot***" (Software Libre, licencia Apache 2)	Gratuito	100%	0 €
	Microsoft Windows 8	Incluido en precio de ordenador	100%	0 €
	Eclipse	Gratuito	100%	0 €
	Android SDK	Gratuito	100%	0 €
	SQLiteAdmin	Gratuito	100%	0 €
	StarUML	Gratuito	100%	0 €
	OpenProj	Gratuito	100%	0 €
	Microsoft Office 2007	130 €	20%	26,00 €
	Material de ofimática, soportes y papelería	200 €	100%	200 €
TOTAL:				**12.011 €**

Tabla 1 - Desglose de los costes del proyecto

2.6. METODOLOGÍA

Toda metodología no es más que un conjunto de métodos, técnicas, herramientas y soporte documental que ayudan a cubrir más de una etapa del ciclo de vida. Ese conjunto de métodos, técnicas y herramientas deben especificar cómo se tiene que dividir el proyecto en las fases, etapas y actividades correspondientes; así como dar una explicación clara y práctica de cómo llevar a cabo cada tarea en concreto.

La metodología representa el camino para desarrollar el software de una forma sistemática. Esto se logra gracias al control y seguimiento del trabajo marcado por las salidas que se producen y por las restricciones aplicadas al comienzo del mismo. Para el desarrollo del Software, también se deberá de tener en cuenta la documentación que se genere.

Son varias las razones de peso para optar por la Metodología Orientada a Objetos, para la realización de este Proyecto Software. En primer lugar, es la metodología que proporciona el Android SDK para la programación en Android. Además, al ser la ampliación y mejora de un proyecto ya existente, se ha optado por continuar usando la metodología del proyecto original, que también corresponde a una metodología Orientada a Objetos. Gracias a ella se pueden manejar sistemas con una alta complejidad y permite estructurar la arquitectura software en niveles y capas de tal manera que se permitiese un acceso flexible, versátil e independiente en cada una de ellas; así como un mejor mantenimiento del programa.

Básicamente, esta metodología permitió escribir el software de tal forma, que el código quedase organizado de la misma manera que la aplicación original. De esta manera se permite aprovechar sus ventajas: la herencia, polimorfismo y sobre-escritura de métodos para conseguir una mayor funcionalidad, manteniendo el código original,

para cumplir con las restricciones de la licencia Apache 2. Desde un principio se desestimaron otros lenguajes de programación y metodologías, por no cubrir las necesidades del proyecto.

Ventajas de la elección de esta metodología:

- **Compresión:** Tanto los datos que componen los objetos, como los procedimientos que los manipulan, están agrupados en clases. Estas clases se corresponden con las estructuras de información.

- **Uniformidad:** La representación de los objetos implica el análisis, el diseño y la codificación de los mismos.

- **Mantenimiento:** Permite realizar un tratamiento diferente entre aquellos objetos que permanecen constantes en el tiempo y los que cambian con frecuencia, se pueden aislar las partes del programa que permanecen inalterables en el tiempo.

- **Reusabilidad:** La noción de *clase* permite la reutilización de las definiciones de clases empleadas para otros programas e incluso de los procedimientos que las manipulan. De esta forma, el desarrollo de un programa nuevo puede llegar a ser una simple combinación de clases ya definidas en diferentes desarrollos, evitando así el famoso dicho de "reinventar la rueda".

La metodología se ha de aplicar dentro de un entorno. Para ello, se necesita conocer, además del entorno mismo sobre el que se aplica la tecnología, los factores que influyen en la elección de la misma.

Estos factores son los siguientes:

- La **Organización**: Comprobar si ésta es innovadora o conservadora y analizar si sus recursos están orientados a los Sistemas de Información.
- **Profesionales**: Evaluar la formación, experiencia y nivel de los profesionales que van a desarrollar el Software.
- **Técnicos**: Analizar las preferencias técnicas del personal con sus correspondientes imposiciones técnicas.
- De **finalidad**: Este factor se hará en función del tipo de Sistema de Información a desarrollar y según la ambición que se quiere tomar con el desarrollo de dicho Sistema.

Las características que debe tener una buena Metodología vienen marcadas por los siguientes puntos:

1. Debe cubrir el mayor número de etapas de desarrollo de un sistema de información.
2. Mostrar una cobertura total del ciclo de vida del Software.
3. Facilitar la comunicación entre aquellas personas que están involucradas en su desarrollo.

4. Tiene que hacer visible y controlable el avance del sistema que se desarrolla.
5. Facilitar la gestión y el seguimiento del proyecto.
6. La metodología debe contener actividades que mejoren el proceso de desarrollo.
7. Debe incluir la definición de las restricciones del sistema.
8. Por último, tiene que soportar la validación y verificación de toda la documentación generada.

Con todo esto, se puede decir, que si el ciclo de vida nos sirve para indicarnos que es lo que hay que obtener a lo largo del proyecto y los procesos que deben de estar presentes para que esto se pueda conseguir, la metodología será la encargada de indicar cómo se tiene que proceder con cada una de las actividades que forman el proceso.

2.7. HERRAMIENTAS EMPLEADAS

Durante el tiempo de desarrollo del Proyecto se ha tenido que recurrir al uso de numerosas herramientas con las que poder materializar la idea planteada.

Para estructurar este apartado, se ha optado por agruparlas en función de la etapa en la que haya sido utilizada. A continuación, se detalla una breve descripción de cada una de ellas:

2.7.1. Hardware

1. Ordenador portátil: SONY® VAIO.

    - Intel® Core™ i7-3632QM a 2’2 GHz
    - 8 GB RAM
    - 750 GB HDD

2. Smartphone: Samsung Galaxy Nexus.

    - Procesador TI OMAP 4460 dual-core 1.2 GHz, GPU PowerVR SGX540
    - 1 GB RAM
    - 16GB memoria interna

3. Conector OBD-II: ELM 327 Bluetooth.

2.7.2. Software

2.7.2.1. Común a todas las etapas

- *Microsoft® Windows 8*: Sistema operativo del ordenador utilizado para el desarrollo del proyecto.
- *Microsoft® Office 2007*: Paquete de aplicaciones de ofimática. Las principales herramientas utilizadas de este paquete han sido:

  - *Microsoft® Word*: Procesador de textos, para la realización de la memoria y distintos documentos auxiliares.
  - *Microsoft® Excel*: Hoja de cálculo, para cálculos auxiliares, visualización de datos exportados, y análisis de los mismos.
  - *Microsoft® Power Point*: Potente herramienta para realizar presentaciones.
- *Microsoft® Paint*: Con esta herramienta se ha realizado el tratamiento de las imágenes utilizadas a lo largo del programa y su memoria. La principal característica que motivó su elección fue la sencillez que presenta ante el usuario y que se trata de una herramienta incluida por defecto en el sistema operativo.
- *OpenProj*: Programa gratuito que permite la realización de la planificación y seguimiento del proyecto. Tras la realización de la planificación, y generación del diagrama Gantt, esta herramienta permite registrar el avance del proyecto y vigilar sus desviaciones, por tanto afectaría tanto a la fase de planificación, en un primer momento como al resto de etapas del proyecto para vigilar su avance.

2.7.2.2. Etapa de análisis y diseño

- *Star UML 5.0*: Programa gratuito que permite diseñar todo tipo de diagramas UML (secuencia, colaboración, clases). Además permite exportar éstos a formato .jpg para su visualización en procesador de textos.

2.7.2.3. Etapa de Implementación

- *Eclipse Kepler*: Ha sido la herramienta empleada en la etapa de implementación. Además para facilitar la programación de la aplicación del Smartphone, ha sido necesario incorporara el Android SDK (Software Development Kit) al Eclipse. Por tanto el lenguaje de desarrollo que se ha utilizado en este entorno es el lenguaje orientado a objetos Java, con las propiedades añadidas de Android.
- *SQLiteAdmin*: Esta herramienta, ha permitido explotar, comprobar y consultar la base de datos utilizada en el proyecto (SQLite 3).

2.7.2.4. Etapa de Documentación

- *ADOBE® Acrobat Reader 9.1*: Software gratuito que le permite abrir, visualizar y buscar archivos .pdf.
- *doPDF*: Impresora virtual, que permite la creación de documentos PDF, a partir de documentos en otros formatos.

CAPÍTULO 3

ANÁLISIS DEL SISTEMA

CAPÍTULO 3
ANÁLISIS DEL SISTEMA

3.1 INTRODUCCIÓN

El objetivo de este capítulo es analizar los requisitos que la aplicación debe cumplir. Se determina de esta manera lo que tiene que realizar el Sistema, definiendo las posibles restricciones sobre su operación y su implementación.

El análisis de requisitos será, por tanto, el proceso de estudio de todas las necesidades de los usuarios que interactúan con el sistema, definiéndose así los requisitos de hardware y de software del sistema.

Hay que señalar que un requisito no es más que una condición que necesita el usuario para poder resolver un problema y conseguir después un objetivo determinado. Puesto que son la base de cualquier anotación o tarea a realizar a posteriori, deben de estar abiertos a una futura interpretación. Para ello, la mejor forma de especificar los requisitos será la de definirlos de una forma clara, concisa y detallada.

Antes de pasar a describir los requerimientos del sistema, conviene recordar al lector el alcance y las características generales del proyecto. Para ello, resumiendo los capítulos anteriores del presente documento, cabe destacar que se trata de una aplicación a ejecutar en un dispositivo móvil (smartphone o tablet), con sistema operativo Android, que aportará al conductor de un vehículo información importante sobre la conducción, es decir, cumple las funciones de un ordenador de a bordo, visible desde el dispositivo móvil comentado anteriormente.

3.2 DESCRIPCIÓN DEL SISTEMA ACTUAL

Como base del proyecto, se ha partido de la aplicación de software libre "*android-obd-reader*", con licencia Apache 2, disponible en:

http://code.google.com/p/android-obd-reader/

Actualmente, esta aplicación provee de una serie de características esenciales para el proyecto, como son una librería para poder interpretar los comandos OBD-II extraídos del vehículo y el establecimiento de la conexión Bluetooth con el conector OBD.

3.3 OBJETIVOS DEL SISTEMA

Lo primero a detallar en este análisis, serán los objetivos que se pretenden alcanzar cuando el sistema software a desarrollar esté finalizado. Éstos objetivos son los siguientes:

OBJ-01	***Visualización de información del vehículo***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Descripción	El sistema deberá ser capaz de obtener la información en tiempo real del vehículo.
Subobjetivos	N/A
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Baja
Comentarios	N/A

Tabla 2 - Objetivo 01

OBJ-02	***Obtención de información de los sensores del terminal***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Descripción	El sistema deberá ser capaz de obtener información en tiempo real de los sensores del dispositivo móvil, para completar la información a visualizar.
Subobjetivos	N/A
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Baja
Comentarios	N/A

Tabla 3 - Objetivo 02

OBJ-03	***Generación de información calculada***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Descripción	El sistema deberá ser capaz de generar información calculada a partir de la información obtenida en los objetivos **OBJ-01** y **OBJ-02**.
Subobjetivos	N/A
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Baja
Comentarios	N/A

Tabla 4 - Objetivo 03

OBJ-04	***Exportación de la información disponible***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Descripción	El sistema deberá ser capaz de exportar la información que se ha

	ido obteniendo y que se ha deseado almacenar, para su posible explotación con herramientas externas.
Subobjetivos	N/A
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Baja
Comentarios	N/A

Tabla 5 - Objetivo 04

OBJ-05	***Generación de estadísticas***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Descripción	El sistema deberá ser capaz de generar gráficas estadísticas, a partir de la información obtenida, de modo que se pueda ver analizar de modo visual esta información.
Subobjetivos	N/A
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Baja
Comentarios	N/A

Tabla 6 - Objetivo 05

OBJ-06	***Configuración del sistema***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Descripción	El sistema deberá proveer de herramientas de configuración para adaptar la visualización y obtención de información, a las necesidades del usuario.
Subobjetivos	N/A
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Baja
Comentarios	N/A

Tabla 7 - Objetivo 06

OBJ-07	***Ayuda***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Descripción	El sistema deberá proveer de herramientas de consulta, que permitan al usuario conocer el funcionamiento y manejo de la aplicación.
Subobjetivos	N/A
Importancia	Alta

Urgencia	Alta
Estado	Validado
Estabilidad	Baja
Comentarios	N/A

Tabla 8 - Objetivo 07

3.4 CATÁLOGOS DE REQUISITOS DEL SISTEMA

En esta sección se describen los requisitos del sistema. Para una mejor legibilidad del documento se han separado en requisitos de información, funcionales y no funcionales.

3.4.1. Requisitos de información

Aquí se especifican los requisitos de almacenamiento y de restricciones de información que se han identificado.

IRQ-01	***Gestión de Viaje***	
Versión	1.0	
Autores	Ángel Martín Velázquez (Universidad de Valladolid)	
Fuentes	Usuario final	
Objetivos asociados	OBJ–04 - Exportación de la información disponible OBJ-05 - Generación de estadísticas	
Requisitos asociados	UC-02 - Almacenamiento de la información UC-03 - Exportación de la información UC-04 - Visualización de estadísticas	
Descripción	El sistema deberá almacenar la información correspondiente a cada viaje. En concreto:	
Datos específicos _	ID Descripción Timestamp	
Tiempo de vida	**Medio**	**Máximo**
	N/A	N/A
Ocurrencias simult.	**Medio**	**Máximo**
	1	1
Importancia	Alta	
Urgencia	Media	
Estado	Validado	
Estabilidad	Alta	
Comentarios	El concepto Viaje, corresponde al periodo de tiempo que inicias la monitorización de la conducción hasta el final de dicho periodo, cuando se pulsa el botón de stop. Usualmente corresponderá con el final del viaje.	

Tabla 9 - Requisitos de información 01

IRQ-02	***Gestión de Secuencia***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)

Fuentes	Usuario final	
Objetivos asociados	OBJ–04 - Exportación de la información disponible OBJ-05 - Generación de estadísticas	
Requisitos asociados	UC-02 - Almacenamiento de la información UC-03 - Exportación de la información UC-04 - Visualización de estadísticas	
Descripción	El sistema deberá almacenar la información correspondiente a cada secuencia o conjunto de comandos que se consultan periódicamente. En concreto:	
Datos específicos _	ID Descripción Timestamp Identificador de viaje	
Tiempo de vida	**Medio**	**Máximo**
	N/A	N/A
Ocurrencias simult.	**Medio**	**Máximo**
	1	1
Importancia	Alta	
Urgencia	Media	
Estado	Validado	
Estabilidad	Alta	
Comentarios	El concepto Secuencia, corresponde al conjunto de comandos, o parámetros que son recuperados periódicamente, según la frecuencia definida. Por ejemplo, si la frecuencia con que queremos visualizar la información es de 1 segundo, a cada segundo se creará la agrupación o secuencia que englobará el valor de todos los parámetros consultados en ese instante.	

Tabla 10 - Requisitos de información 02

IRQ-03	***Gestión de Comando***	
Versión	1.0	
Autores	Ángel Martín Velázquez (Universidad de Valladolid)	
Fuentes	Usuario final	
Objetivos asociados	OBJ–04 - Exportación de la información disponible OBJ-05 - Generación de estadísticas	
Requisitos asociados	UC-02 - Almacenamiento de la información UC-03 - Exportación de la información UC-04 - Visualización de estadísticas	
Descripción	El sistema deberá almacenar la información correspondiente a cada comando. En concreto:	
Datos específicos _	ID Tipo de comando Valor del comando Valor en formato texto del comando Identificador de viaje Identificador de secuencia	
Tiempo de vida	**Medio**	**Máximo**
	N/A	N/A
Ocurrencias simult.	**Medio**	**Máximo**
	15	25

Importancia	Alta
Urgencia	Media
Estado	Validado
Estabilidad	Alta
Comentarios	El concepto Comando, corresponde a cada uno de los parámetros recuperados, que viene definido por el viaje y la secuencia a los que pertenece, y sobre todo por el tipo de comando al que corresponde y su valor, como por ejemplo puede ser las revoluciones por minuto del vehículo.

Tabla 11 - Requisitos de información 03

IRQ-04	***Gestión de Ruta***	
Versión	1.0	
Autores	Ángel Martín Velázquez (Universidad de Valladolid)	
Fuentes	Usuario final	
Objetivos asociados	OBJ–04 - Exportación de la información disponible	
Requisitos asociados	UC-02 - Almacenamiento de la información UC-03 - Exportación de la información	
Descripción	El sistema deberá almacenar la información correspondiente a la ruta (por posicionamiento GS) que ha seguido el vehículo. Es decir se almacena para cada secuencia, la localización del vehículo. Para ello se almacenan los siguientes valores:	
Datos específicos _	ID Latitud Longitud Timestamp Identificador de viaje Identificador de secuencia	
Tiempo de vida	**Medio**	**Máximo**
	N/A	N/A
Ocurrencias simult.	**Medio**	**Máximo**
	1	1
Importancia	Alta	
Urgencia	Media	
Estado	Validado	
Estabilidad	Alta	
Comentarios	En el concepto Ruta, se reúnen cada uno de los puntos de localización, para recuperados por el sensor GPS del dispositivo móvil, de esta manera se puede recuperar información de la distancia recorrida, así como reconstruir posteriormente la ruta seguida por el vehículo en cada viaje.	

Tabla 12 - Requisitos de información 04

3.4.2. Requisitos funcionales

En este apartado se pasan a detallar los requisitos funcionales. Para expresar dichos requisitos se ha elegido hacerlo mediante casos de uso, por ser una manera mucho más visual y directa, y que por tanto facilita la comprensión de

los mismos por parte del lector, frente a la manera tradicional de expresarlos, con el uso de listados y tablas.

3.4.2.1. Definición de los actores

En este apartado se identifican los posibles actores de la aplicación, y se incluyen en el siguiente listado de actores de casos de uso. Este listado se reduce a un único actor, que será en usuario genérico de la aplicación:

ACT–01	***Usuario***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Descripción	Este actor representa al único tipo de usuario que puede utilizar la aplicación, que es un usuario genérico, con acceso a toda la funcionalidad de la aplicación y que denominaremos Usuario
Comentarios	*N/A*

Tabla 13 - Actor 01

3.4.2.2. Diagramas de casos de uso

Un diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Éste representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. Por tanto, un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Cada caso de uso expresa una unidad coherente de funcionalidad y permite representar los requerimientos funcionales de manera muy gráfica.

Además del diagrama de caso de uso de cada uno de los requisitos funcionales y de su correspondiente cuadro de detalle, se complementará cada requerimiento con su diagrama de secuencia, con el fin de facilitar la comprensión del mismo.

El diagrama de Secuencia, permite observar la perspectiva cronológica de las interacciones, destacando el orden temporal de los mensajes. En concreto, muestra los objetos participantes y los mensajes que se intercambian entre ellos. Por el contrario, este diagrama no nos permite comprobar los enlaces que hay entre los objetos existentes, así pues, es un diagrama que complementa al diagrama de casos de uso.

En primer lugar, se expone el caso de uso general de la aplicación, para pasar después a detallar cada uno de los casos de uso correspondientes a las funcionalidades del sistema.

3.4.2.2.1. Diagramas de caso de uso general:

El caso de uso general que define la funcionalidad de todo el sistema es:

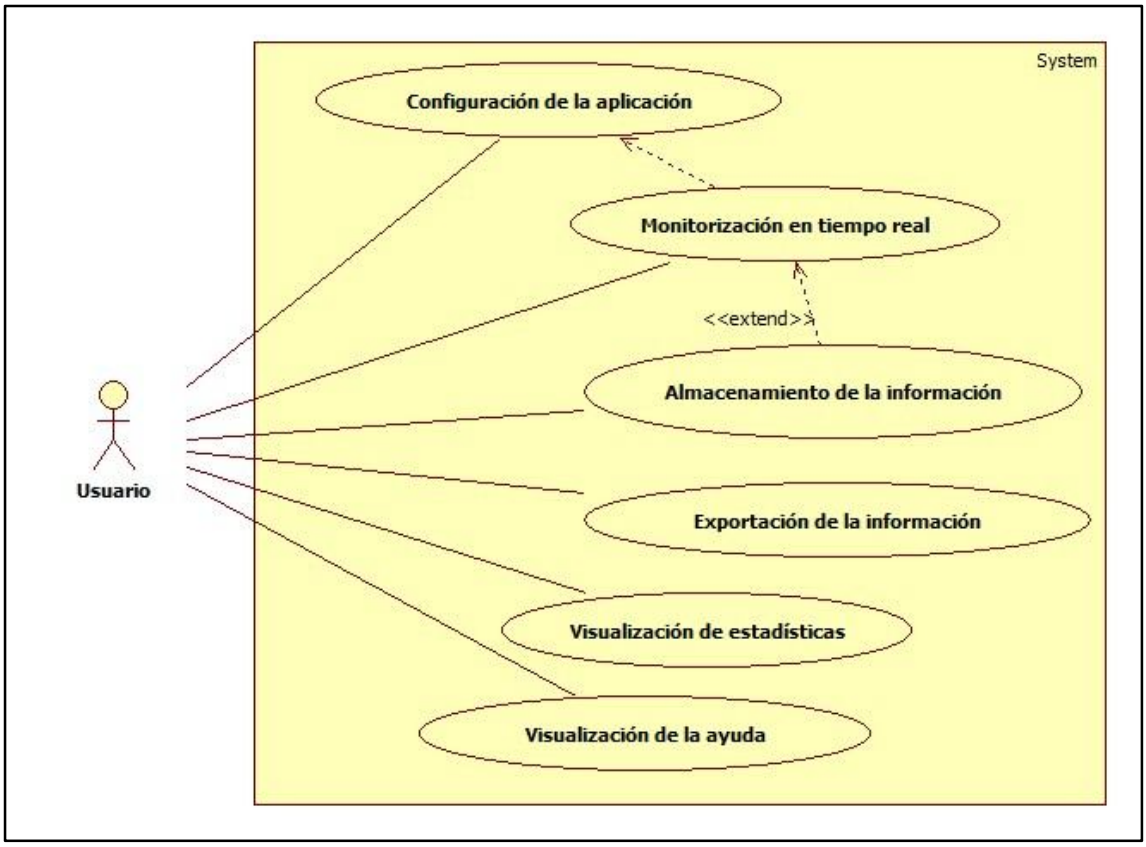


Figura 7- Diagrama de Caso de uso general

A continuación, se procede a explotar el caso de uso genérico, detallando los casos de uso de las correspondientes a las funcionalidades del sistema. Éstos definirán cada uno de los requisitos funcionales y son los siguientes:

3.4.2.2.2. Monitorización en tiempo real:

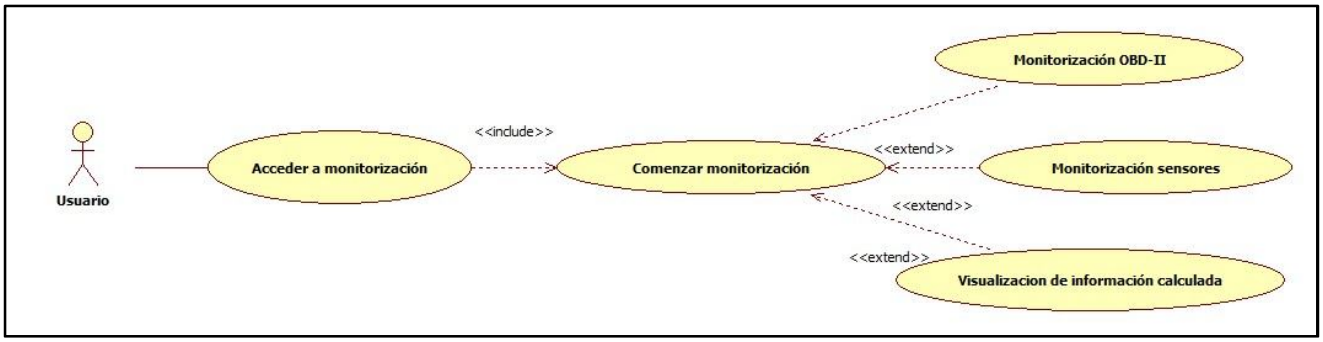


Figura 8 - UC01: Monitorización en tiempo real

UC-01	***Monitorización en tiempo real***	
Versión	1.0	
Autores	Ángel Martín Velázquez (Universidad de Valladolid)	
Fuentes	Usuario final	
Objetivos asociados	OBJ-01 - Visualización de información del vehículo OBJ-02 - Obtención de información de los sensores del terminal OBJ-03 - Generación de información calculada	
Requisitos asociados	NFR–01 - Conexión con el conector OBD por bluetooth	
Descripción	El sistema deberá mostrar por pantalla la información sobre la conducción cuando el usuario seleccione la opción de comenzar la monitorización. Esta información incluirá: - Parámetros obtenidos del OBD-II: * Velocidad instantánea * Revoluciones por minuto * Temperatura del refrigerante - Parámetros obtenidos de los sensores: * Brújula * Cronometro - Parámetros calculados: * Aviso de temperatura del refrigerante (el indicador cambia de color) * Aviso de cambio de marcha (en modo eficiente y en modo rendimiento) * Velocidad media del viaje * Porcentaje de tiempo con el vehículo encendido y detenido * Distancia recorrida.	
Precondición	Conector OBD conectado al vehículo (este debe estar en funcionamiento) Bluetooth del dispositivo móvil encendido, y asociado al conector OBD (acción disponible desde la configuración de la aplicación) Posicionamiento GPS del dispositivo habilitado.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario accede a la opción de monitorización
	p2	El sistema mostrará la pantalla de monitorización en tiempo real del vehículo.
	p3	Si el usuario pulsa el botón "START", se solicita al sistema que se comiencen la monitorización de la conducción.
	p4	El sistema comenzará a obtener y mostrar los parámetros en tiempo real de la conducción. En esta monitorización se incluyen los siguientes tipos de información: - Parámetros obtenidos por el OBD-II. - Parámetros obtenidos de los sensores del

		dispositivo. - Información calculada por el sistema a partir de los parámetros obtenidos previamente.
	p5	El sistema obtiene y muestra los parámetros de conducción anteriores, de forma periódica, según la frecuencia configurada.
	p6	Si el usuario pulsa el botón "STOP", se envía la orden de detener la monitorización.
	p7	La monitorización se detiene, y el sistema muestra de nuevo la pantalla de monitorización preparada para realizar cualquiera de las acciones disponibles.
Postcondición	N/A	
Excepciones	**Paso**	**Acción**
	p1	Si no se cumplen las precondiciones, el sistema mostrará un error y no se podrá continuar con la monitorización hasta que éstas se cumplan.
Rendimiento	**Paso**	**Cota de tiempo**
	p1	N/A
Frecuencia	1 vez por cada secuencia (frecuencia configurable definida en los parámetros de la aplicación)	
Importancia	Alta	
Urgencia	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	N/A	

Tabla 14 - Caso de uso 01

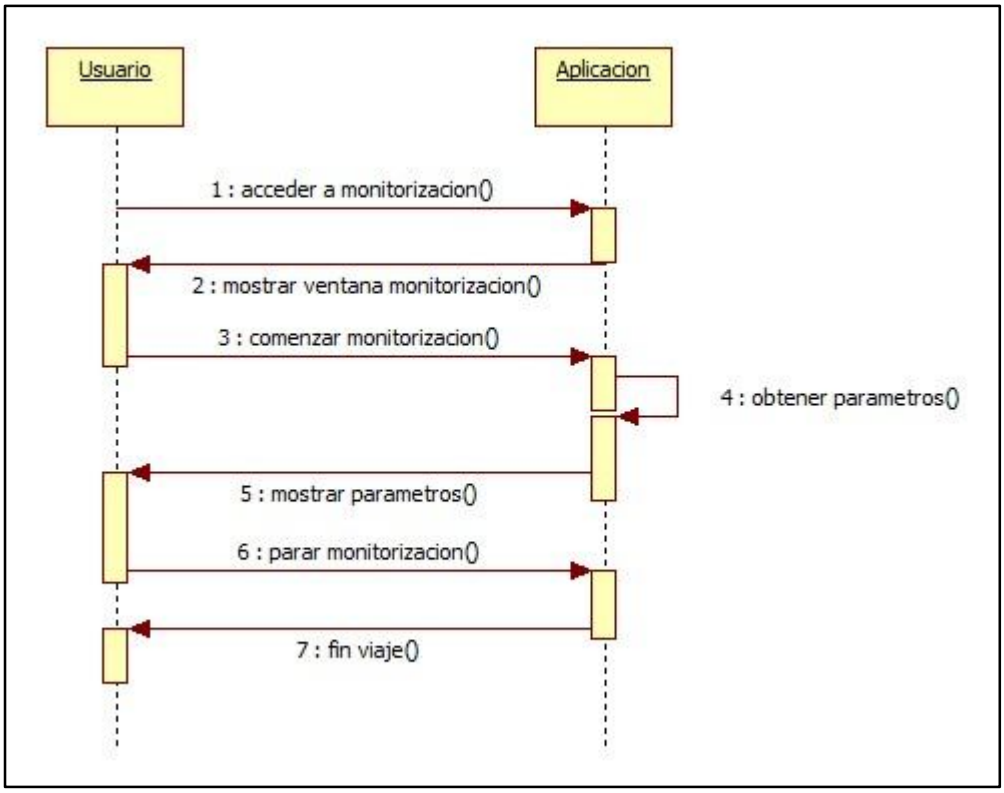


Figura 9 - Diagrama de secuencia UC01

3.4.2.2.3. Almacenamiento de la información:

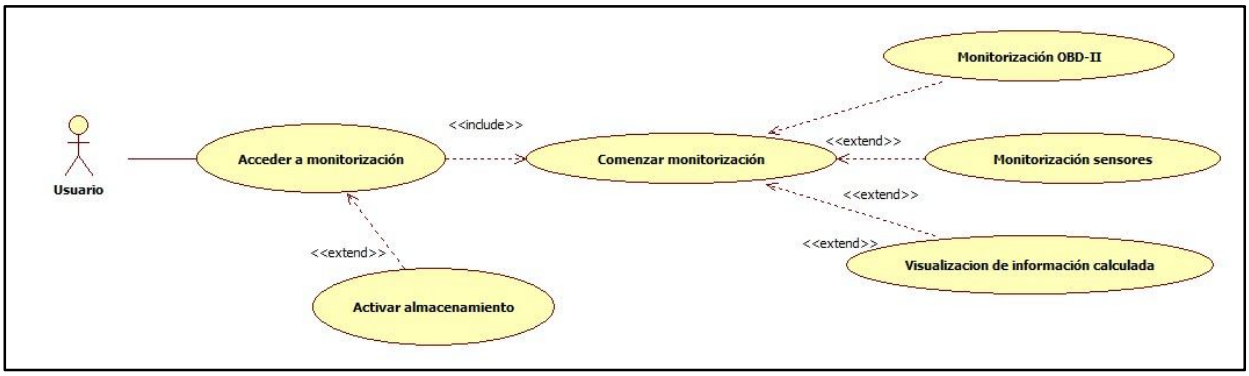


Figura 10 - UC02: Almacenamiento de la información

UC-02	***Almacenamiento de la información***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Objetivos asociados	OBJ-01 - Visualización de información del vehículo OBJ-02 - Obtención de información de los sensores del terminal OBJ-03 - Generación de información calculada
Requisitos asociados	IRQ-01 - Gestión de Viaje IRQ-02 - Gestión de Secuencia IRQ-03 - Gestión de Comando IRQ-04 - Gestión de Ruta UC-01 Monitorización en tiempo real
Descripción	Si se activa el registro en Base de datos, el sistema deberá almacenar en Base de datos la información sobre la conducción cuando el usuario seleccione la opción de comenzar la monitorización. Esta información incluirá: - Parámetros obtenidos del OBD-II: * Velocidad instantánea * Revoluciones por minuto * Temperatura del refrigerante * Mass Air Flow (MAF) * Engine Load (Carga del motor) - Parámetros obtenidos de los sensores: * Cronometro * Posicionamiento GPS. - Parámetros calculados: * Toda la información calculada mostrada. * Parámetros auxiliares.

Precondición	Conector OBD conectado al vehículo (este debe estar en funcionamiento) Bluetooth del dispositivo móvil encendido, y asociado al conector OBD (acción disponible desde la configuración de la aplicación) Posicionamiento GPS del dispositivo habilitado.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario accede a la opción de monitorización.
	p2	El sistema mostrará la pantalla de monitorización en tiempo real del vehículo (con la posibilidad de habilitar el registro de la información en base de datos)
	p3	Si el usuario solicita registrar en base de datos la información (pulsando el botón "DB OFF/DB ON"), al comenzar la monitorización, se guardará toda la información obtenida.
	p4	Queda activado el registro de la información en Base de datos en el sistema.
	p5	Si el usuario pulsa el botón "START", se solicita al sistema que se comiencen la monitorización de la conducción.
	p6	El sistema comenzará a obtener y mostrar los parámetros en tiempo real de la conducción. En esta monitorización se incluyen los siguientes tipos de información: - Parámetros obtenidos por el OBD-II. - Parámetros obtenidos de los sensores del dispositivo. - Información calculada por el sistema a partir de los parámetros obtenidos previamente.
	p7	El sistema obtiene los parámetros de conducción anteriores, de forma periódica, según la frecuencia configurada.
	p7	En cada ciclo, sistema solicita al Sistema Gestor de Base de Datos, que la información sea guardada.
	p8	El Sistema Gestor de Base de Datos, almacena la información requerida en base de datos.
	p9	El sistema muestra por pantalla los parámetros de conducción.
	p10	Si el usuario pulsa el botón "STOP", se envía la orden de detener la monitorización.
	p11	La monitorización se detiene, al igual que el almacenamiento de la información y el sistema muestra de nuevo la pantalla de monitorización preparada para realizar cualquiera de las acciones disponibles.
Postcondición	La información que se visualiza por pantalla, quedará además almacenada en Base de datos para su posterior explotación.	
Excepciones	**Paso**	**Acción**
	p1	Si no se cumplen las precondiciones, el sistema mostrará un error y no se podrá continuar con la

		monitorización hasta que éstas se cumplan.
Rendimiento	**Paso**	**Cota de tiempo**
	p1	N/A
Frecuencia	1 vez por cada secuencia (frecuencia configurable definida en los parámetros de la aplicación)	
Importancia	Alta	
Urgencia	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	N/A	

Tabla 15 - Caso de uso 02

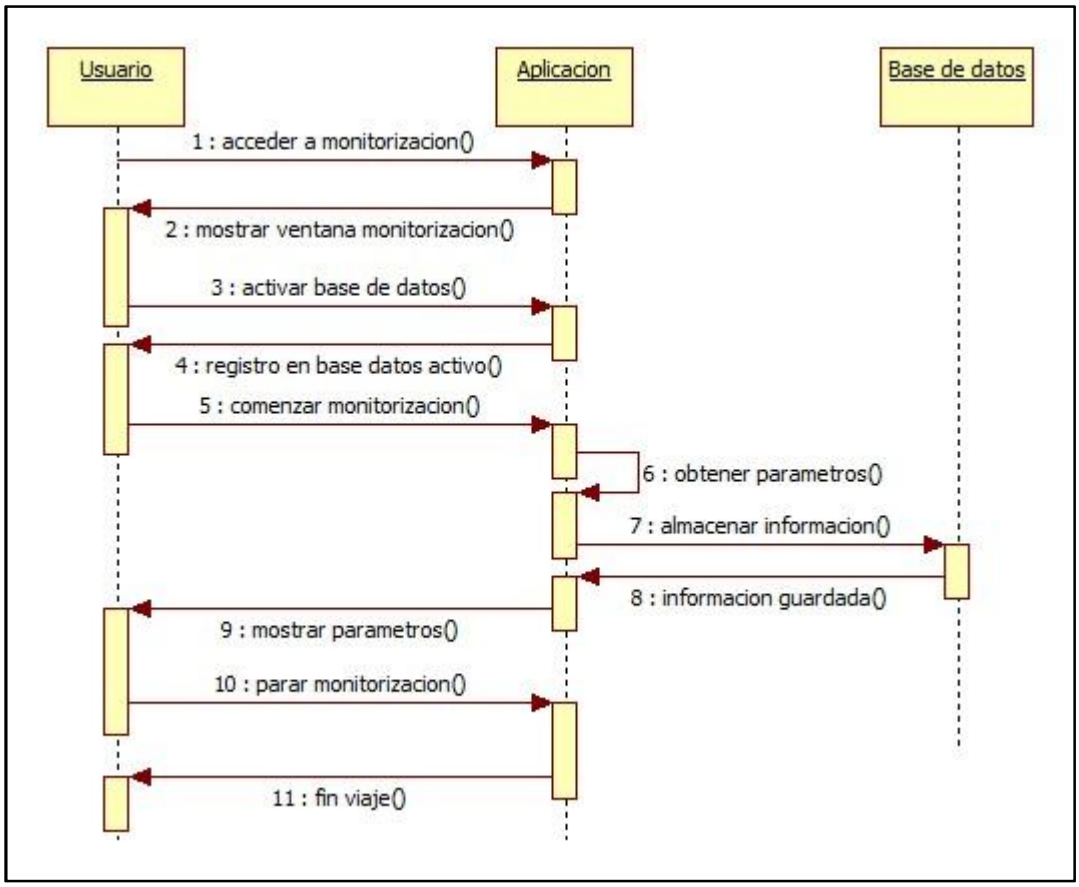


Figura 11 - Diagrama de secuencia UC02

3.4.2.2.4. Exportación de la información:

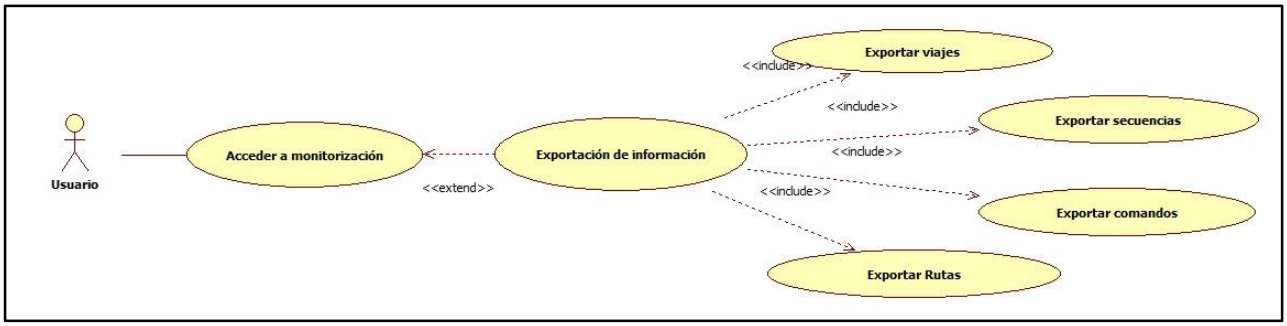


Figura 12 - UC03: Exportación de la información

UC-03	***Exportación de la información***	
Versión	1.0	
Autores	Ángel Martín Velázquez (Universidad de Valladolid)	
Fuentes	Usuario final	
Objetivos asociados	OBJ-04 - Exportación de la información disponible	
Requisitos asociados	IRQ-01 - Gestión de Viaje IRQ-02 - Gestión de Secuencia IRQ-03 - Gestión de Comando IRQ-04 - Gestión de Ruta	
Descripción	El sistema debe exportar la información de las tablas "travel" (viaje), "sequence" (secuencia), "command" (comando) y "route" (ruta), en ficheros con formato csv y con el mismo nombre que el aquí comentado, cuando el usuario realice la acción de exportar Base de datos.	
Precondición	Debe haber almacenados en base de datos, información de viajes, para que ésta pueda ser exportada.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario accede a la opción de monitorización.
	p2	El sistema mostrará la pantalla de monitorización en tiempo real del vehículo
	p3	Si el actor pulsa en botón "Download DB", se solicita al sistema la exportación de la base de datos
	p4	El sistema realiza la consulta, al Sistema Gestor de Base de Datos (SGBD) del contenido de las tablas: - Travel - Sequence - Command - Route
	p5	El SGBD, proporciona a la aplicación el contenido de las tablas consultadas.
	p6	El sistema genera en la memoria externa del dispositivo móvil, un fichero en formato csv, por cada una de las tablas descritas en el paso anterior, con su contenido completo.

	p7	Se muestra al usuario un mensaje para indicar que el proceso ha finalizado con éxito.
Postcondición	Se generan en la memoria externa del dispositivo, los ficheros en formato "csv", con la información de cada una de las tablas.	
Excepciones	**Paso**	**Acción**
	p1	Si surgiera algún problema al intentar escribir los ficheros, como por ejemplo que no se disponga de permisos de escritura o la memoria no estuviera disponible, el sistema mostrará un error y los ficheros no podrán ser generados.
Rendimiento	**Paso**	**Cota de tiempo**
	p1	N/A
Frecuencia	A petición del usuario.	
Importancia	Alta	
Urgencia	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	N/A	

Tabla 16 - Caso de uso 03

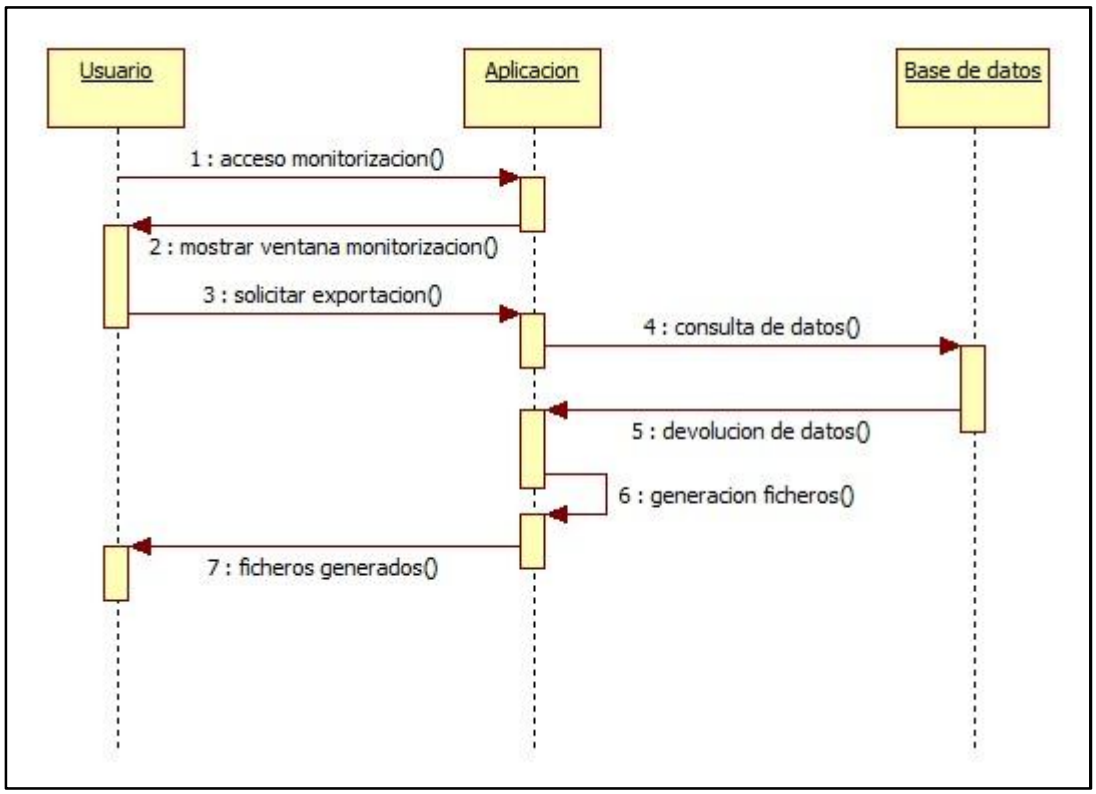


Figura 13 - Diagrama de secuencia UC03

3.4.2.2.5. Visualización de estadísticas:

Figura 14 - UC04: Visualización de estadísticas

UC-04	***Visualización de estadísticas***	
Versión	1.0	
Autores	Ángel Martín Velázquez (Universidad de Valladolid)	
Fuentes	Usuario final	
Objetivos asociados	OBJ-05 - Generación de estadísticas	
Requisitos asociados	IRQ-01 - Gestión de Viaje IRQ-02 - Gestión de Secuencia IRQ-03 - Gestión de Comando	
Descripción	El sistema deberá generar las gráficas estadísticas de los últimos viajes cuando el usuario lo solicite.	
Precondición	Debe haber almacenados en base de datos, información de viajes, para que ésta pueda ser mostrada.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario accede a la opción de monitorización.
	p2	El sistema mostrará la pantalla de monitorización en tiempo real del vehículo.
	p3	Si el actor pulsa en botón "Stats", se solicita al sistema la visualización de estadísticas.
	p4	El sistema realiza la consulta, al Sistema Gestor de Base de Datos (SGBD) de la información necesaria para generar las estadísticas.
	p5	El SGBD, proporciona a la aplicación la información solicitada sobre los últimos viajes.
	p6	El sistema genera las estadísticas.
	p7	Se muestra al usuario por pantalla la gráfica de estadísticas
Postcondición	Se visualizará por pantalla la gráfica seleccionada	
Excepciones	**Paso**	**Acción**
	p1	Si no hay información almacenada en Base de datos, la grafica no podrá ser mostrada.
Rendimiento	**Paso**	**Cota de tiempo**
	p1	N/A
Frecuencia	A petición del usuario.	
Importancia	Alta	
Urgencia	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	N/A	

Tabla 17 - Caso de uso 04

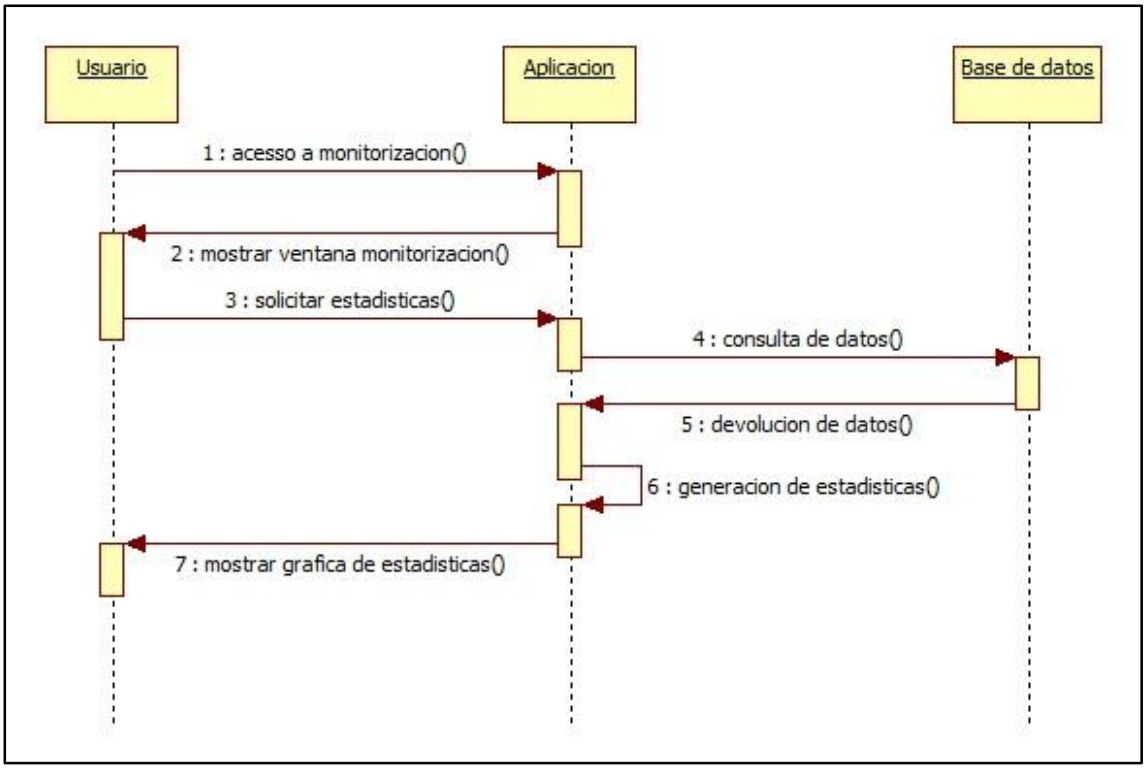


Figura 15 - Diagrama de secuencia UC04

3.4.2.2.6. Configuración de la aplicación:

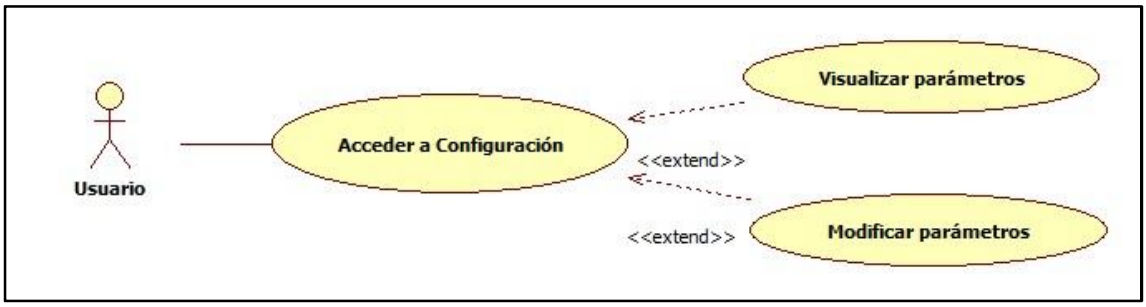


Figura 16 - UC05:Configuración de la aplicación

UC-05	***Configuración de la aplicación***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Objetivos asociados	OBJ-06 - Configuración del sistema
Requisitos asociados	N/A
Descripción	El sistema deberá permitir que se puedan configurar los siguientes parámetros por pantalla, - Dispositivo Bluetooth al que conectarse (para seleccionar el conector OBD-II) - Activar GPS.

	- Frecuencia de actualización (en segundos, solo números enteros, con un valor mínimo de 0, que equivale a 0,25 segundos entre secuencias) - Limite de revoluciones en la grafica. - Revoluciones para indicar el cambio de marcha en modo ECO. - Revoluciones máximas para indicar el cambio de marcha en modo POWER.	
Precondición	N/A	
Secuencia normal	**Paso**	**Acción**
	p1	El actor Usuario, accede a la opción "Settings" o configuración del sistema.
	p2	El sistema mostrará la pantalla de configuración, con los distintos parámetros configurables.
	p3	Si el usuario, pulsa sobre alguno de los parámetros mostrados aquí, se solicita acceder al valor de éste
	p4	El sistema muestra un cuadro de dialogo editable, donde se visualiza el valor del parámetro.
	p5	Por ser editable, el usuario puede modificar el valor del parámetro.
	p6	Se muestra el cuadro de dialogo con el valor modificado.
	p7	Si el usuario escoge la opción "Aceptar", se solicita al sistema que el parámetro sea modificado.
	p8	El sistema almacena el nuevo valor para el parámetro, y será tenido en cuenta en la próxima ejecución.
	p9	Si el usuario selecciona la opción "Cancelar", se solicita al sistema descartar el cambio.
	p10	El sistema mantendrá almacenado el valor anterior del parámetro.
Postcondición	Parámetros de la aplicación establecidos.	
Excepciones	**Paso**	**Acción**
	p1	Si el valor introducido al modificar alguno de los parámetros no es válido, por ejemplo, por formato no compatible o estar fuera de rango, el sistema mostrará un mensaje de error y se mantendrá el valor establecido previamente.
Rendimiento	**Paso**	**Cota de tiempo**
	p1	N/A
Frecuencia	A petición del usuario.	
Importancia	Alta	
Urgencia	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	N/A	

Tabla 18 - Caso de uso 05

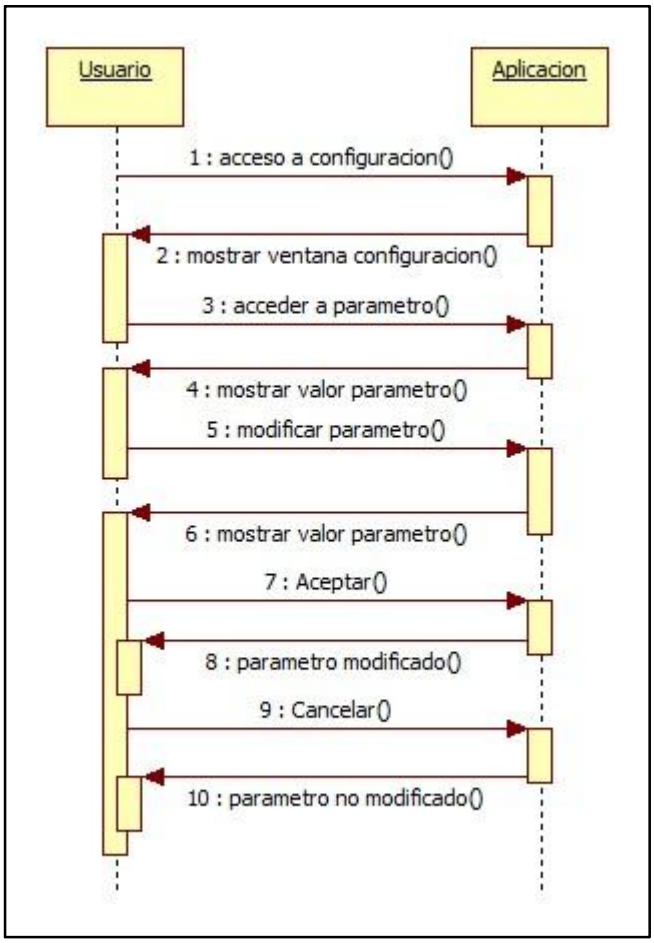


Figura 17 - Diagrama de secuencia UC05

3.4.2.2.7. Visualización de la ayuda:

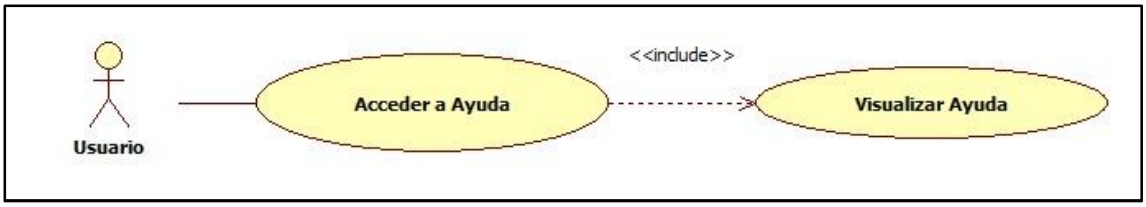


Figura 18 - UC06: Visualización de la ayuda

UC-06	***Visualización de la ayuda***	
Versión	1.0	
Autores	Ángel Martín Velázquez (Universidad de Valladolid)	
Fuentes	Usuario final	
Objetivos asociados	OBJ-07 - Ayuda	
Requisitos asociados	N/A	
Descripción	El sistema deberá mostrar documentación de ayuda sobre el manejo de la aplicación, cuando el usuario lo solicite.	
Precondición	N/A	
Secuencia normal	**Paso**	**Acción**

	p1	El usuario accede a la pantalla de ayuda.
	p2	El sistema mostrará por pantalla la documentación de ayuda sobre el manejo de la aplicación, para poder resolver las posibles dudas del usuario.
Postcondición	N/A	
Excepciones	**Paso**	**Acción**
	p1	N/A
Rendimiento	**Paso**	**Cota de tiempo**
	p1	N/A
Frecuencia	A petición del usuario	
Importancia	Alta	
Urgencia	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	N/A	

Tabla 19 - Caso de uso 06

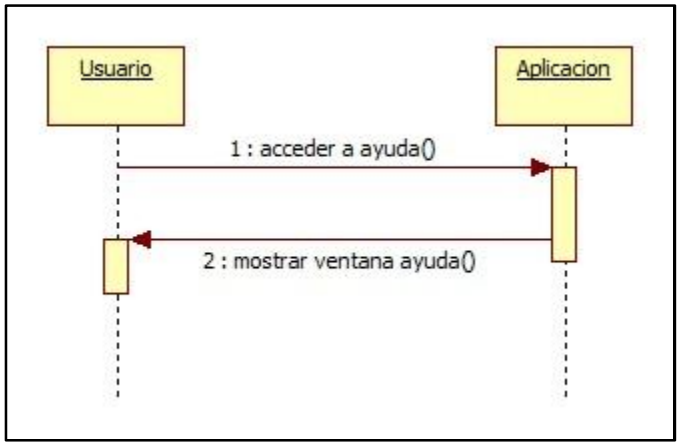


Figura 19 - Diagrama de secuencia UC06

3.4.3. Requisitos no funcionales

En esta subsección se detallan los requisitos no funcionales del sistema, identificados durante el análisis. Son los siguientes:

NFR–01	***Conexión con el conector OBD por bluetooth***
Versión	1.0
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Fuentes	Usuario final
Objetivos asociados	OBJ-01 - Visualización de información del vehículo
Requisitos asociados _	UC-01 Monitorización en tiempo real
Descripción	El sistema deberá ser capaz de establecer y mantener la comunicación con el conector OBD a través de una conexión Bluetooth
Importancia	Alta

Urgencia	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	N/A

Tabla 20 - Requisito no funcional 01

3.5 MATRIZ DE RASTREABILIDAD DE OBJETIVOS Y REQUISITOS

Para tener una visión general sobre qué requisitos están asociados a qué objetivos, a continuación se muestra una matriz objetivo – requisito donde queda clara esta relación:

	OBJ–01	OBJ–02	OBJ–03	OBJ–04	OBJ–05	OBJ–06	OBJ–07
IRQ–01				•	•		
IRQ–02				•	•		
IRQ–03				•	•		
IRQ–04				•			
UC–01	•	•	•				
UC–02	•	•	•				
UC–03				•			
UC–04					•		
UC–05						•	
UC–06							•
NFR–01	•						

Tabla 21 - Matriz de rastreabilidad

3.6 RESUMEN

En este apartado se proporciona una visión global de los requisitos del sistema. Contiene una tabla con los identificadores y nombres descriptivos de cada uno de los elementos principales del documento: Objetivos, Requisitos de Información, Requisitos funcionales (Casos de Uso) y Requisitos no funcionales.

TIPO	ID	Descripción
OBJETIVOS	OBJ–01	Visualización de información del vehículo
	OBJ–02	Obtención de información de los sensores del terminal
	OBJ–03	Generación de información calculada
	OBJ–04	Exportación de la información disponible

	OBJ–05	Generación de estadísticas
	OBJ–06	Configuración del sistema
	OBJ–07	Ayuda
REQUISITOS INFORMACION	IRQ–01	Gestión de Viaje
	IRQ–02	Gestión de Secuencia
	IRQ–03	Gestión de Comando
	IRQ–04	Gestión de Ruta
REQUISITOS FUNCIONALES (CASOS DE USO)	UC–01	Monitorización en tiempo real
	UC–02	Almacenamiento de la información
	UC–03	Exportación de la información
	UC–04	Visualización de estadísticas
	UC–05	Configuración de la aplicación
	UC–06	Visualización de la ayuda
REQUISITOS NO FUNCIONALES	NFR–01	Conexión con el conector OBD por bluetooth

Tabla 22 - Resumen del Análisis

CAPÍTULO 4

DISEÑO DEL SISTEMA

CAPÍTULO 4
DISEÑO DEL SISTEMA

4.1 INTRODUCCIÓN

En este apartado, se abordarán todos los puntos relativos al diseño del proyecto software que abarcará, desde la definición de las clases sobre las que se ha desarrollado la aplicación hasta el especificación y diseño de la base de datos que se requiere para almacenar la información pertinente.

Así pues, en la etapa de diseño lo que se intenta hacer es acercar la idea conceptual que se ha obtenido de los requerimientos obtenidos en la fase de análisis, a un modelo más físico y más próximo a la futura implementación, que servirá de base para ésta.

4.2 DIAGRAMAS DE CLASES

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Para comprender el diagrama de clases, conviene recordar algunos conceptos, que a continuación exponemos:

- Atributos: son las propiedades o características que corresponden a un objeto, como color, material, ...

- Métodos: son las actividades o acciones que permite hacer el objeto, como por ejemplo abrir, cerrar, ...

- Interfaz: es un conjunto de operaciones y/o propiedades que definen el comportamiento de un objeto. En ella se especifica qué se debe hacer pero no su implementación.

- Herencia: la herencia facilita la creación de objetos a partir de otros ya existentes e implica que una clase hija obtiene todo el comportamiento (métodos) y eventualmente los atributos (variables) de su clase padre. Es la relación entre una clase general y otra clase más específica, por ejemplo: las clases Cliente y Empleado, podrían ser subclases de la clase Persona y heredarían de ella su comportamiento y características, pudiendo a su vez extender su funcionalidad.

Una vez recordados estos conceptos, es necesario destacar que este Trabajo Fin de Grado está basado en un proyecto ya existente de software libre. Esto implica que el programa base ya disponía de multitud de clases y no todas ellas han sido modificadas. Por este motivo y persiguiendo una mejor organización del documento, se ha hecho un

primer diagrama con los subsistemas de la aplicación, indicando cuáles de ellos están impactados por nuestro proyecto. El posterior diagrama de clases, corresponderá únicamente a estos subsistemas afectados.

Posteriormente, se realizará una descripción detallada de cada una de esas clases, en la que se mostrarán los métodos y atributos de los que se componen y que permitirá terminar de entender la idea que se plantea para abordar el problema propuesto.

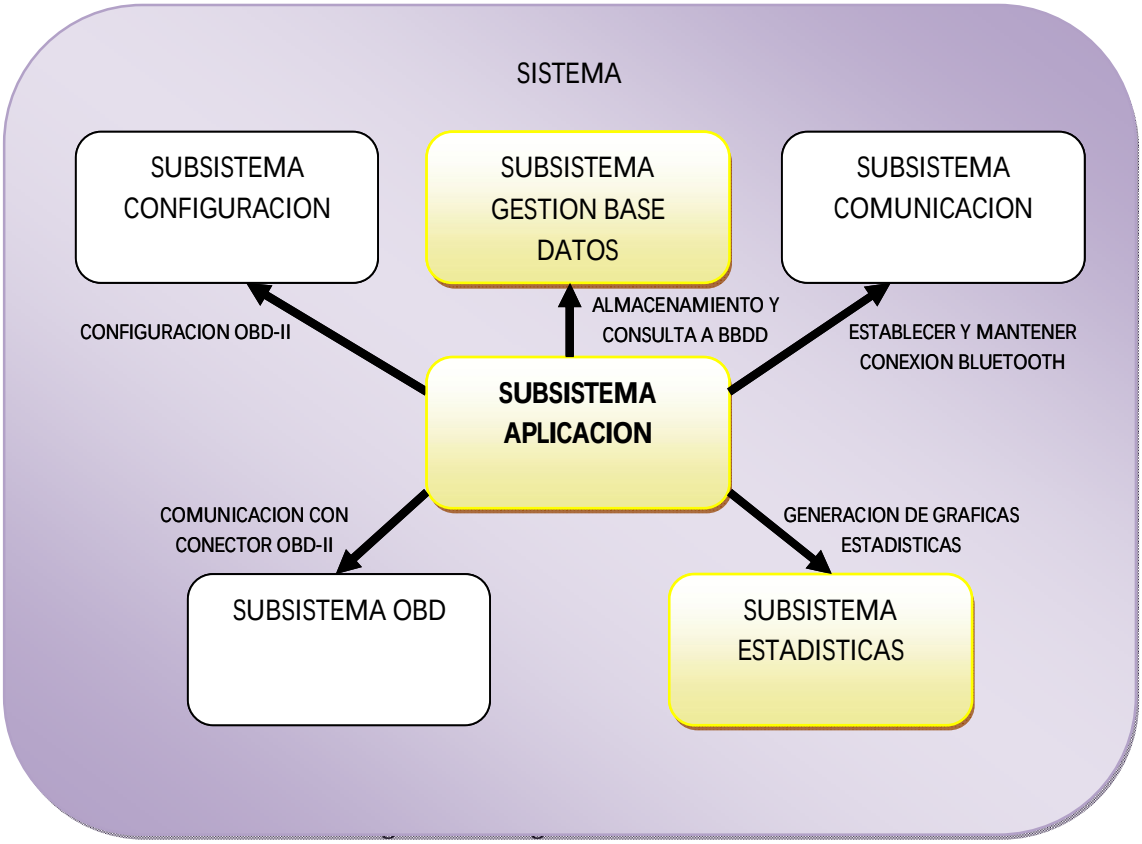
En la fase de diseño, previa a la implementación del proyecto, se obtuvo un diagrama inicial sobre el que se tuvieron que realizar ciertas modificaciones, para adecuar el diseño a las restricciones y requisitos tanto estructurales, como de comportamiento, de la aplicación. Un claro ejemplo, son las clases destinadas a la visualización de estadísticas, que por restricciones de Android y de la solución escogida, se tuvo que rediseñar para ampliarlo, si bien la estructura general no se vio alterada.

La versión aquí expuesta corresponde al diagrama de clases definitivo, tal y como se ha llevado a la implementación.

4.2.1. Diseño general

A continuación se muestra una visión global de la estructura de todo el sistema. Aquí se pueden observar los distintos subsistemas funcionales del sistema y las relaciones entre ellos.

Como se ha comentado anteriormente, la solución para la consecución del proyecto partía de una aplicación base. En esta visión global se incluyen tanto los subsistemas de la aplicación base, como los nuevos o modificados en el presente proyecto. Para diferenciarlos, se indican con fondo amarillo aquellos que se hayan tenido que modificar, desarrollar de cero o incluir:



Para poder identificar cada subsistema y la funcionalidad que cubre, se describe brevemente cada uno de ellos:

- Subsistema de Aplicación:
  Es el subsistema principal. Inicialmente ya existía en la aplicación base, pero modificando éste es donde más se ha trabajado. Éste es el que ofrece la mayor parte de las funcionalidades, así como el interfaz de usuario, y se apoya en el resto de los subsistemas incluidos en el diagrama, para poder proporcionarlas.

- Subsistema de Gestión de Base de Datos:
  Este subsistema se ha realizado desde cero, con el objetivo de encapsular la gestión de la pequeña base de datos incluida en el sistema. Con ello se pretende liberar al subsistema principal, de realizar esta gestión, así como de permitir una mayor reutilización.

- Subsistema de Estadísticas:
  Con el objetivo de poder ofrecer unas gráficas estadísticas de calidad a partir de la información almacenada en Base de Datos, se incluyó este subsistema de estadísticas. Éste se basa en un proyecto de software llamado "Androidplot" que se ha incluido en el proyecto y que es invocado desde el subsistema de aplicación para mostrar las gráficas estadísticas al usuario. La documentación sobre este software está disponible para el público en su web (disponible en la bibliografía), por lo que se desestima incluirlo en el presente estudio.

- Subsistema de Comunicación:

El subsistema de comunicación es el encargado de establecer la comunicación Bluetooth y mantenerla durante el periodo en el que la monitorización se esté ejecutando. Es uno de los puntos claves que permiten cubrir los objetivos del proyecto. Este subsistema es proporcionado por la aplicación original, por lo que se desestima incluirlo en el presente estudio.

- Subsistema OBD:

  El subsistema OBD, permite la comunicación con el conector OBD-II, utilizando en canal bluetooth establecido por el sistema de comunicación (se gestiona de forma centralizada desde el subsistema de aplicación). Este subsistema permite extraer y entender la información del vehículo proporcionada por el conector bajo el protocolo OBD-II. Junto al subsistema de comunicación son piezas claves para conseguir los objetivos del proyecto. Está proporcionado también por el sistema original y este hecho fue decisivo para seleccionar la aplicación base, pues de otra manera el proyecto no podría abordarse en un plazo razonable. Se desestima incluirlo en el estudio, para mantener la claridad del trabajo a realizar.

- Subsistema de Configuración:

  Es un subsistema muy básico para la configuración de los comandos soportados en la aplicación, que serán un subconjunto de los soportados por el subsistema OBD. Es también ofrecido por el sistema base y por ello, se desestima el incluirlo en el presente estudio.

Una vez descritos cada uno de los subsistemas que componen la aplicación completa, vamos a centrarnos en los dos subsistemas que tienen realmente impacto en el Trabajo Fin de Grado, que serían el Subsistema de Aplicación y el Subsistema de Base de Datos. Se presenta a continuación el diagrama de clases, donde tan solo se incluyen las clases afectadas:

4.2.2. Diagrama de clases

En este punto veremos el diagrama de clases simplificado para una mayor claridad, con todas las clases que se ven implicadas en el TFG y las relaciones entre ellas. Tras el diagrama se incluirá el detalle de cada una de estas clases, donde se podrán observar detenidamente, para cada una de ellas, sus métodos y funciones.

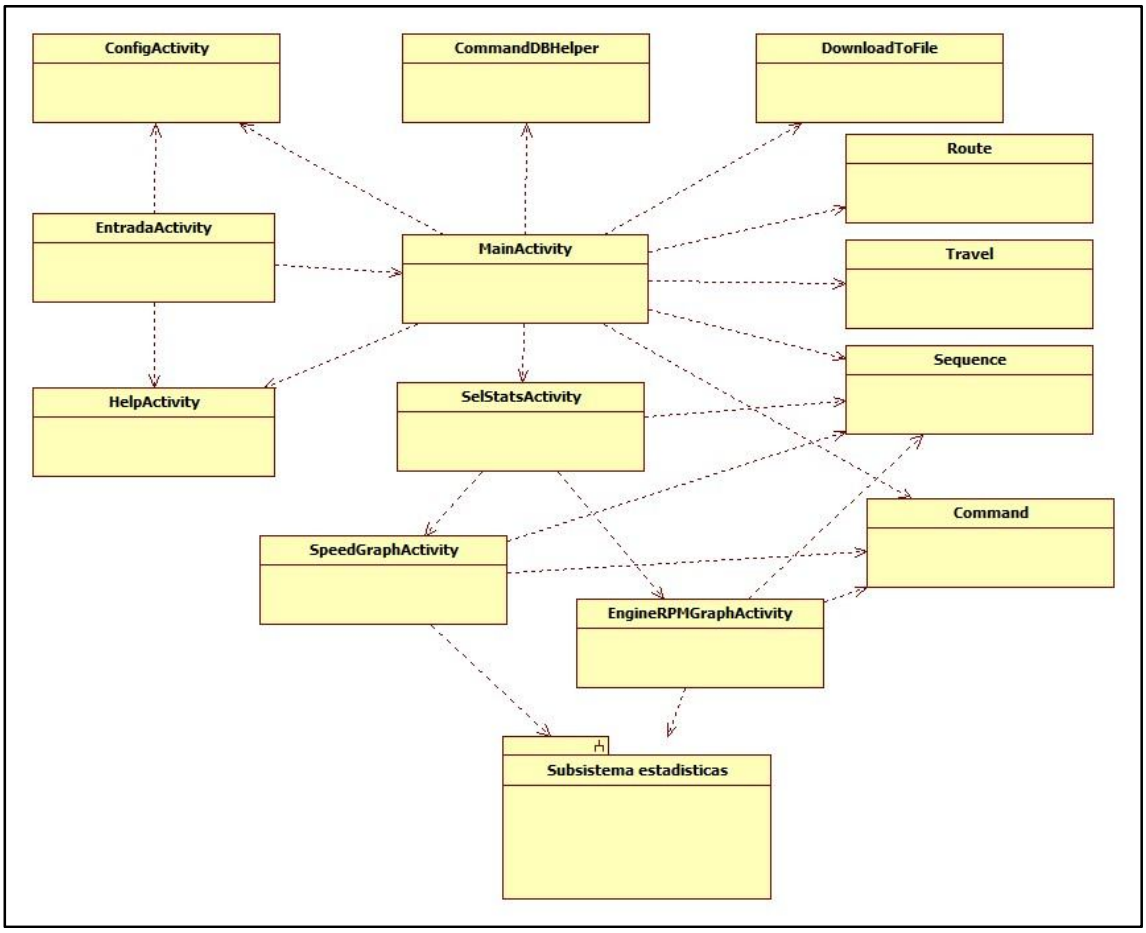


Figura 21 - Diagrama de clases implicadas

4.2.2.1. Clase CommandDBHelper

CommandDBHelper
sqlCreate1 : String sqlCreate2 : String sqlCreate3 : String sqlCreate4 : String
CommandDBHelper() onCreate() onUpgrade()

Figura 22 - Clase CommandDBHelper

La clase *CommandDBHelper* pertenece al subsistema de gestión de base de datos y hereda de la clase *SQLiteOpenHelper* de Android. Esta clase permite la creación inicial de la base de datos y la recreación de la misma, cuando se detecta una nueva versión; y es una clase indispensable para el tratamiento de bases de datos SQLite en cualquier proyecto Android.

Tiene los siguientes atributos:

- **sqlCreate1** : String, contiene la sentencia de creación de la tabla Travel.
- **sqlCreate2** : String, contiene la sentencia de creación de la tabla Sequence.
- **sqlCreate3** : String, contiene la sentencia de creación de la tabla Command.
- **sqlCreate4** : String, contiene la sentencia de creación de la tabla Route.

Los métodos de los que dispone son:

- **CommandDBHelper()**, es el constructor de la clase.
- **onCreate()**, ejecuta en la base de datos las sentencias de creación de las tablas, definidas en los atributos.
- **onUpgrade()**, realiza una recreación de todas la tablas cuando se define una nueva versión de la base de datos.

4.2.2.2. Clase Travel

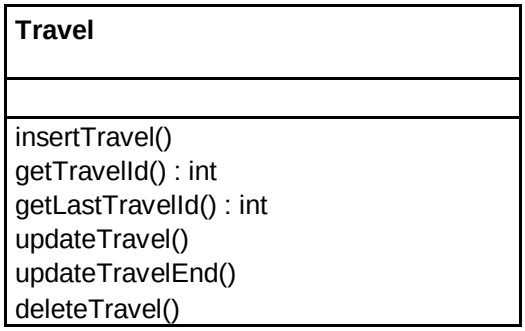


Figura 23 - Clase Travel

La clase *Travel* pertenece también al subsistema de gestión de base de datos y es la encargada de proporcionar las distintas operaciones que se pueden realizar con base de datos, relativas a viajes.

Esta clase no tiene atributos.

Tiene los siguientes métodos:

- **insertTravel()**, realiza la inserción del viaje en base de datos.
- **getTravelId()**: int, recupera el identificador del viaje a partir de la descripción del mismo.
- **getLastTravelId()**: int, devuelve el identificador del último viaje.
- **updateTravel()**, a partir del identificador, se actualiza su descripción.
- **updateTravelEnd(id : int, dbActive : boolean)**, a partir del identificador, se actualiza su fecha/hora de fin.
- **deleteTravel()**, es capaz de borrar en base de datos un viaje a partir de su identificador.

4.2.2.3. Clase Sequence

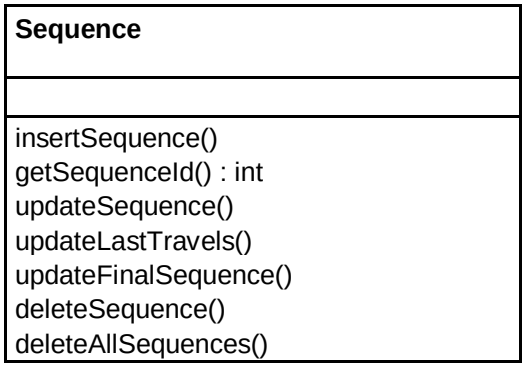


Figura 24 - Clase Sequence

La clase *Sequence* pertenece también al subsistema de gestión de base de datos y es la encargada de proporcionar las distintas operaciones que se pueden realizar con base de datos, relativas a las secuencias de comandos.

Esta clase no tiene atributos.

Tiene los siguientes métodos:

- **insertSequence(travel : int, description : String)**, realiza la inserción de la secuencia en base de datos.
- **getSequenceId()**: int, recupera el identificador de la secuencia a partir de su descripción.
- **updateSequence()**, a partir del identificador se actualiza su descripción.
- **updateLastTravels()**, se marcan como finales las últimas secuencias de los últimos viajes, para poder obtener las estadísticas.
- **updateFinalSequence()**, se marca como final la última secuencia de un viaje dado.
- **deleteSequence()**, permite borrar una secuencia a partir de su identificador.
- **deleteAllSequences()**, permite borrar todas las secuencias de un viaje.

4.2.2.4. Clase Command

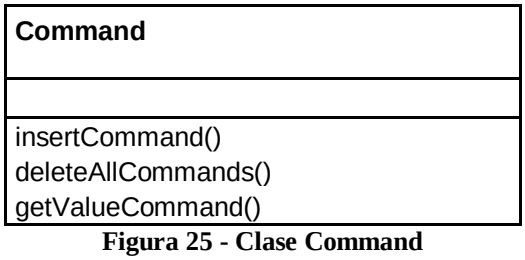


Figura 25 - Clase Command

La clase *Command* pertenece también al subsistema de gestión de base de datos y es la encargada de proporcionar las distintas operaciones que se pueden realizar con base de datos, relativas a los comandos (el comando corresponde a cada uno de los parámetros de conducción obtenidos por la aplicación).

Esta clase no tiene atributos.

Tiene los siguientes métodos:

- **insertCommand()**, inserta el comando en base de datos.
- **deleteAllCommands()**, permite borrar todos los comandos de un viaje.
- **getValueCommand(travelID : int,sequenceID : int, commadType : String)**, devuelve el valor de un comando. Para ello, es necesario indicar el identificador del viaje y el de secuencia y el tipo del comando que queremos recuperar.

4.2.2.5. Clase Route

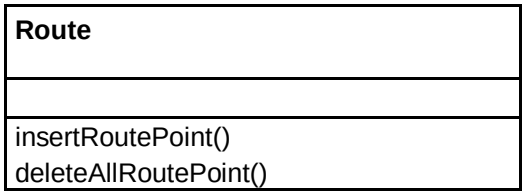


Figura 26 - Clase Route

La clase *Route* pertenece también al subsistema de gestión de base de datos y es la encargada de proporcionar las distintas operaciones que se pueden realizar con base de datos, relativas al posicionamiento GPS, con la que se podrá definir una ruta a partir de sus puntos de posicionamiento.

Esta clase no tiene atributos.

Tiene los siguientes métodos:

- **insertRoutePoint()**, inserta un punto de la ruta en base de datos.
- **deleteAllRoutePoint()**, permite borrar toda la ruta de un viaje, es decir, elimina todos los puntos de ruta asociados a ese viaje.

4.2.2.6. Clase DownloadToFile

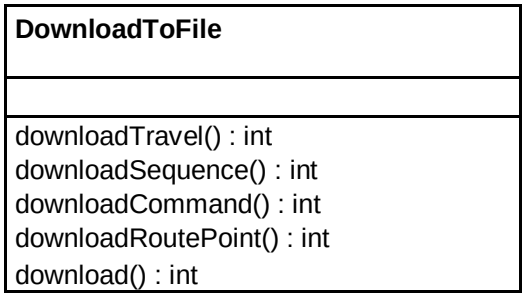


Figura 27 - Clase DownloadToFile

La clase *DownloadToFile* pertenece también al subsistema de gestión de base de datos y es la encargada de realizar la exportación de la b ase de datos, a la memoria del dispositivo, con el objeto de poder compartirlo y poder tratar y analizar la información obtenida en cualquier ordenador.

Esta clase no tiene atributos.

Tiene los siguientes métodos:

- **downloadTravel(): int**, descarga en la memoria del dipositivo un fichero con formato csv, con toda la información de la tabla *Travel*. Se devuelve un valor entero para indicar si el proceso ha finalizado correctamente.
- **downloadSequence(): int**, descarga en la memoria del dipositivo un fichero con formato csv, con toda la información de la tabla *Sequence*. Se devuelve un valor entero para indicar si el proceso ha finalizado correctamente.
- **downloadCommand(): int**, descarga en la memoria del dipositivo un fichero con formato csv, con toda la información de la tabla *Command*. Se devuelve un valor entero para indicar si el proceso ha finalizado correctamente.
- **downloadRoutePoint(): int**, descarga en la memoria del dipositivo un fichero con formato csv, con toda la información de la tabla *Route*. Se devuelve un valor entero para indicar si el proceso ha finalizado correctamente.
- **download(): int**, se invoca a los anteriores métodos para realizar la exportación de toda la base de datos, devolviendo un valor entero para indicar si el proceso ha finalizado correctamente.

4.2.2.7. Clase EntradaActivity

EntradaActivity
onCreate() entrar() onClickSettings() onClickHelp()

Figura 28 - Clase EntradaActivity

La clase *EntradaActivity* pertenece al subsistema de aplicaciones y hereda de la clase *Activity* de Android (con la cual se definen las pantallas de interfaz de usuario).

Esta clase proporciona la pantalla de acceso al sistema y permite acceder a las distintas opciones que ofrece la aplicación, que son el acceso a la monitorización en tiempo real, a la pantalla de configuración, y por último a la ayuda de la aplicación.

Esta clase no tiene atributos.

Tiene los siguientes métodos:

- **onCreate()**, generación de la pantalla de interfaz de usuario.
- **entrar()**, permite acceder a la pantalla de monitorización.
- **onClickSettings()**, permite el acceso a la pantalla de configuración.
- **onClickHelp()**, permite el acceso a la pantalla de ayuda.

4.2.2.8. Clase ConfigActivity

ConfigActivity
BLUETOOTH_LIST_KEY : String UPLOAD_URL_KEY : String UPLOAD_DATA_KEY : String UPDATE_PERIOD_KEY : String VEHICLE_ID_KEY : String ENGINE_DISPLACEMENT_KEY : String VOLUMETRIC_EFFICIENCY_KEY : String IMPERIAL_UNITS_KEY : String COMMANDS_SCREEN_KEY : String ENABLE_GPS_KEY : String MAX_FUEL_ECON_KEY : String CONFIG_READER_KEY : String MAX_REVS_SPIRITED : String MAX_REVS_ECO : String REVS_LIMIT : String TOAST_VISIBILITY : String

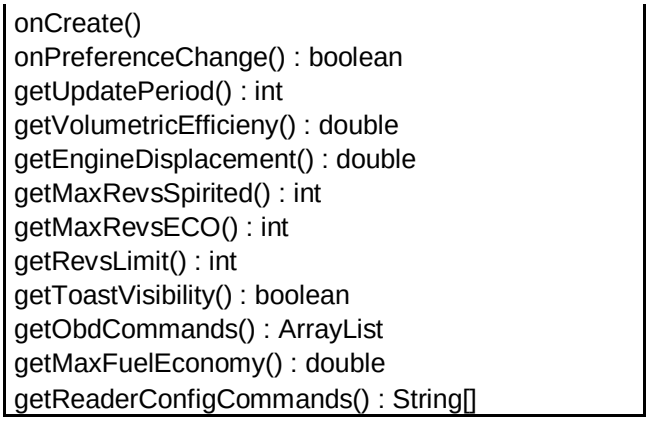


Figura 29 - Clase ConfigActivity

La clase *ConfigActivity* pertenece al subsistema de aplicaciones y hereda de la clase *PreferenceActivity* de Android (con la cual se definen las pantallas de interfaz de usuario) y además implementa el interfaz de Android *OnPreferenceChangeListener*.

Esta clase proporciona la pantalla de preferencias del sistema y era proporcionada por la aplicación base. Ha tenido que ser modificada para adaptarla a las necesidades actuales.

Esta clase tiene los siguientes atributos, si bien no se explicarán en detalle, porque son variables que contienen los identificadores de cada concepto de los incluidos en preferencias.

- **BLUETOOTH_LIST_KEY**: String.
- **UPLOAD_URL_KEY**: String.
- **UPLOAD_DATA_KEY**: String.
- **UPDATE_PERIOD_KEY**: String.
- **VEHICLE_ID_KEY**: String.
- **ENGINE_DISPLACEMENT_KEY**: String.
- **VOLUMETRIC_EFFICIENCY_KEY**: String.
- **IMPERIAL_UNITS_KEY**: String.
- **COMMANDS_SCREEN_KEY**: String.
- **ENABLE_GPS_KEY**: String.
- **MAX_FUEL_ECON_KEY**: String.
- **CONFIG_READER_KEY**: String.
- **MAX_REVS_SPIRITED**: String.
- **MAX_REVS_ECO**: String.
- **REVS_LIMIT**: String.
- **TOAST_VISIBILITY**: String.

Tiene los siguientes métodos:

- **onCreate()**, creación de la pantalla de preferencias.
- **onPreferenceChange()**: boolean, este método valida el valor introducido cuando una de las preferencias es modificada.
- **getUpdatePeriod()**: int, devuelve el periodo de actualización definido en las preferencias.

- **getVolumetricEfficieny()**: double, devuelve el valor de eficiencia volumétrica definido en las preferencias.
- **getEngineDisplacement()**: double, devuelve el valor del desplazamiento de motor definido en las preferencias.
- **getMaxRevsSpirited()**: int, devuelve el valor de revoluciones máximas para indicar el cambio de marchas cuando se busca el mayor rendimiento del motor, definido en las preferencias.
- **getMaxRevsECO()**: int, devuelve el valor de revoluciones máximas para indicar el cambio de marchas cuando se busca la mayor eficiencia del motor, definido en las preferencias.
- **getRevsLimit()**: int, devuelve el valor de revoluciones máximas para establecer el límite en la barra de revoluciones de la pantalla de monitorización, definido en las preferencias.
- **getToastVisibility()**: boolean, devuelve el valor de visibilidad de los mensajes de depuración del sistema.
- **getObdCommands()**: ArrayList, devuelve la lista de comandos del OBD-II que la aplicación puede tratar.
- **getMaxFuelEconomy()**: double, devuelve el valor de Economía de combustible máxima definido en las preferencias.
- **getReaderConfigCommands()**: String[], la lista de claves de configuración del lector OBD-II.

4.2.2.9. Clase HelpActivity

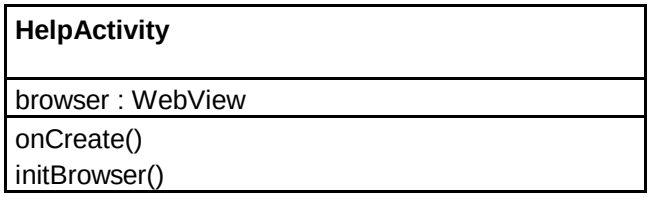


Figura 30 - Clase HelpActivity

La clase *HelpActivity* pertenece al subsistema de aplicaciones y hereda de la clase *Activity* de Android (con la cual se definen las pantallas de interfaz de usuario).

Esta clase proporciona la pantalla de ayuda al usuario donde de incluirá un manual de uso y configuración de la aplicación, para resolver las dudas que el usuario pueda tener.

Esta clase tiene los siguientes atributos:

- **browser: WebView**, es el navegador donde se podrá visualizar el manual de uso.

Tiene los siguientes métodos:

- **onCreate()**, creación y visualización de la pantalla de ayuda.
- **initBrowser()**, inicializar y mostrar el navegador, donde se podrá ver la ayuda.

4.2.2.10. Clase MainActivity

MainActivity
mHandler : Handler mListener : IPostListener mServiceIntent : Intent mServiceConnection : ObdGatewayServiceConnection sensorManager : SensorManager orientSensor : Sensor locationListener : LocationListener locationManager : LocationManager locater : MyLocater powerManager : PowerManager wakeLock : WakeLock orientListener : SensorEventListener mQueueCommands : Runnable pbRevoluciones : ProgressBar mChronometer : Chronometer db : SQLiteDatabase db_cerrada : boolean logFlag : boolean dbActiveFlag : boolean vToastVisibility : boolean idTravel : int descTravel : String idSequence : int descSequence : String idCommand : int speed : int maf : double ltft : float equivRatio : double rpm : int speedAccumulator : float speed0Accumulator : float speedCounter : float speedAv : float percStopped : float maxSpeed : int distanceAccumulator : float vDistance : float TempColor : int period : int vMaxRevsSpirited : int vMaxRevsECO : int vRevsECOMargin : int vRevsLimit : int revsCounter : float revsHighCounter : float revsMidCounter : float revsLowCounter : float revsHighAVG : float

Figura 31 - Clase MainActivity

La clase *MainActivity* pertenece al subsistema de aplicaciones y hereda de la clase *Activity* de Android (con la cual se definen las pantallas de interfaz de usuario).

Esta es la clase principal del sistema, donde se mantiene toda la funcionalidad de monitorización y la que da acceso al resto de funciones de la aplicación. Esta clase estaba proporcionada por la aplicación que ha servido de base, si bien se ha modificado en profundidad para conseguir ofrecer los requerimientos actuales. Es una clase mucho más extensa de lo habitual, pues la funcionalidad soportada es enorme. No solo obtiene y

visualiza los parámetros de conducción extraídos del OBD-II, de los sensores del dispositivo y los calculados, sino que además, invoca a los métodos de los subsistemas de conexión, OBD y base de datos, orquestando la gestión de éstos.

Esta clase tiene muchos atributos, que para su estudio, lo dividiremos por los siguientes tipos:

- Atributos de manejadores, listeners, sensores y componentes visuales :
  - **mHandler**: Handler
  - **mListener**: IPostListener
  - **mServiceIntent**: Intent
  - **mServiceConnection**: ObdGatewayServiceConnection
  - **sensorManager**: SensorManager
  - **orientSensor**: Sensor
  - **locationListener**: LocationListener
  - **locationManager**: LocationManager
  - **locater**: MyLocater
  - **powerManager**: PowerManager
  - **wakeLock**: WakeLock
  - **orientListener**: SensorEventListener
  - **mQueueCommands**: Runnable
  - **pbRevoluciones**: ProgressBar
  - **mChronometer**: Chronometer

- Atributos relacionados con base de datos y registro de logs:
  - **db**: SQLiteDatabase
  - **db_cerrada**: boolean
  - **logFlag**: boolean
  - **dbActiveFlag**: boolean
  - **vToastVisibility**: boolean

- Atributos que permiten identificar en cada instante, el viaje, secuencia y comando actual.
  - **idTravel**: int
  - **descTravel**: String
  - **idSequence**: int
  - **descSequence**: String
  - **idCommand**: int

- Atributos para almacenar los valores de los parámetros de la última secuencia calculados.
  - **speed** : int
  - **maf**: double
  - **ltft**: float
  - **equivRatio**: double
  - **rpm**: int
  - **speedAccumulator**: float

- **speed0Accumulator**: float
- **speedCounter**: float
- **speedAv**: float
- **percStopped**: float
- **maxSpeed**: int
- **distanceAccumulator**: float
- **vDistance**: float
- **TempColor**: int
- **period**: int
- **vMaxRevsSpirited**: int
- **vMaxRevsECO**: int
- **vRevsECOMargin**: int
- **vRevsLimit**: int
- **revsCounter**: float
- **revsHighCounter**: float
- **revsMidCounter**: float
- **revsLowCounter**: float
- **revsHighAVG**: float
- **revsMidAVG**: float
- **revsLowAVG**: float
- **vLocation**: Location
- **latitud**: Double
- **longitud**: Double

- Atributos auxiliares:
  - **prefs**: SharedPreferences
  - **preRequisites**: boolean
  - **elapsedMillis**: long
  - **provider**: String
  - **numberOfTravelsStats**: int

Tiene los siguientes métodos:

- **onCreate()**, es el método principal. Crea la pantalla y deja todo preparado para poder comenzar la monitorización en el momento que se indique (crea los listener, cronometro, sensor de brújula,...).
- **onDestroy()**, ejecuta las operaciones pertinentes para destruir el objeto.
- **onPause()**, ejecuta las acciones a realizar cuando la activity entra en el estado de pausa.
- **releaseWakeLockIfHeld()**, libera el bloqueo de pantalla activa.
- **onResume()**, ejecuta las acciones necesarias para recuperar la activity desde el estado de pausa.
- **onCreateDialog(): Dialog**, si durante la creación se detecta algún problema con el bluetooth, GPS, o sensor de orientación, este método construye el mensaje de error que se mostrará al usuario.

- **onCreateOptionsMenu(): boolean**, crea las opciones de menú contextual.
- **onPrepareOptionsMenu(): boolean**, prepara las opciones del menú contextual, para que puedan ser utilizadas.
- **onOptionsItemSelected(): boolean**, recoge la opción de menú contextual seleccionada y ejecuta la acción correspondiente.
- **updateTextView()**, actualiza el valor de la brújula (en formato texto).
- **mostrarVelocidad()**, muestra por pantalla la velocidad obtenida del OBD y todos los parámetros asociados a ella y los almacena en base de datos si el flag estuviera activo.
- **mostrarRevoluciones()**, muestra por pantalla las revoluciones por minuto del motor y todos los parámetros asociados a este parámetro y los almacena en base de datos si el flag estuviera activo.
- **mostrarCoolantTemp()**, muestra por pantalla la temperatura del liquido refrigerante del motor y todos los parámetros asociados a este parámetro y los almacena en base de datos si el flag estuviera activo.
- **setDebug()**, método para activar/desactivar el flag de depuración.
- **getDBbActiveFlag(): boolean**, devuelve el estado del flag de registro en base de datos, es decir, si el usuario ha indicado que desea guardar la información en base de datos.
- **setDBActiveFlag()**, método para activar/desactivar el flag de base de datos.
- **recuperarPrefs()**, método para recuperar las preferencias indicadas en la configuración, que puedan afectar a esta activity.
- **updateConfig()**, se invoca a la pantalla de configuración.
- **helpView()**, se invoca a la pantalla de ayuda.
- **startLiveData()**, comienza la monitorización en tiempo real de la conducción.
- **bt_startLiveData()**, método onClick del botón START, que invoca al método *startLiveData()*.
- **stopLiveData()**, detiene la monitorización on-line de la conducción.
- **bt_stopLiveData()**, método onClick del botón STOP, que invoca al método *stopLiveData()*.
- **onClickDownload()**, método para la exportación de la base de datos a ficheros csv.
- **onClickStats()**, se invoca a la pantalla de selección de estadísticas.
- **addTableRow()**, incluye el valor del parámetro a cuadro de visualización de parámetros auxiliar.
- **queueCommands()**,es un método invocado desde un thread secundario, para iniciar la consulta de los parámetros al OBD de una secuencia. Se ejecuta con la periodicidad definida, desde el comienzo hasta el fin de la monitorización.

- **abrirDB()**, se encarga de abrir la base de datos, para su utilización por el programa.
- **initLocalizacion()**, inicia el proceso de posicionamiento GPS.
- **mostrarPosicion()**, calcula la distancia recorrida, la muestra por pantalla y almacena el posicionamiento en base de datos si el flag estuviera activo.

4.2.2.11. Clase SelStatsActivity

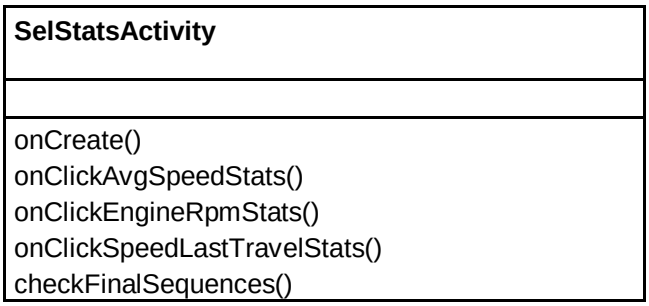


Figura 32 - Clase SelStatsActivity

La clase *SelStatsActivity* pertenece al subsistema de aplicaciones y hereda de la clase *Activity* de Android (con la cual se definen las pantallas de interfaz de usuario).

Esta clase proporciona la pantalla de selección de la estadística que se desea visualizar.

Esta clase no tiene atributos.

Tiene los siguientes métodos:

- **onCreate()**, crea la pantalla de selección y llama a *checkFinalSequences()* para dejar los datos preparados y facilitar la posterior generación y visualización de las estadísticas.
- **checkFinalSequences()**, comprueba si los últimos viajes tienen secuencia final y si no se marca la última secuencia como la final (si el usuario ha salido de la aplicación sin parar la monitorización)
- **onClickAvgSpeedStats()**, se accede a la ventana de estadísticas de velocidades medias de los últimos viajes.
- **onClickEngineRpmStats()**, se accede a la ventana de estadísticas con los porcentajes de exigencia del motor, medidos por rango de revoluciones por minuto, de los últimos viajes.
- **onClickSpeedLastTravelStats()**, generación de una gráfica de ejemplo.

4.2.2.12. Clase SpeedGraphActivity

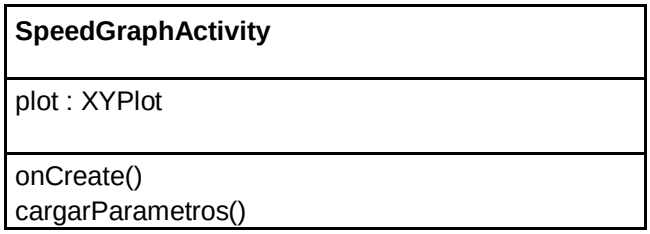


Figura 33 - Clase SpeedGraphActivity

La clase *SpeedGraphActivity* pertenece al subsistema de aplicaciones y hereda de la clase *Activity* de Android (con la cual se definen las pantallas de interfaz de usuario).

Esta clase proporciona la pantalla de visualización de la gráfica estadística de velocidades medias de los últimos viajes.

Esta clase tiene los siguientes atributos:

- **plot: XYPlot**, objeto del subsistema de estadísticas que facilita la visualización de la gráfica.

Tiene los siguientes métodos:

- **onCreate()**, crea la pantalla e invoca al subsistema de estadísticas para generar la gráfica.
- **cargarParametros()**, método que recupera los valores necesarios para la visualización de la gráfica.

4.2.2.13. Clase EngineRPMGraphActivity

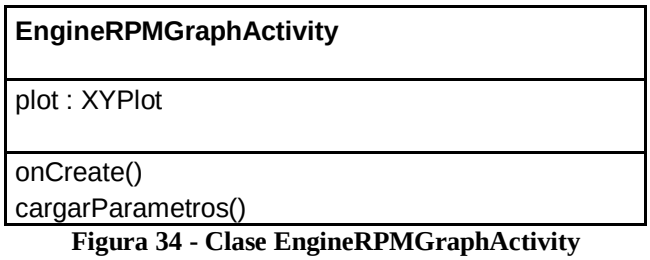


Figura 34 - Clase EngineRPMGraphActivity

La clase *EngineRPMGraphActivity* pertenece al subsistema de aplicaciones y hereda de la clase *Activity* de Android (con la cual se definen las pantallas de interfaz de usuario).

Esta clase proporciona la pantalla de visualización de la gráfica estadística de velocidades medias de los últimos viajes.

Esta clase tiene los siguientes atributos:

- **plot: XYPlot**, objeto del subsistema de estadísticas que facilita la visualización de la gráfica.

Tiene los siguientes métodos:

- **onCreate()**, crea la pantalla e invoca al subsistema de estadísticas para generar la gráfica.
- **cargarParametros()**, método que recupera los valores necesarios para la visualización de la gráfica.

4.3 DISEÑO DE BASE DE DATOS

4.3.1. Introducción

Un punto importante y diferenciador con respecto a las alternativas actuales que existen al presente proyecto es que el sistema consta de almacenamiento de la información obtenida, para permitir su posterior análisis. El resto de aplicaciones similares no guardan la información obtenida, con lo que la información es volátil y solo aprovechable en un momento dado.

Es muy conveniente puntualizar que debido al tipo de dispositivos a los que va dirigido, normalmente con una capacidad de procesamiento y sobre todo de almacenamiento muy limitadas, la base de datos debe ser lo más ligera y sencilla posible, siempre que ofrezca la funcionalidad buscada.

El sistema gestor de base de datos utilizado será SQLite3, por dos razones importantes:

- la primera es que el sistema gestor de base de datos es muy compacto y eficiente, ya que se integra en el propio programa, lo cual evita tener un proceso independiente y así se reducen los tiempos de latencia en el acceso a datos. El acceso se realizará desde el mismo dispositivo y la base de datos estará dedicada al programa en exclusiva, por lo que tampoco es necesario un complejo gestor que facilite la compartición de ésta.
- la segunda de ellas es que Android ya incluye SQLite, por lo que se asegura una integración perfecta y un comportamiento más que probado.

Esta simpleza, que tan bien se ajusta a nuestras necesidades, no implica que la base de datos pierda alguna de sus características esenciales, ya que SQLite es una base de datos relacional y compatible con ACID (**A**tomicity, **C**onsistency, **I**solation and **D**urability), es decir, que sus transacciones cumplen con las características de Atomicidad, Consistencia, Aislamiento y Durabilidad:

- **Atomicidad**: es la propiedad que asegura que la operación se ha realizado o no y por lo tanto ante un fallo del sistema no puede quedar a medias.
- **Consistencia**: Integridad. Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos. La propiedad de consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido.
- **Aislamiento**: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.

- **Durabilidad**: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

Una vez puestos en situación con esta pequeña introducción, se pasa a detallar paso por paso el diseño de la base de datos del proyecto.

4.3.2. Diseño Conceptual de la Base de Datos

4.3.2.1. Modelo Entidad-Relación

El modelo entidad-relación (E-R "Entity relationship") es la herramienta más utilizada para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

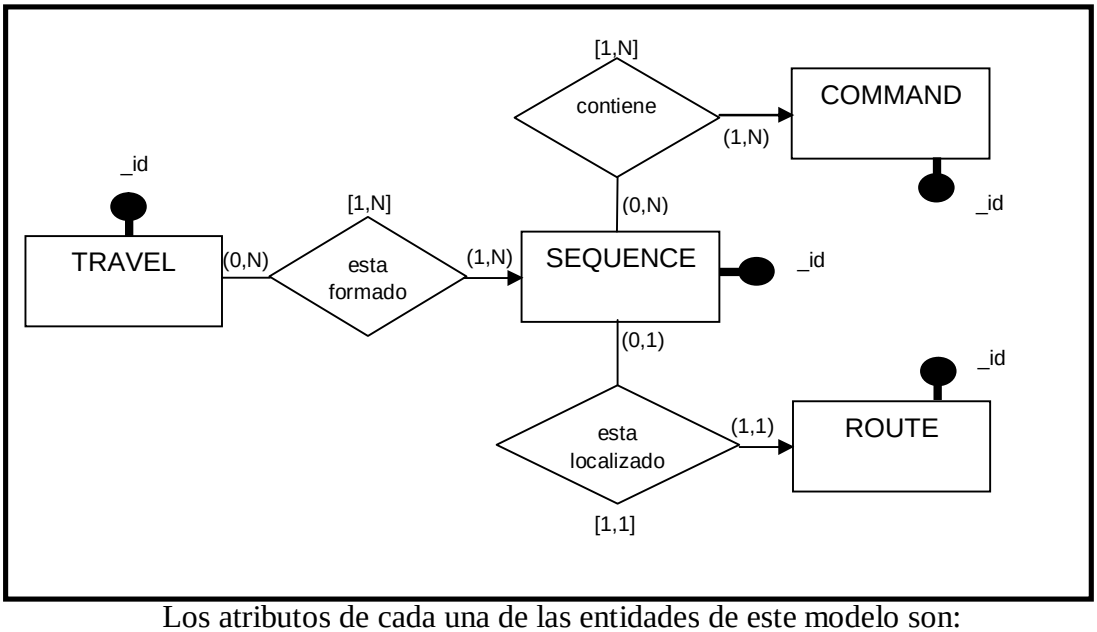
Éste es un concepto de modelado para bases de datos, propuesto por Peter Chen en 1976, mediante el cual se pretende 'visualizar' los objetos que pertenecen a la Base de Datos como entidades (se corresponde al concepto de objeto de la Programación Orientada a Objetos) las cuales tienen unos atributos y se vinculan mediante relaciones.

Brevemente consiste en los siguientes pasos:

1. Se parte de una descripción textual del problema o sistema de información a automatizar (los requisitos).
2. Se hace una lista de los sustantivos y verbos que aparecen.
3. Los sustantivos son posibles entidades o atributos.
4. Los verbos son posibles relaciones.
5. Analizando las frases se determina la cardinalidad de las relaciones y otros detalles.
6. Se elabora el diagrama (o diagramas) entidad-relación.
7. Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se pueden reflejar en el diagrama.

Para conseguir unos buenos resultados, esta técnica requiere de cierta experiencia.

A continuación se muestra el Esquema Entidad/Relación de la aplicación, que para una mejor comprensión no se han incluido los campos de las tablas.



Los atributos de cada una de las entidades de este modelo son:

Entidad TRAVEL:
- _id: Clave Principal
- description
- time_from
- time_to

Entidad SEQUENCE:
- _id: Clave Principal
- id_travel
- description
- timestamp

Entidad COMMAND:
- _id: Clave Principal
- id_travel
- id_sequence
- commandType
- formattedValue
- value

Entidad ROUTE:
- _id: Clave Principal
- id_travel
- id_sequence
- latitude
- longitude

Además se proporciona a continuación el diagrama del Modelo Entidad-Relación extendido:

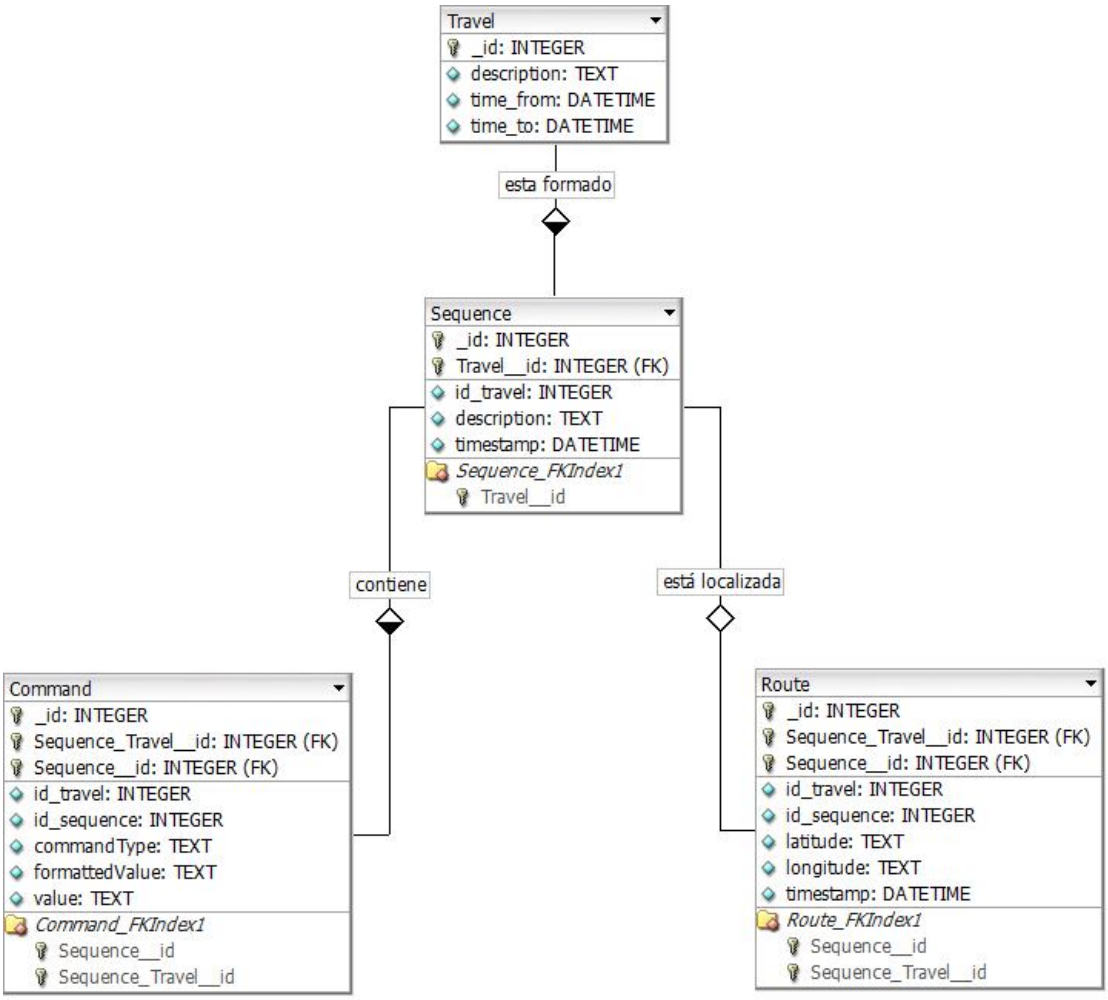


Figura 36 - Diagrama Entidad/Relación Extendido

4.3.3. Diseño Lógico de la Base de Datos

4.3.3.1. Modelo Relacional

En este apartado veremos la transformación del modelo Entidad-Relación al modelo Relacional.

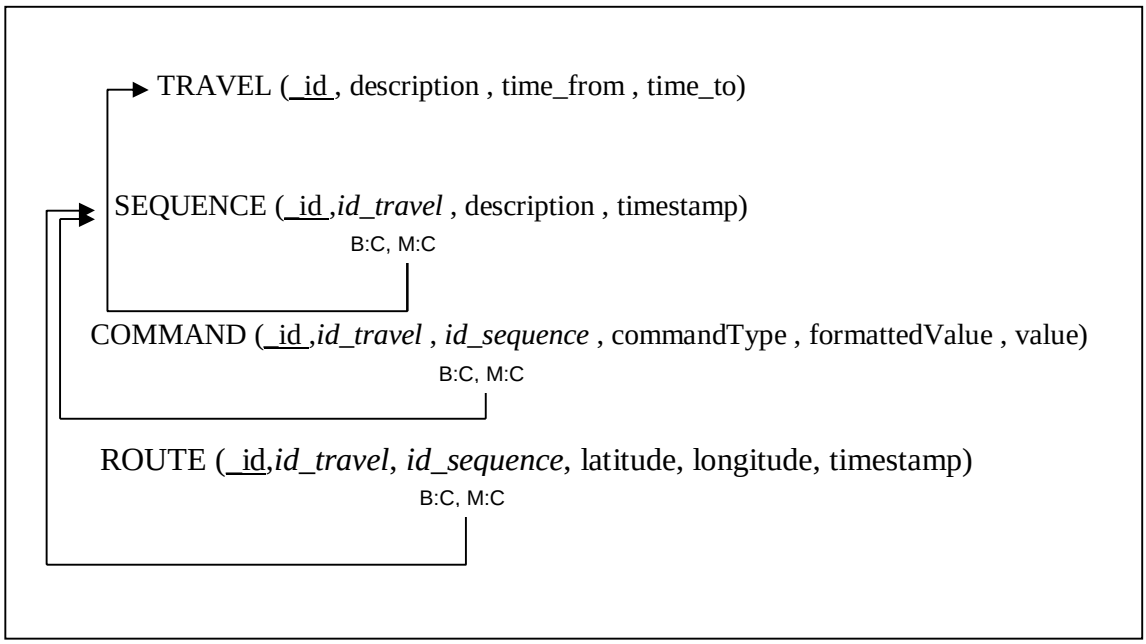


Figura 37 - Diagrama paso de Entidad/Relación a Relacional

4.3.4. Descripción de la base de datos

Tipo	**Base de datos**		
Versión	1.0 (05/07/2013)		
Autores	Ángel Martín Velázquez (Universidad de Valladolid)		
Requisitos Asociados	IRQ–01: Gestión de Viaje IRQ–02: Gestión de Secuencia IRQ–03: Gestión de Comando IRQ–04: Gestión de Ruta UC–02: Almacenamiento de la información UC–03: Exportación de la información UC–04: Visualización de estadísticas		
Descripción	Esta base de datos es utilizada para el uso de la aplicación Android OBD-II		
Tablas	**Nombre**	**Tipo OCL**	**Mult.**
	TRAVEL	Tabla	Indicado en ER
	SEQUENCE	Tabla	Indicado en ER
	COMMAND	Tabla	Indicado en ER
	ROUTE	Tabla	Indicado en ER
Comentarios	N/A		

Tabla 23 - Descripción de Base de datos

4.3.5. Tablas de la base de datos

4.3.5.1. Tabla TRAVEL

4.3.5.1.1. Descripción de la tabla TRAVEL:

Tipo	**Tabla**
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Requisitos Asociados	IRQ–01: Gestión de Viaje UC–02: Almacenamiento de la información UC–03: Exportación de la información UC–04: Visualización de estadísticas
Descripción	Esta es la tabla TRAVEL, destinada a almacenar la información del viaje.
Comentarios	N/A

Tabla 24 - Descripción de tabla TRAVEL

4.3.5.1.2. Campos de la tabla TRAVEL:

Campo	**Tipo**	**Descripción**	**Valor por Def**	**Clave Primaria (PK)**	**Clave Ajena (FK)**	**Restric**
_id	INTEGER	Identificador de viaje		S	N	No nulo
description	TEXT	Campo de texto donde almacenar la descripción de la secuencia		N	N	
time_from	DATETIME	Fecha/hora de comienzo del viaje		N	N	
time_to	DATETIME	Fecha/hora de fin del viaje		N	N	

Tabla 25 - Campos de tabla TRAVEL

4.3.5.1.3. Claves de la tabla TRAVEL:

Tipo	**Clave Primaria**
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Tablas Asociadas	- TRAVEL
Descripción	Esta clave identifica unívocamente una tupla de la tabla TRAVEL
Expresión SQL	*PRIMARY KEY(_id)*
Comentarios	N/A

Tabla 26 - Clave primaria de tabla TRAVEL

4.3.5.2. Tabla SEQUENCE

4.3.5.2.1. Descripción de la tabla SEQUENCE:

Tipo	**Tabla**
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Requisitos Asociados	IRQ–02: Gestión de Secuencia UC–02: Almacenamiento de la información UC–03: Exportación de la información UC–04: Visualización de estadísticas
Descripción	La tabla SEQUENCE se encarga de agrupar todos los parámetros y posicionamiento GPS, recuperados en el mismo momento.
Comentarios	N/A

Tabla 27 - Descripción de tabla SEQUENCE

4.3.5.2.2. Campos de la tabla SEQUENCE:

Campo	**Tipo**	**Descripción**	**Valor por Def**	**Clave Primaria (PK)**	**Clave Ajena (FK)**	**Restric**
_id	INTEGER	Identificador de secuencia		S	N	No nulo
id_travel	INTEGER	Identificador de viaje		N	S	No nulo
description	TEXT	Campo de texto donde almacenar la descripción de la secuencia		N	N	
timestamp	DATETIME	Timestamp		N	N	

Tabla 28 - Campos de tabla SEQUENCE

4.3.5.2.3. Claves de la tabla SEQUENCE:

Tipo	**Clave Primaria**
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Tablas Asociadas	- ROUTE
Descripción	Esta clave identifica unívocamente una tupla de la tabla SEQUENCE
Expresión SQL	*PRIMARY KEY(_id)*
Comentarios	N/A

Tabla 29 - Clave primaria de tabla SEQUENCE

Tipo	**Clave Foránea**
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Tablas Asociadas	- TRAVEL - SEQUENCE
Descripción	Esta clave es ajena de la tabla TRAVEL (_id)
Expresión SQL	*FOREIGN KEY(id_travel) REFERENCES Travel(_id)*
Comentarios	N/A

Tabla 30 - Clave foránea de tabla SEQUENCE

4.3.5.3. Tabla COMMAND

4.3.5.3.1. Descripción de la tabla COMMAND:

Tipo	Tabla
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Requisitos Asociados	IRQ–03: Gestión de Comando UC–02: Almacenamiento de la información UC–03: Exportación de la información UC–04: Visualización de estadísticas
Descripción	La tabla COMMAND, es la encargada de guardar los valores de los parámetros de conducción obtenidos.
Comentarios	N/A

Tabla 31 - Descripción de tabla COMMAND

4.3.5.3.2. Campos de la tabla COMMAND:

Campo	Tipo	Descripción	Valor por Def	Clave Primaria (PK)	Clave Ajena (FK)	Restric
_id	INTEGER	Identificador de comando		S	N	No nulo
id_travel	INTEGER	Identificador de viaje		N	S	No nulo
id_sequence	INTEGER	Identificador de secuencia		N	S	No nulo
commandType	TEXT	Tipo de comando		N	N	
formattedValue	TEXT	Valor del parámetro obtenido y formateado para mostrar		N	N	
value	TEXT	Valor del parámetro obtenido		N	N	

Tabla 32 - Campos de tabla COMMAND

4.3.5.3.3. Claves de la tabla COMMAND:

Tipo	Clave Primaria
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Tablas Asociadas	- COMMAND
Descripción	Esta clave identifica unívocamente una tupla de la tabla COMMAND
Expresión SQL	PRIMARY KEY(_id)
Comentarios	N/A

Tabla 33 - Clave primaria de tabla COMMAND

Tipo	Clave Foránea
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Tablas Asociadas	- COMMAND - SEQUENCE
Descripción	Esta clave es ajena de la tabla SEQUENCE (_id, id_travel)
Expresión SQL	FOREIGN KEY(id_sequence,id_travel) REFERENCES Sequence(_id, id_travel)
Comentarios	N/A

Tabla 34 - Clave foránea de tabla COMMAND

4.3.5.4. Tabla ROUTE

4.3.5.4.1. Descripción de la tabla ROUTE:

Tipo	Tabla
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Requisitos Asociados	IRQ–04: Gestión de Ruta UC–02: Almacenamiento de la información UC–03: Exportación de la información
Descripción	La tabla ROUTE, es la encargada de guardar la información de posición GPS, de cada secuencia.
Comentarios	N/A

Tabla 35 - Descripción de tabla ROUTE

4.3.5.4.2. Campos de la tabla ROUTE:

Campo	Tipo	Descripción	Valor por Def	Clave Primaria (PK)	Clave Ajena (FK)	Restric
_id	INTEGER	Identificador de punto de ruta		S	N	No nulo
id_travel	INTEGER	Identificador de viaje		N	S	No nulo
id_sequence	INTEGER	Identificador de secuencia		N	S	No nulo
latitude	TEXT	Almacena la latitud del posicionamiento GPS		N	N	
longitude	TEXT	Almacena la longitud del posicionamiento GPS		N	N	
timestamp	DATETIME	Timestamp		N	N	

Tabla 36 - Campos de tabla ROUTE

4.3.5.4.3. Claves de la tabla ROUTE:

Tipo	Clave Primaria
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Tablas Asociadas	- ROUTE
Descripción	Esta clave identifica unívocamente una tupla de la tabla ROUTE
Expresión SQL	PRIMARY KEY(_id)
Comentarios	N/A

Tabla 37 - Clave primaria de tabla ROUTE

Tipo	Clave Foránea
Versión	1.0 (05/07/2013)
Autores	Ángel Martín Velázquez (Universidad de Valladolid)
Tablas Asociadas	- ROUTE - SEQUENCE
Descripción	Esta clave es ajena de la tabla SEQUENCE (_id, id_travel)
Expresión SQL	FOREIGN KEY(id_sequence,id_travel) REFERENCES Sequence(_id, id_travel)
Comentarios	N/A

Tabla 38 - Clave foránea de tabla ROUTE

4.4 DISEÑO DE INTERFACES DEL USUARIO

4.4.1. Introducción

Una ventaja de Android, respecto al diseño de los interfaces de Android, es que dispone de una herramienta de construcción de las pantallas integrado en su herramienta de desarrollo (Android SDK, que se integra en Eclipse), que permite la creación y visualización de prototipos de pantalla de forma rápida y sencilla.

En Android las pantallas se definen en archivos XML, donde se van incluyendo cada uno de los componentes del interfaz y sus características.

Siempre hay que tener en mente, que las aplicaciones Android van destinadas a un tipo de dispositivos que disponen de pantallas de reducido tamaño con las restricciones que ello supone. Estas restricciones afectan a la cantidad de información mostrada, al tipo de componentes visuales que componen la ventana y al manejo de éstos (habitualmente táctil) que tiene una precisión mucho menor que los habituales punteros de un ordenador.

Teniendo todo lo anterior en cuenta y debido al limitado número de horas del trabajo, se ha optado por unas interfaces sencillas y muy claras. La claridad es una premisa básica por el uso que se va a dar a la aplicación, porque está involucrada en la conducción y debe aportar información, sin distraer ni molestar al conductor. Para futuras evoluciones se debe tener presente que este apartado es una de las tareas más importantes a realizar para conseguir un interfaz igual de clara, pero mucho más visual y atractiva.

A continuación se exponen los prototipos de las principales pantallas del sistema.

4.4.2. Pantalla de acceso al sistema

Es una pequeña presentación de la aplicación, que contiene el nombre de la aplicación, una imagen y diversos botones que darán el acceso a las diferentes pantallas del sistema.

Figura 38 - Pantalla de acceso

4.4.3. Pantalla de monitorización

En esta pantalla se muestran los principales parámetros de conducción que el usuario puede necesitar en tiempo real. Además se incluyen los accesos a las funcionalidades de estadísticas y exportación de la base de datos.

Figura 39 - Pantalla de monitorización

4.4.4. Pantalla de selección de estadísticas

Se trata de una pantalla sencilla que dispone de botones para poder seleccionar la gráfica estadística seleccionada. Se pensó en crear esta pantalla en lugar de incluir accesos directos desde la monitorización, para facilitar la evolución de la aplicación, con la inclusión de un mayor número de estadísticas.

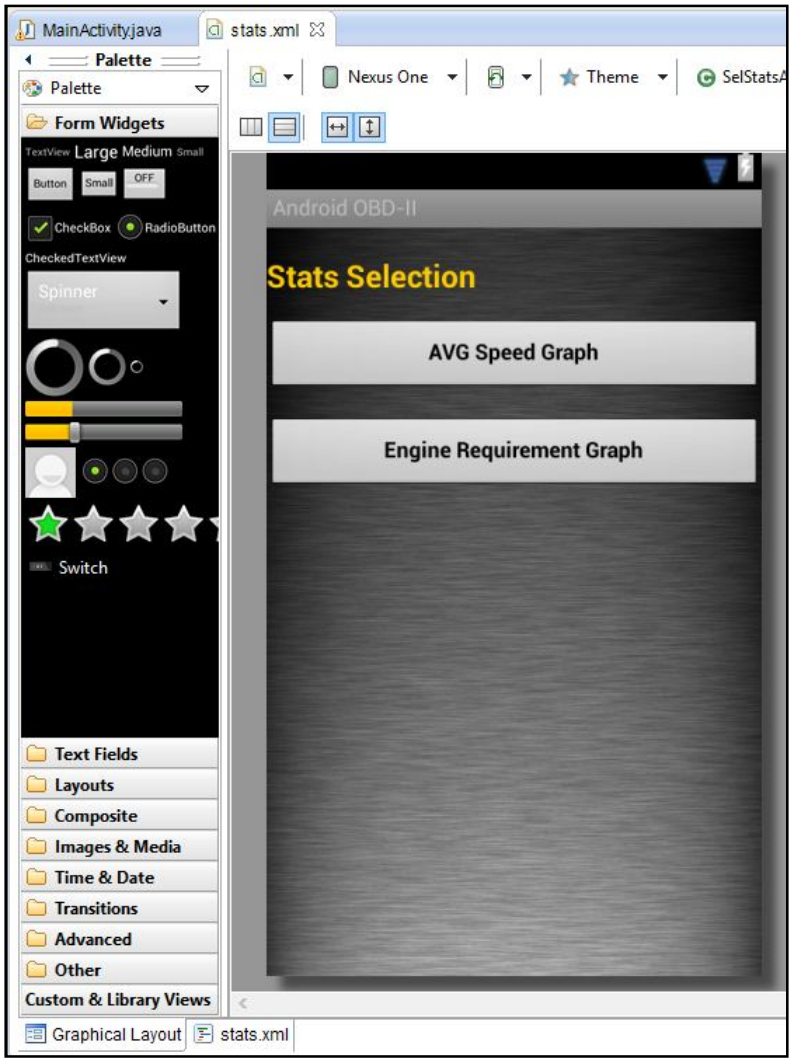


Figura 40 - Pantalla de selección de estadísticas

4.4.5. Otras

No se han incluido las pantallas de ayuda y de visualización de estadísticas. El motivo es que únicamente constan del componente visual que las muestra y no pueden ser plasmados hasta su implementación.

CAPÍTULO 5

PRUEBAS DEL SISTEMA

CAPÍTULO 5
PRUEBAS DEL SISTEMA

5.1 INTRODUCCIÓN

Las pruebas de software son los procesos que permiten identificar posibles fallos de implementación, calidad o usabilidad de un producto software. Todos los programas software tienen errores y esta fase es indispensable para localizarlos y poder corregirlos, pero como cita *E. W. Dijkstra - "El testing puede probar la presencia de errores pero no la ausencia de ellos"*. Con ello quería decirnos que es imposible asegurar la ausencia total de errores, ya siempre puede darse una situación no contemplada en las pruebas realizadas que pueda provocar un nuevo error.

Por tanto, el objetivo de esta fase es identificar el mayor número de errores posible, dentro de las limitaciones de tiempo y esfuerzo del proyecto. Para ello, sin duda, lo más conveniente es estructurar las pruebas y buscar los casos de pruebas más adecuados, para conseguir que esta fase sea lo más eficiente posible. La estructura de pruebas realizada es la siguiente:

- Pruebas Unitarias.
  - Caja Blanca: encargadas de probar función a función, cada uno de los módulos de la aplicación.
  - Caja Negra: encargadas de probar las funcionalidades, entradas y salidas de cada uno de los módulos de la aplicación.
- Pruebas de Integración.
  Son las encargadas de probar la coherencia e integración entre módulos, es decir, si la llamadas entre ellos funcionan correctamente y la información intercambiada entre ellos es la esperada.
- Pruebas de Validación.
  Son las encargadas de probar el cumplimiento de los requerimientos del sistema.
- Pruebas de Sistema.
  Son las encargadas de comprobar el comportamiento del sistema bajo situaciones donde es altamente ejercitado.

Centrándonos en las pruebas realizadas en este TFG, queríamos poner de manifiesto las dificultades añadidas con la que nos hemos encontrado para llevarlas a cabo, debido a la naturaleza de la aplicación. Éstas dificultades especiales han sido:

- En primer lugar, el canal de entrada al dispositivo es una comunicación bluetooth. Este canal no es soportado por el emulador de Android que se proporciona en su entorno de desarrollo, con lo que tenemos una primera dificultad añadida a subsanar.

- Además del punto anterior, la fuente de información es un conector OBD-II, que proporciona la información obtenida de un vehículo. Normalmente, los vehículos no son muy accesibles desde el lugar de trabajo.
- Ante estas dificultades, se pensó que una posible solución, sería realizar un sistema de simulación para inyectar datos de entrada, pero tiene dos importantes pegas: la primera es que el impacto en la planificación del proyecto no es asumible; y la segunda es que esta simulación no garantiza que la lectura de la fuente real se realice correctamente, incluso puede llegar a inyectar al sistema, errores propios del simulador. Por tanto, la imposibilidad de abordar este sistema de simulación es la siguiente dificultad con la que nos encontramos.

La opción por la que se optó finalmente fue desplazar toda la infraestructura de pruebas al vehículo, para poder *testear* toda la parte de monitorización (Conector OBD-II conectado al dispositivo Android y éste, al ordenador). Esta solución, si bien era la óptima para nuestro caso, complicó bastante la realización de las pruebas, que requirieron una dedicación y esfuerzo mayores de lo habitual para completar esta fase.

Destacar además, el tiempo invertido en detectar y corregir los errores del proyecto software del que se partía, pues su calidad y robustez no estaban aseguradas.

Podemos decir entonces que esta fase ha sido clave en el proyecto y ha sido la fase que ha provocado una mayor desviación en la planificación.

5.2 PRUEBAS UNITARIAS

Estas pruebas unitarias no se han realizado en una fase independiente, sino que se han integrado con la fase de implementación, para asegurar un correcto funcionamiento de cada una de las funcionalidades, en cuanto finalizaba su desarrollo. De esta forma, se disponía de un alcance de las pruebas más acotado, que permitía comprobar tanto el software desarrollado, como el del proyecto base usado para alcanzar la funcionalidad.

5.2.1. Pruebas de Caja Blanca

Estas pruebas de Caja Blanca se realizaron para cada uno de los métodos de las funcionalidades desarrolladas, revisando el código interno y utilizando métodos para su *Debug* cuando fueron necesarios, para asegurar su correcta ejecución.

Estas pruebas de Caja Blanca son necesarias para todo desarrollo software, pero no suficientes, ya que no permiten comprobar que el software hace lo que debe hacer, solo que lo que hace, lo hace bien.

5.2.2. Pruebas de Caja Negra

Las pruebas de Caja Negra, son realmente las pruebas funcionales del sistema, es decir, las que se encargan de comprobar que el sistema hace lo que se ha especificado. Estas pruebas complementan a las pruebas de Caja Blanca, y solo se centran en los resultados obtenidos de ese modulo, en función de la entrada, despreocupándose de lo que el modulo realice en su interior.

Las pruebas de Caja Negra también se ejecutaron al finalizar el desarrollo de cada una de las funcionalidades, gracias a que la planificación se pensó para que el desarrollo fuera incremental y poder disponer así de los datos de entrada necesarios para poder probar cada módulo, pues el anterior ya estaba desarrollado y probado.

5.3 PRUEBAS DE INTEGRACIÓN

Las pruebas de Integración permiten comprobar la comunicación entre los módulos y que el comportamiento general de la aplicación es coherente. Para estas pruebas de Integración se persigue completar escenarios de prueba *end-to-end* (de extremo a extremo) para comprobar que los resultados finales del sistema obtenidos a partir de unos datos de prueba concretos son los esperados.

Con escenario *end-to-end* no quiere decir que cada escenario de prueba tenga que diseñarse para pasar obligatoriamente por todos los módulos del sistema, pues no siempre es posible. Tan solo es necesario que pase por los módulos implicados en ese escenario, para obtener el resultado esperado. A cambio, lo ineludible, es contemplar todos los escenarios de prueba que permitan testear todas las comunicaciones posibles entre módulos.

Por supuesto, en nuestro proyecto, estas pruebas de Integración se han tenido que acometer una vez finalizado el desarrollo de todos los módulos del sistema. Se ha comprobado de esta manera que los datos devueltos de un módulo correspondían con la entrada adecuada del siguiente. Así, podemos destacar como escenario *end-to-end* principal, que los datos obtenidos del Conector OBD-II eran visualizados correctamente por pantalla y a su vez, eran almacenados en base de datos con el formato esperado, para recuperarlos y generar las estadísticas según lo requerido.

Este flujo y los correspondientes a las distintas fuentes del sistema, como sensores del dispositivo y la información calculada, fueron puestos a prueba, obteniendo un resultado correcto.

5.4 PRUEBAS DE VALIDACIÓN

Las pruebas de Validación se encargan de verificar que los requisitos del sistema están cubiertos por el software desarrollado. Por ello, se han diseñado todos los casos de prueba necesarios, para asegurarlo. Estos casos de prueba son los siguientes:

Caso de Prueba	Prueba	Resultado Esperado	Requisitos Asociados	Resultado Prueba
PR01	Monitorización en tiempo real de revoluciones (OBD-II).	Las revoluciones se muestran correctamente en la conducción tanto en formato textual, como en formato gráfico (barra de revoluciones). Además coinciden con las revoluciones indicadas en la instrumentación del vehículo	UC–01	Correcto
PR02	Monitorización en tiempo real de velocidad instantánea (OBD-II).	La velocidad instantánea se muestra correctamente durante la conducción y ésta coincide con la indicada en la instrumentación del vehículo	UC–01	Correcto
PR03	Visualización de estado del tanque de combustible (OBD-II).	El estado del tanque de combustible se muestra correctamente durante la conducción y ésta coincide con la indicada en la instrumentación del vehículo	UC–01	Correcto
PR04	Monitorización de temperatura del refrigerante (OBD-II)	La temperatura del refrigerante se muestra correctamente durante la conducción y ésta coincide con la indicada en la instrumentación del vehículo.	UC–01	Correcto
PR05	Monitorización de parámetros auxiliares extraídos del OBD-II	Los parámetros "Mass Air Flow" y "Engine Load" son obtenidos. No se muestran por su escasa relevancia para el conductor durante la conducción, pero si se almacenan para posibilitar su posterior análisis.	UC–01	Correcto
PR06	Monitorización de indicador de cambio de marchas (calculado).	El indicador de cambio de marchas reacciona en función del los rangos establecidos en la configuración.	UC–01	Correcto
PR07	Monitorización de brújula (sensores del dispositivo)	La brújula reacciona correctamente en función de la orientación.	UC–01	Correcto
PR08	Monitorización de velocidad media (calculado)	Se realiza el cálculo correcto durante la conducción de la velocidad media del viaje, y se muestra por pantalla.	UC–01	Correcto
PR09	Monitorización de porcentaje de tiempo detenido (calculado)	Se realiza el cálculo correcto durante la conducción del porcentaje del tiempo que el vehículo se encuentra detenido a lo largo del viaje y se muestra por pantalla.	UC–01	Correcto
PR10	Monitorización de cronómetro del viaje (dispositivo)	Se visualiza correctamente el tiempo de viaje.	UC–01	Correcto
PR11	Monitorización de distancia recorrida (sensores del dispositivo)	Se obtiene la distancia recorrida, a partir del posicionamiento GPS.	UC–01	Correcto
PR12	Almacenamiento de información monitorizada en base de datos	Si se activa el registro en base de datos, toda la información anterior es almacenada	UC–01 UC–02	Correcto
PR13	Exportación de tabla de base de datos "TRAVEL" (viaje)	Se obtiene la información de la tabla TRAVEL de forma completa y correcta, en un fichero con formato csv.	UC–03	Correcto

PR14	Exportación de tabla de base de datos "SEQUENCE" (secuencia)	Se obtiene la información de la tabla SEQUENCE de forma completa y correcta, en un fichero con formato csv.	UC–03	Correcto
PR15	Exportación de tabla de base de datos "COMMAND" (parámetros)	Se obtiene la información de la tabla COMMAND de forma completa y correcta, en un fichero con formato csv.	UC–03	Correcto
PR16	Exportación de tabla de base de datos "ROUTE" (puntos de ruta)	Se obtiene la información de la tabla ROUTE de forma completa y correcta, en un fichero con formato csv.	UC–03	Correcto
PR17	Generación de la gráfica estadística de Velocidad Media de los últimos viajes.	Se visualiza correctamente la gráfica lineal, donde la línea que indica la velocidad media de cada unos de los últimos viajes.	UC–04	Correcto
PR18	Generación de la gráfica estadística de porcentaje de esfuerzo requerido al motor de los últimos viajes.	Se visualiza correctamente la gráfica lineal, que consta de 3 líneas, cada una de las cuales significa: - Roja: % de tiempo de exigencia máxima del motor. - Azul: % de tiempo de exigencia media del motor. - Verde: % de tiempo de exigencia baja del motor. (conducción eficiente) Estos valores se obtienen para los últimos viajes, de modo que se pueda comparar visualmente la eficiencia en la conducción entre ellos.	UC–04	Correcto
PR19	Modificar los distintos datos de configuración	Los nuevos datos de configuración se almacenan correctamente, y modifican el comportamiento de la monitorización según lo indicado.	UC–05	Correcto
PR20	Mostrar ayuda	Se accede a la opción de ayuda y esta se visualiza correctamente	UC–06	Correcto

Tabla 39 - Tabla de casos de prueba

Todos ellos han obtenido un resultado correcto, pero cabe señalar el caso de prueba "PR03 - Visualización de estado del tanque de combustible (OBD-II)", porque la visualización de este nos devuelve el valor 0. El resultado es correcto, ya que es el valor que proporciona el OBD-II, pero puede resultar extraño. Esto se debe a que no todos los vehículos proporcionan todos y cada uno de los parámetros que el protocolo es capaz de ofrecer y en este caso, el vehículo disponible para realizar las pruebas (OPEL ASTRA), no proporcionaba este parámetro.

5.5 PRUEBAS DEL SISTEMA

Las pruebas del Sistema son aquellas que nos van a permitir comprobar el comportamiento del sistema bajo exigencias especialmente altas.

Estas pruebas se realizan después de comprobar que el sistema tiene un funcionamiento correcto y cubre los requisitos para los que ha sido concebido. Con ellas se somete al sistema a situaciones extremas, para hallar el límite que el sistema puede

soportar y ofrecer una respuesta aceptable. Si los resultados obtenidos no son asumibles, el sistema debe ser optimizado; e incluso en ocasiones debe ampliarse el hardware para conseguir los objetivos.

En cuanto a las pruebas del sistema a las que nuestro proyecto ha sido sometido, han sido especialmente de rendimiento y carga, soportando viajes de larga duración (de al menos una hora y media) con la frecuencia máxima de monitorización (0,250 segundos), obteniendo unos resultados satisfactorios.

CAPÍTULO 6

CONCLUSIONES PERSONALES

CAPÍTULO 6
CONCLUSIONES PERSONALES

6.1 CONCLUSIONES PERSONALES

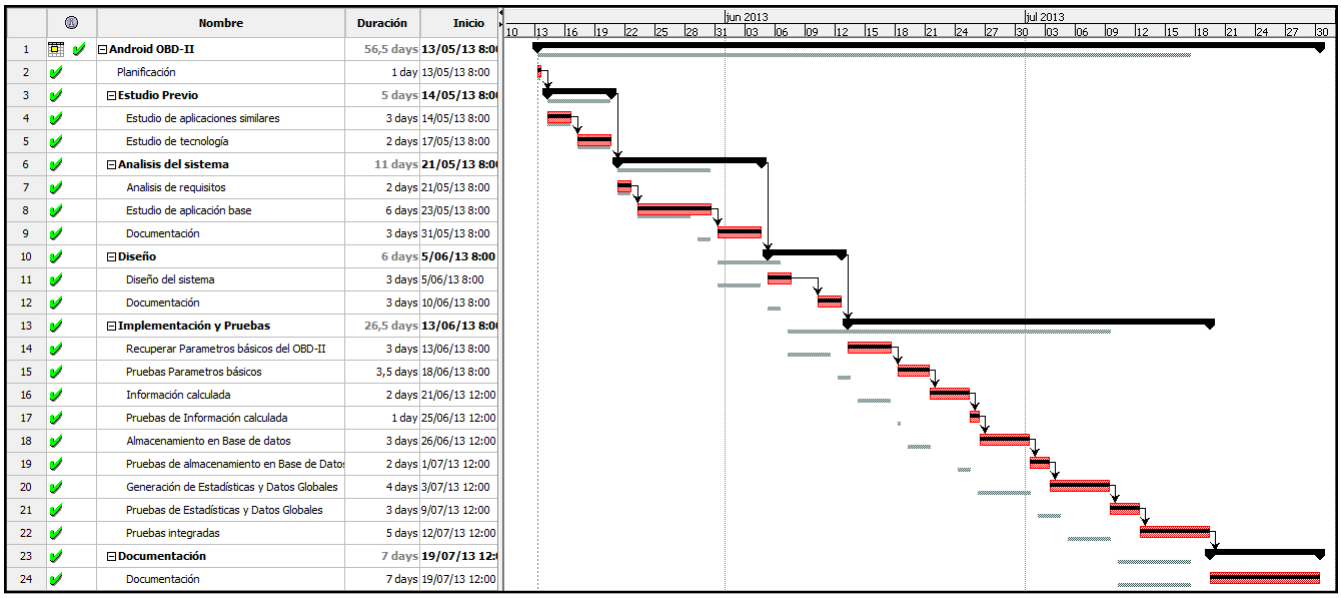
En este apartado hemos llegado a las conclusiones personales sobre el Trabajo Fin de Grado.

Desde mi punto de vista personal, este Trabajo Fin de Grado ha supuesto un gran reto a nivel de programación y funcional. Por un lado, tanto el lenguaje Java como el entorno de desarrollo de Android con todas sus librerías eran algo novedoso para mí. Funcionalmente, también ha sido desafiante descubrir desde cero la monitorización y diagnosis de vehículos y el protocolo OBD-II. La librería a utilizar para la recuperación de los parámetros OBD-II ha sido otro de los escalones a superar a la hora de finalizar el proyecto con éxito, porque ésta era totalmente desconocida y no se disponía de documentación alguna que facilitara su uso.

A cambio, es muy gratificante haber podido obtener tantos nuevos conocimientos sobre dos temas tan actuales como la programación de aplicaciones para dispositivos móviles y el campo del info-entretenimiento y monitorización del vehículo, que están en plena expansión. Día a día los grandes fabricantes realizan grandes inversiones por incluir sistemas complejos que faciliten al conductor funciones e información que hasta hace pocos años parecían de ciencia-ficción.

El esfuerzo para conseguir los ambiciosos objetivos marcados inicialmente ha sido muy grande y las dificultades no han sido pocas, pero llevar a cabo y finalizar este proyecto cumpliendo todos los objetivos y requerimientos marcados, me ha aportado una gran satisfacción.

Repasando toda la vida del proyecto, cabe reseñar las tareas más complicadas y que más esfuerzo han requerido, que han sido el estudio de la aplicación base, con multitud de nuevos conceptos técnicos y funcionales; y las pruebas, con las dificultades ya comentadas, pasando horas en el coche con el ordenador conectado al móvil, y éste al conector OBD para conseguir depurar la aplicación. La generación de la documentación también ha requerido algo más de tiempo del estimado inicialmente, desviando entre todas la planificación inicial:



	Nombre	Duración	Inicio
1	**Android OBD-II**	56,5 days	13/05/13 8:0
2	Planificación	1 day	13/05/13 8:00
3	**Estudio Previo**	5 days	14/05/13 8:0
4	Estudio de aplicaciones similares	3 days	14/05/13 8:00
5	Estudio de tecnología	2 days	17/05/13 8:00
6	**Analisis del sistema**	11 days	21/05/13 8:0
7	Analisis de requisitos	2 days	21/05/13 8:00
8	Estudio de aplicación base	6 days	23/05/13 8:00
9	Documentación	3 days	31/05/13 8:00
10	**Diseño**	6 days	5/06/13 8:00
11	Diseño del sistema	3 days	5/06/13 8:00
12	Documentación	3 days	10/06/13 8:00
13	**Implementación y Pruebas**	26,5 days	13/06/13 8:0
14	Recuperar Parametros básicos del OBD-II	3 days	13/06/13 8:00
15	Pruebas Parametros básicos	3,5 days	18/06/13 8:00
16	Información calculada	2 days	21/06/13 12:00
17	Pruebas de Información calculada	1 day	25/06/13 12:00
18	Almacenamiento en Base de datos	3 days	26/06/13 12:00
19	Pruebas de almacenamiento en Base de Datos	2 days	1/07/13 12:00
20	Generación de Estadísticas y Datos Globales	4 days	3/07/13 12:00
21	Pruebas de Estadísticas y Datos Globales	3 days	9/07/13 12:00
22	Pruebas integradas	5 days	12/07/13 12:00
23	**Documentación**	7 days	19/07/13 12:
24	Documentación	7 days	19/07/13 12:00

Figura 41 - Planificación final

Como se puede observar, respecto a la línea base existe una ligera desviación, pasando de una duración de 48 jornadas a 56,5. Si este cálculo lo hacemos en horas se ha pasado de un total de 384 estimadas inicialmente a 452 horas reales, lo que supone una incremento de menos de un 18%.

Teniendo en cuenta que es un proyecto con fines educativos y la falta de experiencia en las tecnologías utilizadas, la desviación de la planificación no es muy grande y parece asumible. La razón para poder afirmar esto, es que según los diferentes estudios realizados sobre proyectos reales y mi experiencia laboral (más de 12 años en proyectos software), los proyectos software suelen sufren una desviación media de la planificación de aproximadamente un 25%.

CAPÍTULO 7

FUTURAS LÍNEAS DE TRABAJO

CAPÍTULO 7
FUTURAS LÍNEAS DE TRABAJO

7.1 FUTURAS LÍNEAS DE TRABAJO

Una vez cubiertos los objetivos de este Trabajo Fin de Grado, en este punto trataremos de recoger algunas de las múltiples opciones de evolución que ofrece.

Los primeros aspectos que se me ocurren para futuras evoluciones serían dotarla de un interfaz más avanzado y una ampliación de la información recuperada del OBD-II, es decir, extender la librería con un mayor número de parámetros.

Además, las posibilidades que ofrece este software como proyecto de base son muy amplias: desde aplicaciones para el gran público, hasta sistemas más grandes para empresas, como por ejemplo de logística. Pasamos a ampliar estás ideas:

- Aplicaciones para el gran público:
  - La primera de la vías explotables destinada a los usuarios sería completar la aplicación con los dos puntos anteriores, con la funcionalidad añadida de poder subir la información a un servidor de internet. Éste servidor constaría de una plataforma abierta a los usuarios que recoja la información generada de cada uno de ellos. Esto permitiría establecer comparaciones de rendimiento entre los distintos modelos de vehículos y conocer el consumo real de los vehículos; además de poder comparar tu comportamiento al conducir con el de los usuarios más eficientes con la finalidad de mejorar la eficiencia de tu conducción, etc.

    Actualmente, existe una plataforma que permite llevar los gastos del coche y conocer el consumo real de cada modelo, que se llama "**Spritmonitor**" y es usada por miles de personas, pero la introducción de datos es manual y se limita principalmente al gasto de combustible. En nuestra aplicación se podrían ampliar las funcionalidades e información enormemente, con la ventaja de no tener que introducirlos manualmente, ya que éstos se subirían automáticamente desde el dispositivo.

  - Otra alternativa sería evolucionar el programa y orientarlo hacia un sistema de telemetría, que permitiría obtener un sinfín de información valiosa para aficionados a la competición del motor con recursos limitados. Por supuesto no ofrecería todas las ventajas de un sistema profesional, pero el registro de datos, combinando los obtenidos del vehículo y los aportados por el dispositivo (brújula, GPS, acelerómetro), permitirían el análisis de éstos datos a un coste irrisorio.

- Aplicaciones para empresas:
  - Mirando hacia sistemas empresariales más grandes, podría ser un punto de partida para empresas logísticas, de alquiler de coches o cualquiera que disponga de una flota de vehículos. Desde el dispositivo Android se podría

proporcionar la información a un sistema que la centralice y aporte las siguientes funciones:

- Rastreo de vehículos, para poder realizar un seguimiento de rutas óptimas, indicar la posición exacta de un envío o conocer la localización exacta si un coche se ha desviado o incluso lo han robado.
- Visualización y registro desde la central de las posibles incidencias, accidentes y averías que pueda haber sufrido cada vehículo.
- Monitorización on-line en remoto, que permitiría conocer desde el sistema central el estado de cada vehículo, si se está rebasando la velocidad permitida, si se está *maltratando* el vehículo, etc.
- Estudio de conducta del conductor. Esta funcionalidad permitiría a empresas analizar el tipo de conducción de cada conductor y poder ofrecerle formación en conducción eficiente, si fuera necesario.
- Aviso a emergencias en caso de accidente o avería, indicando la posición exacta del vehículo y permitiendo una asistencia más rápida.
- Optimización de intervalos de mantenimiento, pues conociendo como se ha exigido al vehículo, se pueden llegar a ampliar los plazos de mantenimiento.

Todo ello, reportaría a la empresa una mayor información y control de toda su flota, así como un ahorro en costes de mantenimiento, combustible, multas y tiempo.

CAPÍTULO 8

GLOSARIO

CAPÍTULO 8
GLOSARIO

8.1 GLOSARIO

En este apartado se expone brevemente la definición de los distintos conceptos que se han utilizado en la Memoria.

- **Android**: Android es un Sistema Operativo basado en Linux, diseñado principalmente para dispositivos móviles, propiedad de Google.
- **OBD-II**: *On-Board Diagnosis*. Es un sistema de diagnóstico a bordo en vehículos.
- **OBD-II PIDs**: *On-board diagnostics Parameter IDs* son códigos para consultar información de un vehículos, utilizados por una herramienta de diagnosis.
- **Bluetooth**: Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz
- **ABS**: *Antilock Brake System* o sistema antibloqueo de ruedas. Es un dispositivo utilizado en vehículos, que hace variar la fuerza de frenado para evitar que los neumáticos pierdan la adherencia con el suelo.
- **ESP**: Programa Electrónico de Estabilidad. El control de estabilidad es un elemento de seguridad activa del automóvil que actúa frenando individualmente las ruedas en situaciones de riesgo para evitar derrapes.
- **USB**: *Universal Serial Bus* o bus universal en serie es un estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores y periféricos y dispositivos electrónicos.
- **Smartphone**: Es un teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades semejantes a una mini computadora y conectividad que un teléfono móvil convencional.
- **Tablet**: Es una computadora portátil de mayor tamaño que un teléfono inteligente o una PDA, integrado en una pantalla táctil con la que se interactúa primariamente con los dedos o un estilete, sin necesidad de teclado físico ni ratón.
- **GPS**: *Global Positioning System* ó sistema de posicionamiento global, es un sistema global de navegación por satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo.
- **Ficheros CSV**: (*comma-separated values*) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma) y las filas por saltos de línea.
- **MATLAB**: (MATrix LABoratory, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

- **Herencia**: es uno de los mecanismos de los lenguajes de programación orientada a objetos basados en clases, por medio del cual una clase se deriva de otra de manera que extiende su funcionalidad.
- **Polimorfismo**: se refiere a la posibilidad de enviar un mensaje a un grupo de objetos cuya naturaleza puede ser heterogénea. El único requisito que deben cumplir los objetos que se utilizan de manera polimórfica es saber responder al mensaje que se les envía.
- **Licencia Apache 2**: es una licencia de software libre creada por la Apache Software Foundation (ASF). La licencia Apache requiere la conservación del aviso de copyright y el disclaimer, pero no es una licencia copyleft, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.
- **BBDD**: Base de datos.
- **Flag**: tipo de variable, normalmente booleana, que actúa de indicador.
- **Thread**: hilo de procesamiento paralelo al proceso principal.
- **Modelo E/R**: Modelo Entidad Relación.
- **B:C:** Borrado en Cascada. Siglas utilizadas en el Modelo Relacional para indicar esa opción de integridad referencial.
- **B:N** : Borrado con puesta a Nulos. Siglas utilizadas en el Modelo Relacional para indicar esa opción de integridad referencial.
- **B:R** : Borrado Restringido. Siglas utilizadas en el Modelo Relacional para indicar esa opción de integridad referencial.
- **M:C** : Modificación en Cascada. Siglas utilizadas en el Modelo Relacional para indicar esa opción de integridad referencial.
- **M:N** : Modificación con puesta a nulos. Siglas utilizadas en el Modelo Relacional para indicar esa opción de integridad referencial.
- **M:R**: Modificación Restringido. Siglas utilizadas en el Modelo Relacional para indicar esa opción de integridad referencial.
- **Testear**: Probar.
- **Debug**: Depuración.
- **Prueba end-to-end**: prueba en la que se analiza el sistema desde una de sus entradas hasta la salida correspondiente, poniendo a prueba cada uno de los módulos por los que pasa y las comunicaciones entre ellos.
- **Info-entretenimiento**: así es como se denomina comercialmente al conjunto de sistemas electrónicos de un vehículo que engloban las funciones de información y entretenimiento al conductor, tales como el ordenador de a bordo, radio, navegador, conexión bluetooth para conectar el teléfono móvil, etc.

CAPÍTULO 9

BIBLIOGRAFÍA

CAPÍTULO 9
BIBLIOGRAFÍA

9.1 BIBLIOGRAFÍA

Durante todo el ciclo de vida del proyecto, desde el estudio previo hasta el final de su construcción, han sido numerosas las fuentes de información consultadas. Principalmente estas fuentes han sido obtenidas de enlaces web, que permitían encontrar información de forma ágil y eficiente. A continuación indicamos dichos enlaces web, categorizados según el tipo de información ofrecida:

- **Conocimiento general sobre el OBD-II:**

  - Wikipedia (2013). *On-board diagnostics.* http://en.wikipedia.org/wiki/On-board_diagnostics.

  - Wikipedia (2013). *OBD-II PIDs (On-board diagnostics Parameter IDs).* http://en.wikipedia.org/wiki/OBD-II_PIDs.

  - OBD-II Resource (2010). *OBD-II PIDs.* http://obdcon.sourceforge.net/2010/06/obd-ii-pids/.

  - González Melis, P (2008). *On Board Diagnostic II.* http://tec.upc.es/eau/OBDII.pdf (Universidad Politécnica de Cataluña).

  - obd2crazy.com. *Interpreting OBD2 Data.* http://www.obd2crazy.com/techdata.html.

  - OBD-Codes.com. *OBD-II Trouble Codes.* http://www.obd-codes.com/.

  - OBD-Codes.com. *OBD2 Codes Explained.* http://www.obd-codes.com/faq/obd2-codes-explained.php.

  - OBD Experts. *OBD II Software.* http://www.obdexperts.co.uk/stack.html.

- **Programación y comunicación con el OBD-II:**

- code.google.com (2012).*OBD-II reader for Android devices.* http://code.google.com/p/android-obd-reader/

- code.google.com (2012). OBD-II PID's API 1.3 http://code.google.com/p/android-obd-reader/downloads/detail?name=api-1.3.jar&can=2&q=

- Androidplot (2013). *API for creating dynamic and static charts within your Android application.* http://androidplot.com/

- Developers Android. (2013). *Bluetooth.* http://developer.android.com/guide/topics/connectivity/bluetooth.html.

- Developers Android. (2013). *Location and Sensors APIs.* http://developer.android.com/guide/topics/sensors/index.html.

- Developers Android. (2013). *Storage Options.* http://developer.android.com/guide/topics/data/data-storage.html#netw.

- Salvador Gómez (2012). *Curso Programación Android.* http://www.sgoliver.net/blog/?page_id=3011

- stackoverflow.com. *Resolución de múltiples problemas encontrados* http://stackoverflow.com/questions/tagged/android

- en.opensuse.org (2013) *SDB:ELM327 based ODB2 scan tool*. http://en.opensuse.org/SDB:ELM327_based_ODB2_scan_tool

- Arduino Forum (2012).*My home-brew Arduino OBD-II connection kit* http://arduino.cc/forum/index.php?topic=95037.0

- Francesco Zanitti (2010). Android Bluetooth Simulator https://github.com/cheng81/Android-Bluetooth-Simulator

- Off on a Tangent(2011). *Bluetooth support on Android Emulator*. http://niro-offonatangent.blogspot.com.es/2011/06/bluetooth-support-on-android-emulator.html

- **Licencias Software:**

  - The Apache Software Foundation (2004). *Apache License, Version 2.0.* http://www.apache.org/licenses/LICENSE-2.0.html

  - Wikipedia. (2013). *Apache License.* https://es.wikipedia.org/wiki/Apache_License.

  - Cenatic. *Análisis de licencias de fuentes y contenidos abiertos.* http://wiki.cenatic.es/wikiesp/index.php/CAP%C3%8DTULO_TRES:_An%C3%A1lisis_de_licencias_de_fuentes_y_contenidos_abiertos.

  - Miriam Ruiz (2011). Licencias de Software Libre http://www.slideshare.net/inniyah/licencias-de-software-libre

CAPÍTULO 10

ÍNDICE DE TABLAS

CAPÍTULO 10
ÍNDICE DE TABLAS

10.1 ÍNDICE DE TABLAS

CAPÍTULO 11

ÍNDICE DE FIGURAS

CAPÍTULO 11
ÍNDICE DE FIGURAS

11.1 ÍNDICE DE FIGURAS

SECCIÓN II

MANUAL DE USUARIO

CAPÍTULO 1

REQUISITOS DEL SISTEMA

CAPÍTULO 1
REQUISITOS DEL SISTEMA

1.1 REQUISITOS HARDWARE Y SOFTWARE

Para la instalación y correcto funcionamiento de la aplicación, el hardware requerido es:

- Conector OBDII Bluetooth.
- Dispositivo Android.

  La aplicación podrá ser instalada en cualquier dispositivo Android, con una versión igual o superior a 2.3 también conocida como *Gingerbread*, que disponga de Bluetooth, GPS y sensor de orientación.

CAPÍTULO 2

INSTALACIÓN DEL SISTEMA

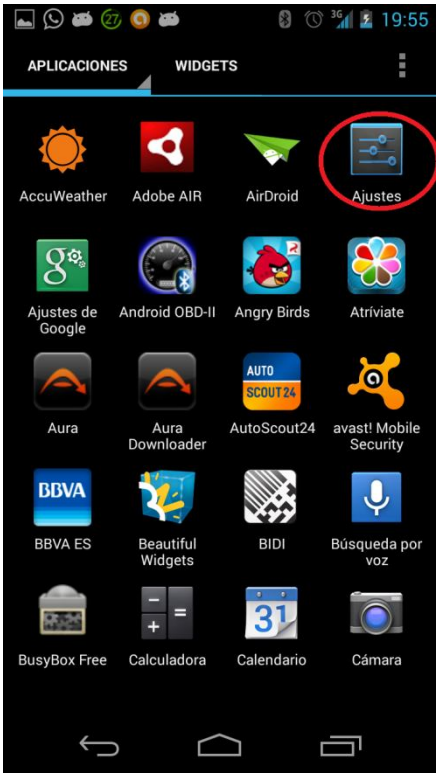
CAPÍTULO 2
INSTALACIÓN DEL SISTEMA

2.1 INSTALACIÓN

La instalación en Android es muy simple e intuitiva. Para instalarlo lo único que hay que hacer es mover el archivo ***AndroidODBII.apk*** incluido en el CD, al dispositivo móvil y ejecutar éste para que Android comience su instalación.

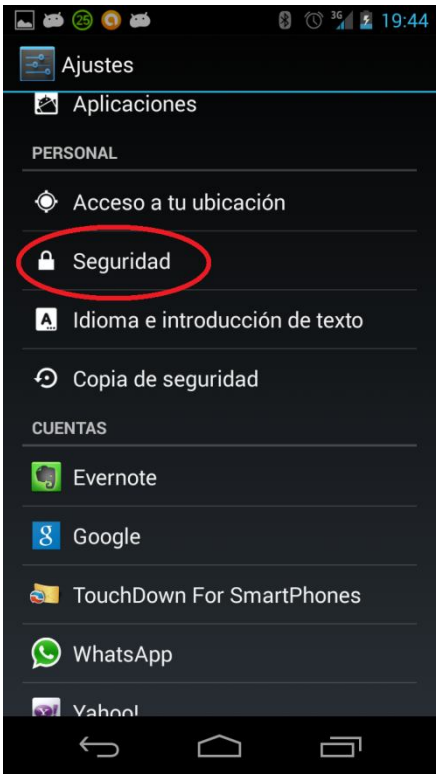
Vamos a repasar este sencillo proceso paso a paso, para resolver cualquier pequeña duda sobre la instalación:

- Paso 1: Permitir la instalación de aplicaciones desde un origen distinto a Google Play.

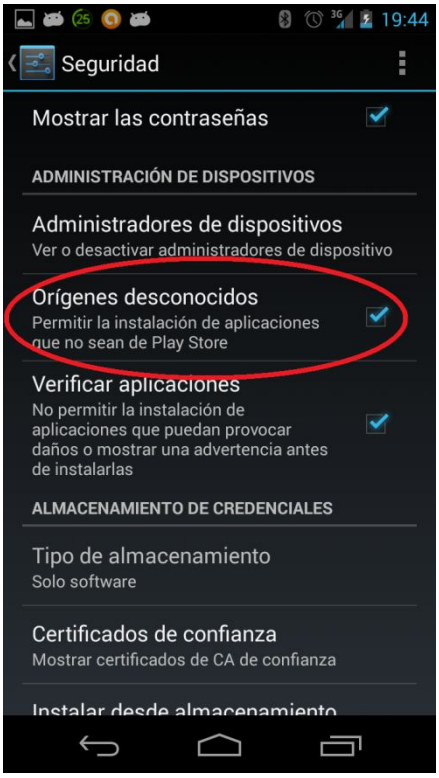


Para ello, lo primero es acceder a los *Ajustes* de Android.

Este acceso lo encontramos una vez que accedemos a las aplicaciones. En nuestro caso es el icono marcado con un círculo.



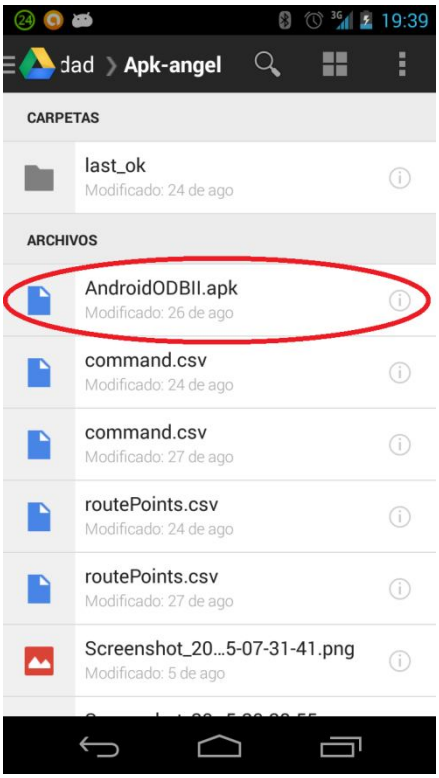
Una vez dentro de Ajustes, debemos seleccionar la opción de *Seguridad*.



Dentro de los ajustes de seguridad, hay que comprobar que la opción *Orígenes desconocidos* está habilitada.

En caso de no estar activa, por favor, actívela.

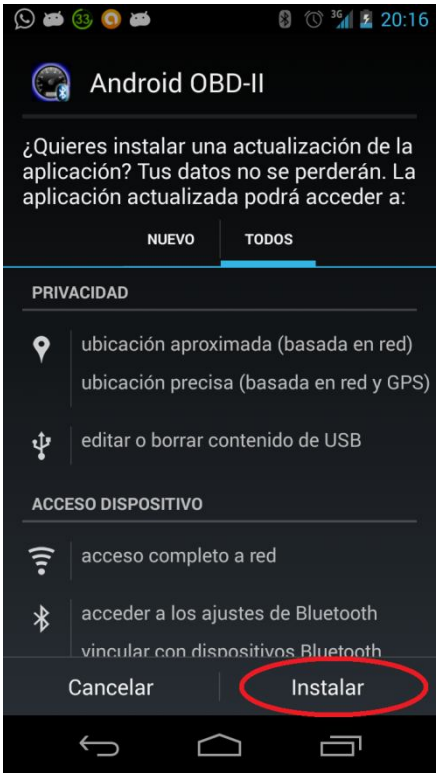
- Paso 2: Abrir el explorador de archivos para localizar el fichero.



Una vez guardado en el teléfono móvil, abrir el explorador de archivos para localizar el fichero (en el ejemplo se utilizando Google Drive para moverlo al móvil).

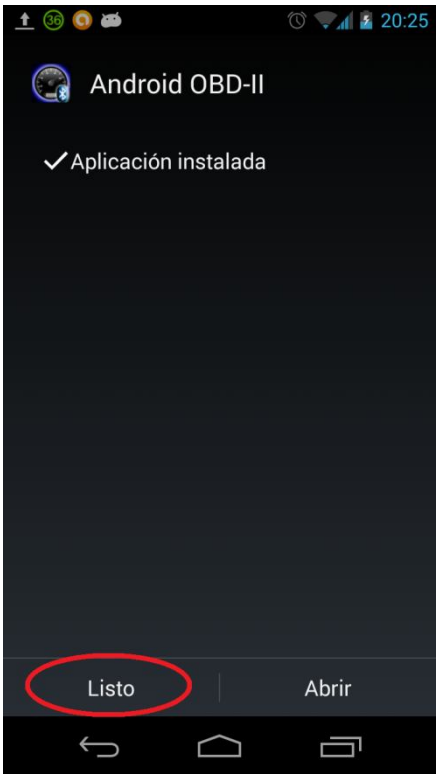
Seleccionar el fichero pulsando sobre el icono para comenzar su instalación.

- Paso 3: Instalación.



Una vez lanzada, aparece la pantalla de instalación donde se pueden ver los permisos requeridos.

Para continuar, hay seleccionar el botón *Instalar*.



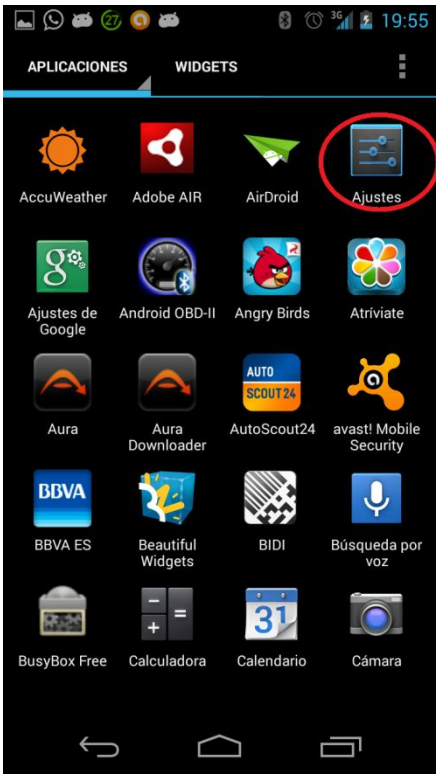
¡¡Enhorabuena!! La aplicación ha sido correctamente instalada.

Pulsar *Listo* para salir de la instalación.

2.2 DESINSTALACIÓN

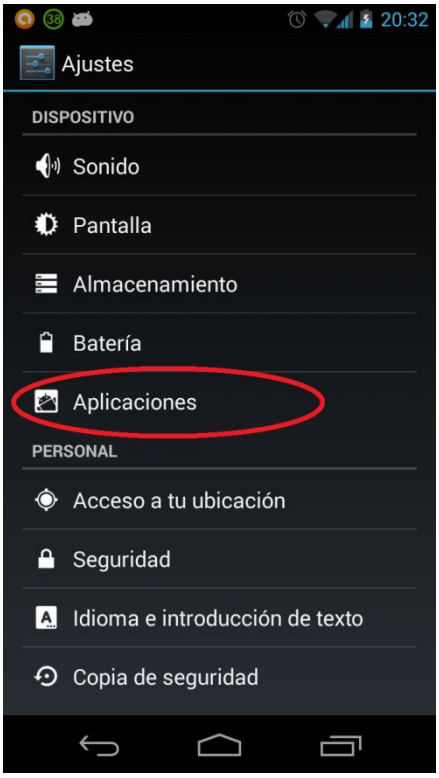
La desinstalación en Android es igual de sencilla que la instalación. Para realizar dicha acción hay que seguir los siguientes pasos:

- Paso 1: Buscar la aplicación en las Aplicaciones de Android.

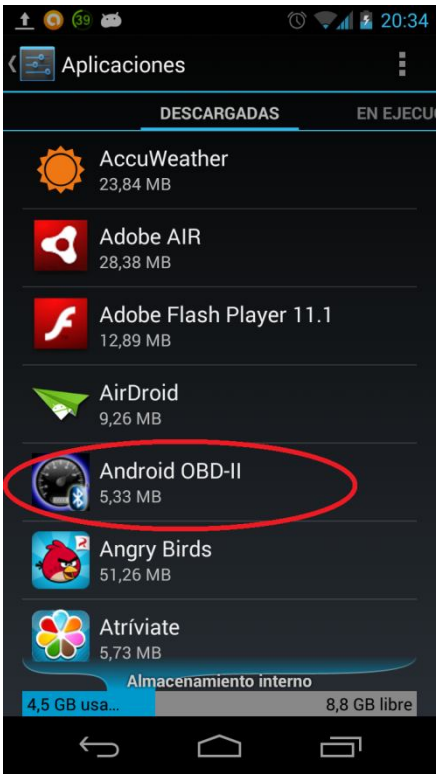


Lo primero es acceder a los *Ajustes* de Android.

Este acceso lo encontramos una vez que accedemos a las aplicaciones. En nuestro caso es el icono marcado con un círculo.

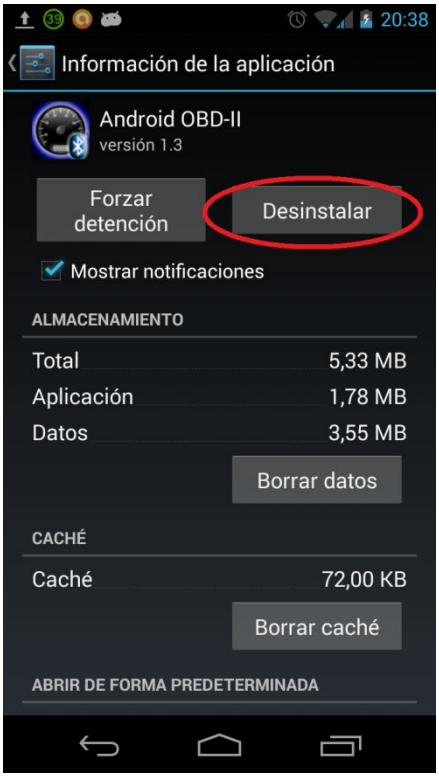


Una vez dentro de Ajustes, debemos seleccionar la opción de *Aplicaciones*.



Dentro de Aplicaciones encontraremos nuestra aplicación. Para seguir con la desinstalación, pulsar sobre ella.

- Paso 2: Desinstalar.



Para desinstalar la aplicación, simplemente hay que pulsar el botón Desinstalar y seguir los pasos indicados. El proceso es idéntico al de cualquier otra aplicación Android.

Muchas gracias por haber utilizado Android OBDII.

CAPÍTULO 3

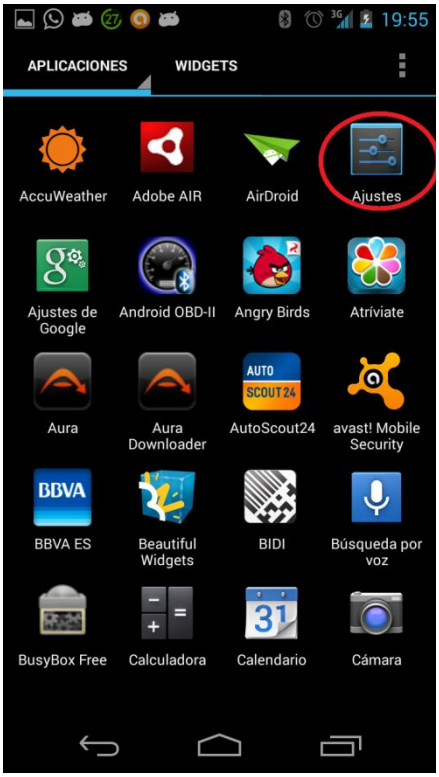
CONFIGURACIÓN

CAPÍTULO 3
CONFIGURACIÓN

3.1 PUESTA EN MARCHA

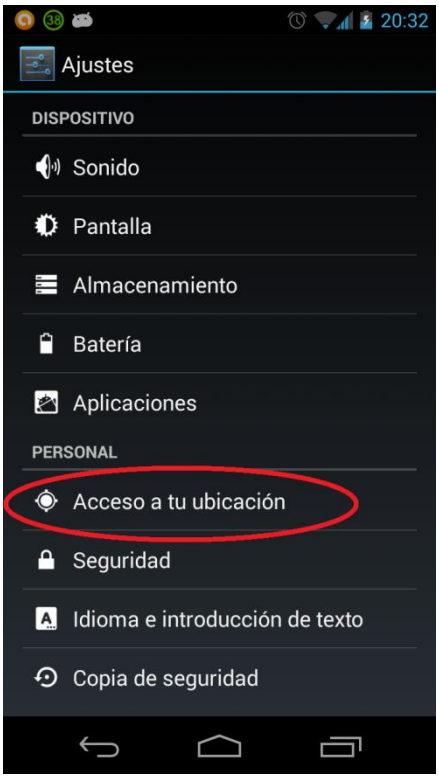
Antes de poner en funcionamiento la aplicación es básico realizar las siguientes tareas:

- Activar sensores de GPS y Bluetooth.



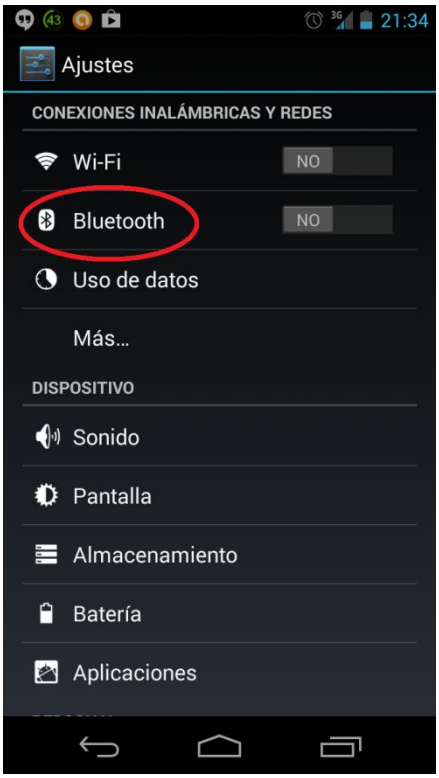
Para ello, lo primero es acceder a los *Ajustes* de Android.

Este acceso lo encontramos una vez que accedemos a las aplicaciones. En nuestro caso es el icono marcado con un círculo.



Para activar el GPS, seleccionar dentro de Ajustes, la opción *Acceso a tu ubicación*.

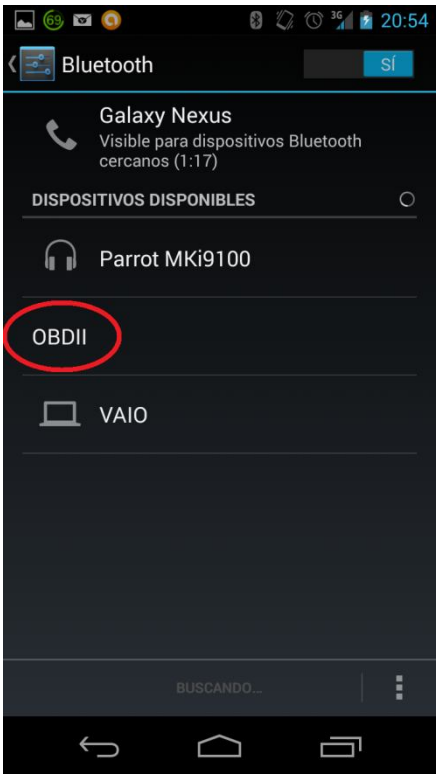
Para un correcto funcionamiento de la aplicación, las opciones *Permitir el acceso a tu ubicación* y *Satélites GPS* deben está activas.



Volvemos al menú principal de Ajustes para seleccionar ahora la opción de *Bluetooth*.

Si el bluetooth está desactivado, debe activarse, para poder conectar sincronizar el dispositivo con el conector OBD, como se indica en el paso siguiente.

- Paso 2: Sincronizar con el Conector OBD (solo necesario la primera vez).

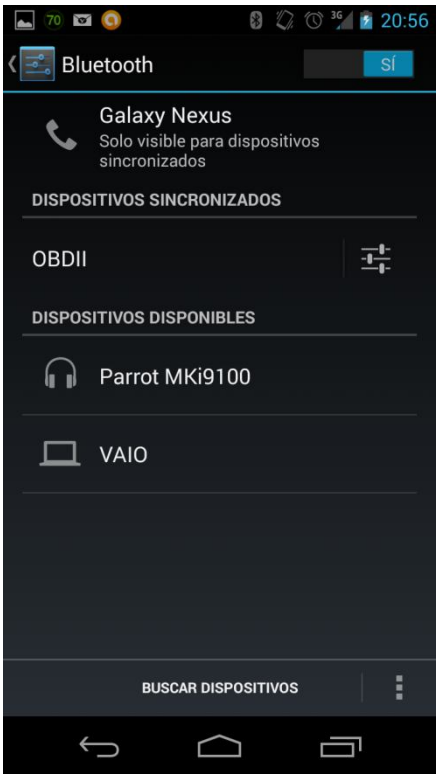


Una vez activado el bluetooth, aparecerán el listado de dispositivos disponibles.

Para que el Conector OBDII aparezca en este listado, éste debe estar conectado al vehículo y el vehículo debe estar arrancado.

Si selecciona el dispositivo OBDII pulsando se mostrará la ventana de vinculación.

En esta ventana se debe introducir el código *PIN* del conector, en mi caso "1234", y posteriormente pulsar en *Aceptar*.



Con esto hemos finalizado el proceso y el Conector OBDII ya está vinculado con nuestro dispositivo Android.

Este proceso solo hay que realizarlo la primera vez. Posteriormente, tan solo será necesario encender el Bluetooth para que se establezca la conexión entre el OBD y el teléfono.

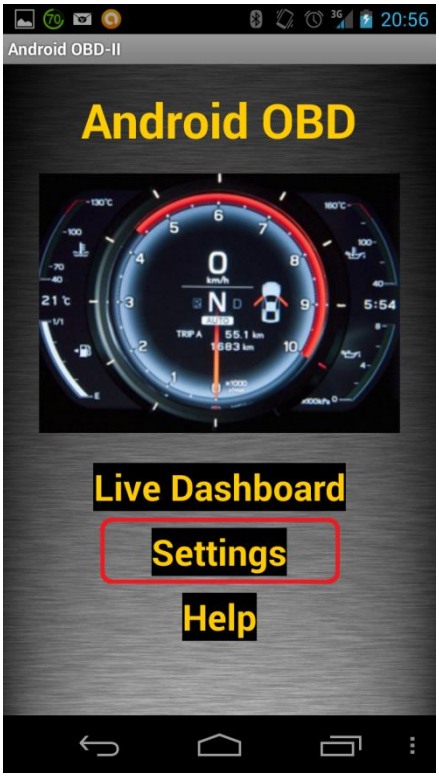
3.2 CONFIGURACIÓN

Ahora ya tenemos el dispositivo con los sensores necesarios encendidos y vinculado al conector OBDII, por lo que ya podemos acceder a la aplicación y realizar la configuración inicial para que el sistema funcione según nuestras necesidades. A continuación se describe como se puede realizar la configuración inicial del sistema:

- Acceso a la aplicación.

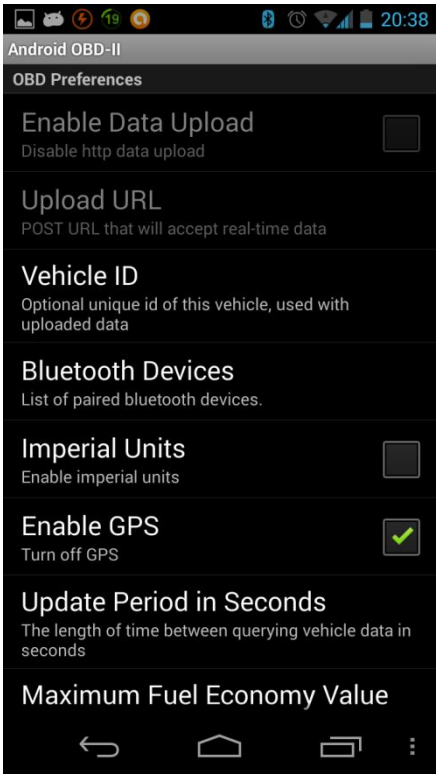
Desde el escritorio de Android o bien desde las aplicaciones, se accede a la aplicación pulsando sobre el icono de la aplicación.

- Acceso a la configuración de la aplicación.



Ya estamos dentro de la aplicación. Para poder realizar la configuración inicial debemos pulsar en el botón *Settings* de esta pantalla inicial.

- Realizar la configuración.



Ya estamos en la pantalla de configuración y no encontramos con las siguientes preferencias:

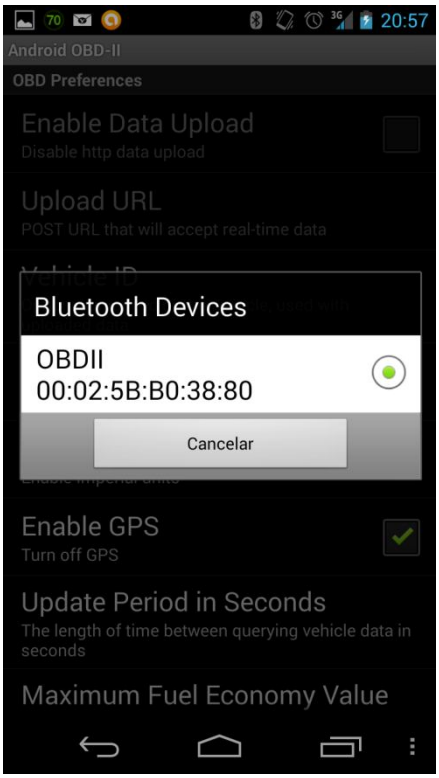
Vehicle ID: Parámetro opcional para indicar el identificador del vehículo.

***Bluetooth Devices*: Lista seleccionable de dispositivos enlazados (ver siguiente pantalla).**

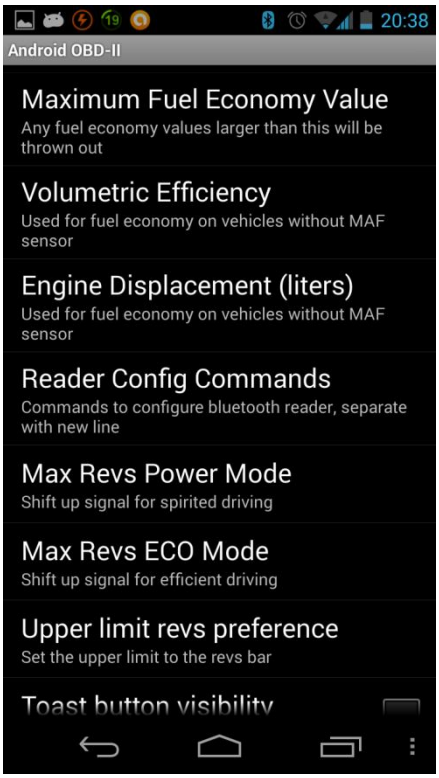
Imperial Units: Seleccionar unidades (por ahora, solo disponible en unidades métricas).

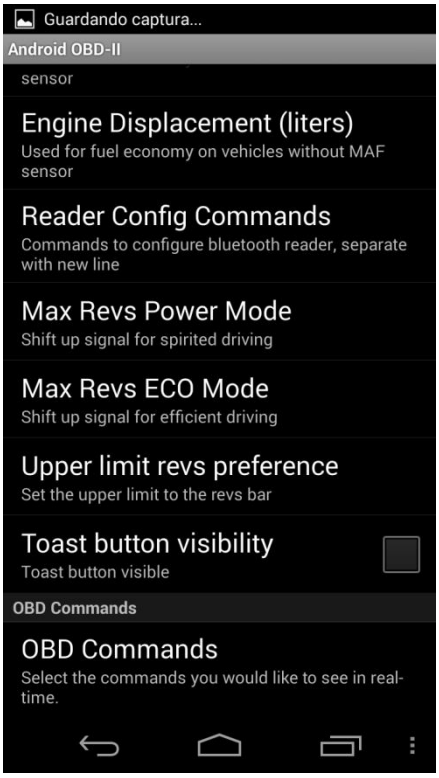
Enable GPS: Activar/desactivar GPS.

Update Period in Seconds: Por defecto aparecerá 1. Se recomienda para una monitorización más dinámica indicar el valor 0 (corresponde a 0,25 segundos) y para almacenar largos viajes un mayor valor, para no ocupar demasiado espacio.



Al acceder a la lista de dispositivos bluetooth enlazados *Bluetooth Devices*, es muy importante seleccionar el Conector OBD, que será al dispositivo al que se le solicitarán los parámetros del vehículo.





Maximum Fuel Economy Value: indica el valor máximo de consumo de combustible. Por defecto 70.

Volumetric Efficiency: Valor utilizado para calcular la eficiencia en vehículos sin sensor MAF (Mass Air Flow). Por defecto 0.85.

Engine Displacement: Cilindrada del motor. Por defecto 1.6.

Reader Config Commands: Comandos de configuración del lector OBDII. Se recomienda no modificar. Por defecto:

"atsp0
atz"

Max Revs Power Mode: marca el comienzo del rango para mostrar el indicador de cambio de marcha POWER. Por defecto 6000.

Max Revs ECO Mode: marca el comienzo del rango para mostrar el indicador de cambio de marcha ECO. Por defecto 2000.

Upper limit revs preference: Límite de revoluciones del vehículo. Marcará el valor máximo de la barra de revoluciones. Por defecto 7000.

Toast button visibility: Permite visualizar un botón para activar la depuración.

OBD Commands: Lista de parámetros OBDII disponibles.

CAPÍTULO 4

MANUAL DE USO

CAPÍTULO 4
MANUAL DE USO

4.1 PANTALLA INICIAL

Esta es la pantalla con la que accedemos al sistema. Ofrece una serie de botones para acceder a las distintas funcionalidades de la aplicación:

Ofrece los siguientes 3 botones para acceder a las distintas funcionalidades de la aplicación:

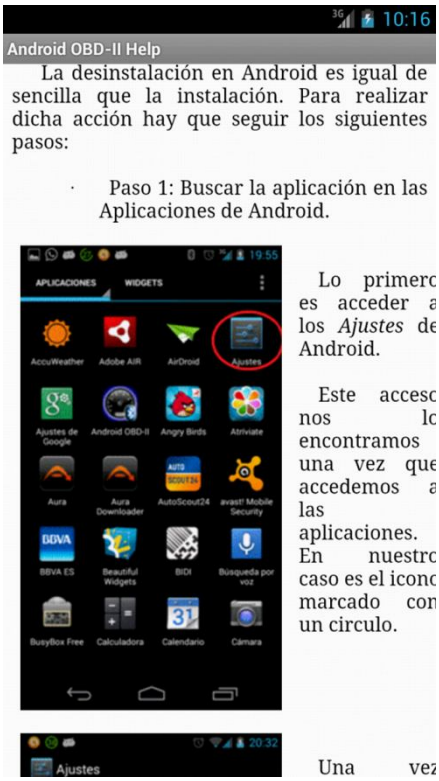
Live Dashboard: Da acceso a la monitorización, que es la funcionalidad principal del sistema.

Settings: Da acceso a la pantalla de configuración, que hemos visto en el capítulo anterior.

Help: Da acceso a la pantalla de ayuda, donde se visualizará un manual para resolver al usuario las posibles dudas que le puedan surgir.

4.2 PANTALLA DE AYUDA (HELP)

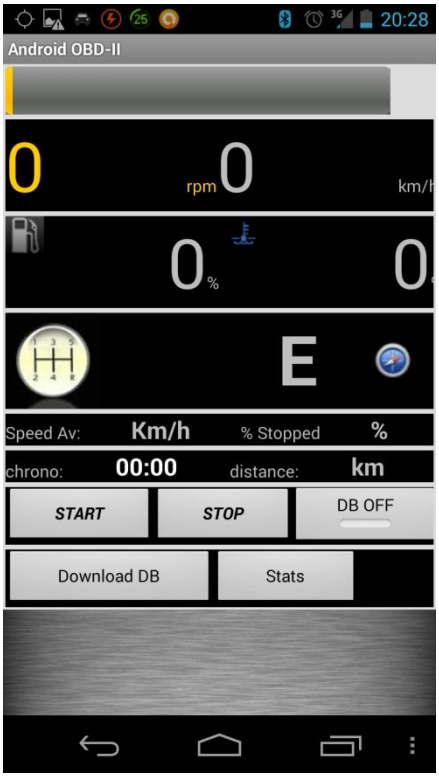
Esta pantalla de ayuda incorpora la visualización de un manual que el usuario puede consultar para poder resolver las dudas que le puedan surgir:



Ofrece la visualización del manual de usuario de la aplicación.

4.3 PANTALLA DE MONITORIZACIÓN (LIVE DASHBOARD)

Esta es la pantalla de monitorización en tiempo real de la aplicación, que para una mejor comprensión merece la pena explicar campo por campo y exponer varios ejemplos, como se hace a continuación:



Ofrece la visualización de los parámetros que a continuación se explican:

1. *Barra de revoluciones por minuto*:
   Gráfica para indicar el número de revoluciones por minuto. Su límite se establece por configuración con el parámetro *Upper limit revs preference.*

2. *Revoluciones por minuto*:
   Campo textual para mostrar el número de revoluciones por minuto, extraída del OBDII.

3. *Velocidad instantánea*:
   Campo textual para mostrar la velocidad instantánea, extraída del OBDII.

4. *Nivel de combustible*:
   Campo textual para mostrar el nivel del tanque de combustible, extraída del OBDII.

5. *Temperatura de refrigerante*:
   Campo textual para mostrar la temperatura de refrigerante, extraída del OBDII. Cambia de color para visualizar fácilmente si la temperatura es baja (azul), es normal (blanco) o es muy alta (rojo).

6. *Indicador de cambio de marcha*:
   Indicador para recomendar el cambio de marcha. Aparece una flecha verde, para recomendar una conducción eficiente y para recomendar el cambio de marcha en conducción deportiva, aparece una flecha roja. Los parámetros *Max Revs Power Mode* y *Max Revs ECO Mode* permiten su configuración.

7. *Brújula*:
Campo textual para mostrar la orientación del vehículo.

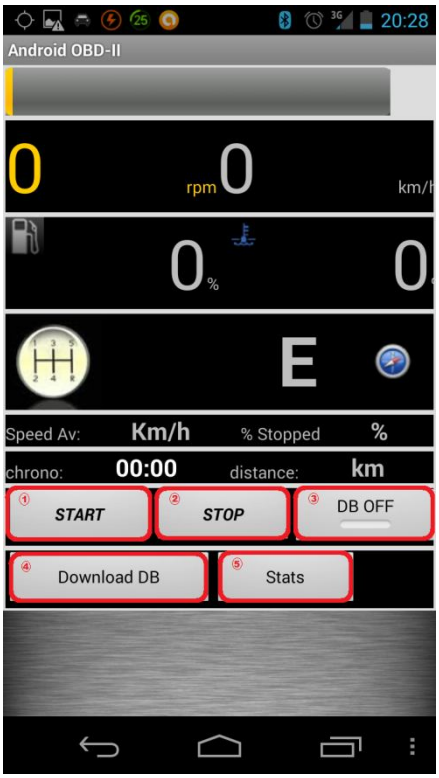
8. *Velocidad media*:
Campo textual para mostrar la velocidad media calculada por la aplicación a partir de la media de todas las velocidades extraídas.

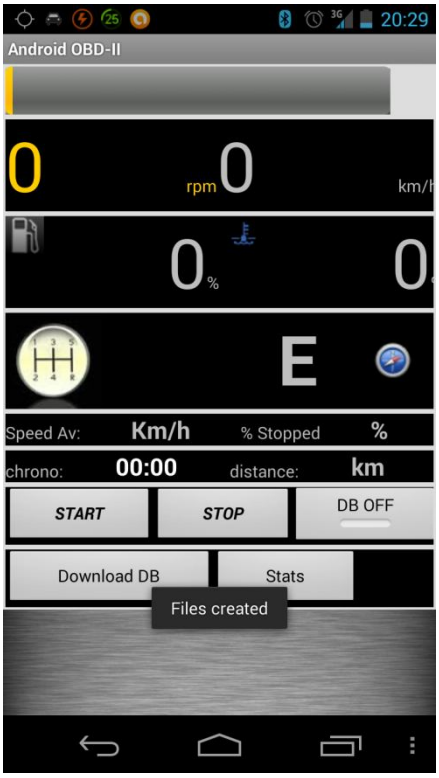
9. *Porcentaje de tiempo detenido*:
Campo textual para mostrar el porcentaje de veces que el vehículo ha estado detenido, respecto al total del viaje.

10. *Cronómetro*:
Campo textual para mostrar el tiempo transcurrido de viaje.

11. *Distancia*:
Campo textual para mostrar la distancia de viaje recorrida, calculada por GPS.

12. *Tabla de parámetros auxiliares*:
Tabla para mostrar datos adicionales extraídos del OBDII, que por su poca relevancia no han sido incluidos como campos principales.





Las posibles acciones que esta pantalla ofrece, vienen proporcionadas por los siguientes botones:

1. *Start*:
   El botón Start permite iniciar la monitorización del viaje.
   Si el viaje estuviera en marcha, o alguno de los sensores/conexiones no estuvieran disponibles, no se permitirá iniciarlo.

2. *Stop*:
   El botón Stop permite detener la monitorización del viaje.

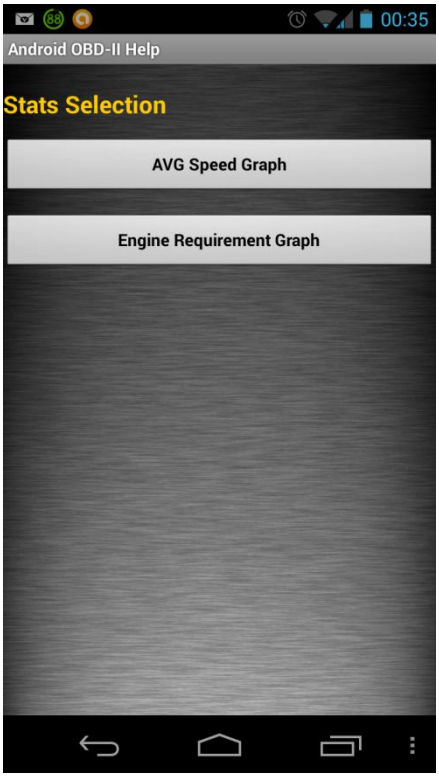
3. *DB ON/OFF*:
   Este botón permite activar y desactivar el registro de la información monitorizada en base de datos, para su posterior consulta.
   Se permite su activación en cualquier momento del viaje, incluso aunque ya esté iniciado. Esto posibilita en viajes largos, guardar el viaje antes de finalizar, almacenando los datos globales sin mermar apenas la capacidad de almacenamiento y permitiendo incluir el viaje en las estadísticas.

4. *Descarga de Base de datos*:
   Este botón permite la exportación en formato csv a la memoria externa del dispositivo. Los ficheros generados son *travel.csv*, *sequence.csv*, *command.csv* y *route.csv*; y permiten compartir esta información con otras plataformas para poder realizar análisis más exhaustivos.

5. *Estadísticas*:
   Este botón da acceso a la pantalla de selección de estadísticas, que se explicará en los siguientes puntos.

4.4 PANTALLA DE SELECCIÓN DE ESTADÍSTICAS (STATS SELECTION)

Es una simple pantalla de selección de las estadísticas que se quieren visualizar. Consta tan solo de 2 botones, que dan acceso a las dos graficas disponibles, pero se deja espacio suficiente para poder incluir mas en futuras versiones:



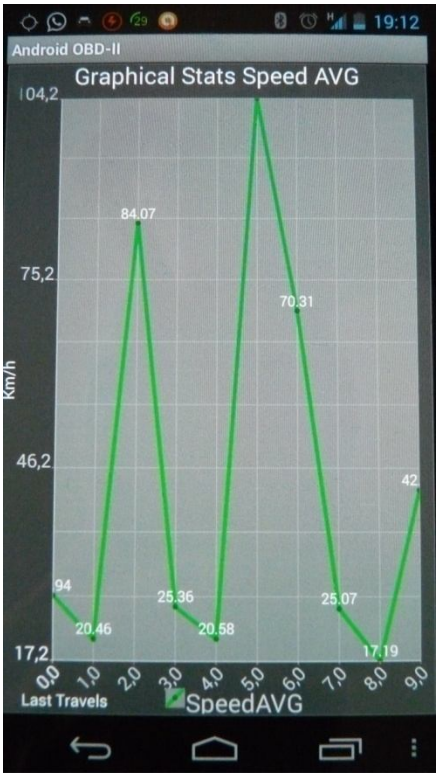
Las opciones que ofreces son:

AVG Speed Graph: Grafica estadística, que muestra las velocidades medias de los últimos viajes.

Engine Requirement Graph: Grafica estadística, que muestra el grado de exigencia al que motor ha sido sometido en los últimos viajes.

4.5 GRÁFICA DE VELOCIDAD MEDIA (AVG SPEED GRAPH)

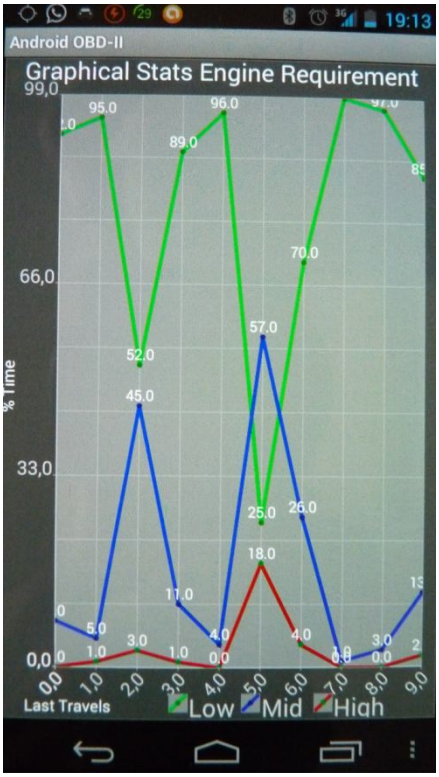
Esta pantalla muestra la gráfica lineal de la velocidad media de los últimos diez viajes:



Gráfica lineal, donde la línea indica la velocidad media de cada unos de los últimos viajes.

4.6 GRÁFICA DE EXIGENCIA DE MOTOR (ENGINE REQUIREMENT GRAPH)

Esta pantalla muestra la gráfica lineal del porcentaje (%) de nivel de exigencia requerido al motor de los últimos diez viajes. Es decir, que porcentaje (%) del viaje se ha conducido dentro de cada nivel de exigencia (bajo, medio, alto), en función del rango de revoluciones del motor:



Esta gráfica lineal, consta de 3 líneas indicadoras, cada una de las cuales significa:

- *Roja*: % viaje en el que el nivel de exigencia del motor ha sido máxima.

- *Azul*: % viaje en el que el nivel de exigencia del motor ha sido media.

- *Verde*: % viaje en el que el nivel de exigencia del motor ha sido baja, promoviendo así una conducción eficiente.

Estos valores se obtienen para los últimos viajes, de modo que se pueda comparar visualmente la eficiencia en la conducción entre ellos. De ello podemos deducir, que:

- Unos valores altos en la línea roja, se corresponden a una conducción deportiva.
- Los valores altos de la línea azul, pueden corresponder a una conducción despreocupada o bien, podrían corresponder a viajes por autopista.
- Unos valores altos en la línea verde, se corresponden a una conducción eficiente, en busca del menor consumo de combustible.