Universidad de Valladolid

Escuela de Ingeniería Informática de Segovia

Trabajo Fin de Grado

Grado en Ingeniería Informática
de Servicios y Aplicaciones

Plataforma para la captura, procesamiento y visualización de precios de carburantes mediante una aplicación móvil con Flutter

---

Alumno: Pablo Martínez Pérez

Tutores: Aníbal Bregón Bregón

Pedro César Álvarez Esteban

Plataforma para la captura, procesamiento y visualización de precios de carburantes mediante una aplicación móvil con Flutter

Pablo Martínez Pérez

Agradecimientos

Este proyecto ha podido salir adelante gracias a todas las personas que me han apoyado durante este trayecto. Gracias en primer lugar a mis tutores Aníbal Bregón y Pedro César, por proponerme y prestarse a dirigir este bonito proyecto. A todos los profesores que durante estos años han sabido transmitir su conocimiento, sin el cual sería imposible haber realizado este proyecto.

En lo personal, agradecer a mis compañeros, mi familia y pareja por haberme apoyado a lo largo de esta etapa.

Resumen

A pesar del ascenso de la flota eléctrica en el parque automovilístico español, en 2019 solamente un 1,4 % de las ventas fueron coches 100 % eléctricos. Mientras se sigue esperando que estas cifras aumenten, el consumo medio por familia en carburante se mantiene por encima de los 1.200 euros anuales. Estas cifras invitan a pensar que proporcionar herramientas tecnológicas a los consumidores para disminuir su factura mensual es una opción de negocio viable.

El presente proyecto persigue la construcción de una sistema capaz de capturar y procesar en tiempo real el precio de los carburantes en todas y cada una de las gasolineras del estado español. Para ello se construirán varios servicios independientes capaces de, por un lado, descargar la información y, por otro, procesarla para ser guardada de forma estructurada en una base de datos. Otro de los servicios permitirá consultar esta información a través una interfaz de programación de aplicaciones o API de tipo REST.

Este sistema se complementará con una aplicación móvil utilizando el framework Flutter para la visualización de las estaciones de servicio y sus precios. La utilización de este framework permitirá que la aplicación esté disponible para las plataformas iOS y Android al mismo tiempo.

Palabras claves: Flutter, Gasolineras, API REST, carburantes.

Abstract

Despite the rise of the electric fleet in the Spanish car park, in 2019 only 1,4 % of sales were electric cars. While these figures are still expected to increase, the average fuel consumption per family remains above 1.200 euros per year. These figures suggest that providing technological tools to consumers to reduce their monthly bill is a viable business option in the medium term.

This project seeks to build a system capable of capturing and processing the price of fuel in real time at each and every gas station throughout Spain. To achieve this goal, several independent systems will be built capable of, on the one hand, downloading the information and, on the other, processing it to be stored in a structured way in a database. Another service will allow the usr to consult this information through an application programming interface or REST-type API.

This system will be complemented with a mobile application using the Flutter framework to display the location of gas stations and their prices. The use of this framework will allow the application to be available for both iOS and Android platforms at the same time with the same user interface.

Keywords: Flutter, Petrol Stations, API REST, fuels.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Según la última públicación sobre presupuestos familiares del Instituto Nacional de Estadística, el gasto medio que realizan las familias españolas al año en carburante asciende a más de 1.200 euros [1]. Por tanto, cualquier herramienta tecnológica que permita ahorrar a la hora de repostar tendrá un impacto positivo en la economía española.

El sector de la energía en España supone aproximadamente un 3 % del PIB del país [2]. En particular, según la Corporación de Reservas Estratégicas de Productos Petrolíferos, en el 2019 España importó 66,3 millones de toneladas de crudo, equivalente a unos 500 millones de barriles, por un precio aproximado de 60 dólares el barril [3]. Esto supone un gasto para nuestro país de algo más de 30.000 millones de dólares, en su mayoría destinado al transporte por carretera.

Por otro lado, el Ministerio para la Transición Ecológica y el Reto Demográfico dispone de un portal [4] en el cual expone de forma pública los precios de los carburantes en todas las gasolineras de España. Es responsabilidad de las gasolineras informar a través de este portal con al menos 12 horas de antelación cualquier cambio en el precio de carburantes. Incumplir esta normativa puede suponer graves sanciones. Es por ello que el portal cuenta con información actualizada en todo momento y la proporciona de manera pública a través de varios métodos, siendo uno de ellos la publicación de un fichero Excel que se actualiza cada hora. Este fichero cuenta con información de cada una de las gasolineras de España (coordenadas, provincia, dirección, horarios, etc.), los carburantes que vende y sus precios. Esta es la fuente de datos con la que se trabaja en este proyecto.

El objetivo final del proyecto es la elaboración de una plataforma que capture y procese estos datos para luego exponerlos a través de un servicio REST y visualizarlos a través de una aplicación móvil.

1.1. Motivación

Este proyecto surge como propuesta del profesor del grado en Matemáticas de la Universidad de Valladolid Pedro César Álvarez Esteban. Este lleva desde 2010 recolectando datos de precios de carburantes en España con un total de más de 4.000 ficheros Excel, donde cada uno de estos ficheros cuenta con más de 9.000 filas. Sin embargo, estos datos necesitaban ser procesados y estructurados, pues por sí mismos no tienen valor. La propuesta original consistía en la elaboración de un Trabajo de Fin de Grado en matemáticas que predijese el precio del carburante a corto plazo, pero inmediatamente surge la idea de enlazar este proyecto con una aplicación móvil que muestre los precios en tiempo real y que sirva como Trabajo de Fin de Grado en informática.

Este proyecto además puede llegar a tener un impacto real para sus usuarios, pues permitirá encontrar el punto de repostaje más económico y cercano. Como ya se ha mencionado, el gasto medio en España en carburante asciende a más de 1.200 euros anuales. El desarrollo de una aplicación que permita disminuir el gasto mensual en hidrocarburos supone un aliciente pues trae consigo, en estos tiempos de crisis, una redistribución del gasto que permitirá invertir este ahorro en otros fines más necesarios o productivos.

1.2. Alcance del proyecto

Inicialmente el proyecto tiene un alcance nacional, esto es, solo se es están procesando y almacenando precios de gasolineras en España. No obstante, en otros países como Alemania o Italia existen portales que ofrecen también precios en tiempo real a través de ficheros Excel o APIs. Si el proyecto es exitoso, el sistema está preparado para poder integrarse con estos portales extranjeros de manera sencilla. Además está previsto que el sistema cuente con un sistema de internacionalización que permita añadir nuevos idiomas a la aplicación. Es por ello que se ha construido un sistema flexible y escalable que se adapte fácilmente ante un cambio de demanda a medio plazo. Más aún, debido a que el sistema se ha construido utilizando tecnologías híbridas, si el proyecto es exitoso en el ecosistema Android, se puede poner a disposición de los usuarios del Sistema Operativo iOS sin apenas esfuerzo.

Asimismo, el proyecto puede hacerse más completo con el tiempo, pudiéndose incluir más filtrados de búsqueda, un menú de configuración, un sistema de registro o incluso llegando a monetizar vía publicidad o algún sistema de suscripción.

Igualmente, en el futuro serán importantes los puntos de recarga eléctricos. Por ahora el portal desde el que se obtienen los datos no muestra información sobre estos puntos de recarga, pero cuando lo haga, el sistema está preparado para poder incluir esta información adicional.

1.3. Objetivos del trabajo

El objetivo principal del trabajo es la elaboración de una plataforma o conjunto de sistemas que por un lado permitan capturar y guardar precios de carburantes en tiempo real, y que por otro lado permitan visualizar estos precios a través de una aplicación móvil que consuma los datos mediante un servicio REST. Se pretende la aplicación móvil sea utilizada para ahorrar a la hora de repostar, para lo cual la deberá mostrar un mapa con iconos de gasolineras y sus precios, clasificando las gasolineras en tres grupos: económicas, costosas e intermedias.

Tradicionalmente los consumidores han tenido problemas a la hora de comparar precios y conocer los distintos puntos de repostaje. Esto hace que muchos usuarios no puedan comparar y desconozcan el ahorro que puede suponer repostar en estaciones de servicio más económicas. Este problema se agudiza cuando se viaja, puesto que habitualmente se desconoce la ubicación y precio de las estaciones de servicio a lo largo del trayecto y en el destino. El presente proyecto pretende dar respuesta a estos problemas.

Para lograr implementar la plataforma será necesario desarrollar con éxito dos sistemas que precedan a la aplicación móvil. Uno encargado de descargar, estructurar y almacenar la información obtenida desde el portal proveedor de los datos y un segundo sistema encargado de proporcionar esta información mediante un servicio REST. Por tanto se han identificado tres subobjetivos para este proyecto:

- Creación de un sistema que automatice la descarga de ficheros Excel cada hora desde el portal del Ministerio para la Transición Ecológica y el Reto Demográfico, procese y limpie los datos de estos ficheros e ingeste esa información en una base de datos.
- Creación de un servicio REST que exponga esta información para poder ser consumida desde la aplicación móvil o cualquier otro cliente. Este servicio REST estará alojado en una aplicación web.
- Creación de una aplicación móvil que muestre la información de los precios de las gasolineras en tiempo real.

La creación de un sistema automático de descarga, limpieza, estructurado e inserción de datos ya tiene valor por sí mismo, pues proporciona información en tiempo real e histórica a través de un gestor de bases de datos frente a únicamente información en tiempo real y sin estructurar en un fichero Excel. Si a este sistema automático se le añade un servicio REST, estamos frente a un proyecto que expone los datos a través de una API pública en tiempo real además de ofrecer un histórico de los datos. De nuevo este sistema tiene un gran valor añadido, pues podría ser utilizado por terceras aplicaciones frente al arcaico sistema proporcionado por el portal del

Ministerio para la Transición Ecológica y el Reto Demográfico. No parece existir ningún otro sistema REST que exponga estos datos de forma pública. Por último, al añadir la aplicación móvil se está creando un sistema complejo que da al usuario un servicio completo y valioso.

1.4. Organización de la memoria

El presente documento se encuentra dividido en nueve capítulos y sus respectivas secciones, un apartado de bibliografía y los apéndices. A continuación se realiza una breve descripción del contenido de cada uno de los apartados:

- **Introducción:** En este capítulo se introduce al lector en los conceptos fundamentales del proyecto. Además se expone el porqué del trabajo, la motivación, alcance, identificación de los usuario y objetivos del proyecto.
- **Metodología:** El capítulo 2 incluye la metodología utilizada durante el desarrollo del proyecto y las herramientas y tecnologías utilizadas. También incluye una estimación del esfuerzo que va a suponer, junto a una planificación temporal y un presupuesto económico.
- **Estado del arte:** Este tercer capítulo incluye una contextualización del proyecto frente a otras alternativas existentes en el mercado. Se estudiarán tanto aplicaciones Android como aplicaciones iOS.
- **Fuentes de datos:** El cuarto capítulo explica cómo es la estructura de datos que provee de forma pública el portal geogasolineras y cómo se accede a esos datos.
- **Análisis:** En este capítulo se detalla la funcionalidad que deberá cumplir el sistema a través de la especificación de requisitos: funcionales, no funcionales y de información.
- **Diseño:** En este capítulo se expone el trabajo de diseño realizado previo a la creación del sistema. Se tratarán tanto las arquitecturas lógicas y físicas, como los diseños de algunas interfaces. También incluye diversos diagramas que facilitan la comprensión del funcionamiento de la aplicación.
- **Implementación:** En este capítulo se especifica a nivel técnico la implementación llevada a cabo.
- **Pruebas:** En este capítulo se muestran las pruebas que se han realizado para comprobar el correcto funcionamiento del sistema.
- **Documentación:** Este capítulo contiene las guías para el despliegue de la aplicación y para su uso.

- **Conclusiones y trabajo futuro:** Este último capítulo recoge las reflexiones finales tras el desarrollo del proyecto y las líneas futuras a seguir para su mejora.

Capítulo 2

Metodología, planificación y presupuestos

2.1. Herramientas y tecnologías utilizadas

En esta sección se detallan tanto las herramientas utilizadas para la consecución de los objetivos generales del proyecto (tales como IDEs y editores de texto), como las tecnologías utilizadas para su desarrollo (lenguajes de programación, *frameworks* y librerías).

Las herramientas que se han utilizado son:

- **PyCharm**: PyCharm es un entorno de desarrollo integrado especialmente diseñado para optimizar el desarrollo en Python. Es propiedad de la empresa checa JetBrains. He utilizado la versión gratuita.
- **WinSCP**: cliente SFTP gráfico para windows por SSH. También es capaz de emplear el protocolo SCP. Se basa en la implementación del protocolo SSH de PuTTY y el protocolo FTP de FileZilla.
- **Texmaker**: editor LaTeX multiplataforma de código abierto con visor integrado de documentos PDF y soporte Unicode. Este editor ha sido utilizado para la elaboración del presente documento.
- **PuTTY**: Cliente SSH, Telnet, rlogin, y TCP raw con licencia libre. Se ha utilizado como cliente SSH para gestionar la máquina virtual proporcionada por la universidad.
- **Android Studio**: Android Studio es el entorno de desarrollo integrado (IDE) oficial para la plataforma Android, construido sobre el software IntelliJ IDEA de JetBrains y diseñado específicamente para el desarrollo de Android. Contiene una suite de plugins que facilitan

el desarrollo. En particular, se puede configurar de manera sencilla para trabajar con el SDK de flutter, el cual se ha utilizado en el desarrollo de este proyecto.

- **Notion**: Notion es una aplicación que proporciona componentes como bases de datos, tableros kanban, wikis, calendarios y recordatorios. Estos componentes se pueden utilizar para para crear sistemas de gestión del conocimiento propios, toma de notas, gestión de datos, gestión de proyectos, etc. Estos componentes y sistemas se pueden utilizar individualmente o en colaboración con otros. En particular, lo he utilizado como gestor de documentación técnica y para llevar la trazabilidad del estado del proyecto.

- **GitHub Desktop**: GitHub Desktop es una herramienta que permite interactuar con GitHub desde el escritorio. Hace uso del sistema de gestión de versiones git, y se ha utilizado para mantener un histórico de las distintas iteraciones del proyecto.

- **HeidiSQL**: HeidiSQL es una herramienta de administración de bases de datos gratuita y de código abierto para MySQL y sus bifurcaciones. Se ha utilizado para administrar las bases de datos, ver rendimientos, mejorar consultas, indexar, etc.

- **TexMaker**: TexMaker es una editor de texto LaTeX de código abierto con un visualizador de PDF integrado. Se ha utilizado para la elaboración de la memoria.

Por otro lado, las tecnologías que se han empleado para el desarrollo del proyecto son:

- **Python**: Lenguaje de programación interpretado de tipado dinámico, de alto nivel y de propósito general, cuya filosofía hace hincapié en una sintaxis que favorezca el código legible. Se trata de un lenguaje de código abierto, multiplataforma y multiparadigma, soportando orientación a objetos, programación imperativa y programación funcional. En particular, se ha utilizado para escribir scripts de automatización y programar la aplicación.

- **Pandas**: librería de código abierto de Python que provee estructuras de datos rápidas y flexibles diseñadas para trabajar con datos estructurados y series temporales de forma sencilla e intuitiva. Entre las principales características que aportan mayor valor al actual proyecto destacan los tipos de datos *DataFrame* que permite trabajar a grandes velocidades con datos estructurados e indexados.

- **Flask**: Flask es un micro web framework escrito en Python. Se clasifica como un microframework porque no requiere herramientas o bibliotecas particulares. No tiene una capa de abstracción de base de datos, validación de formularios ni ningún otro componente donde las bibliotecas de terceros preexistentes proporcionen funciones comunes. Sin embargo, Flask admite extensiones que pueden agregar funciones a la aplicación como si estuvieran

implementadas en Flask. Existen extensiones para mapeadores relacionales de objetos, validación de formularios, manejo de carga, varias tecnologías de autenticación abierta, etc. En particular, se ha utilizado para escribir la aplicación web y el servicio REST.

- **MariaDB**: MariaDB es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL (General Public License). El motor de almacenamiento que se ha utilizado ha sido Aria, que reemplaza al tradicional MyISAM. Tiene una alta compatibilidad con MySQL ya que posee las mismas órdenes, interfaces, API y bibliotecas.

- **Apache**: Servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual según la normativa RFC 2616. El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd).

- **GitHub Actions**: GitHub Actions es una API que permite orquestar cualquier flujo de trabajo, en función de cualquier evento. Se ha utilizado para hacer integración continua en el proyecto. Esto es, cada vez que se añade un nuevo *commit* en github, este código automáticamente se envía a la máquina virtual y se reinicia el servidor web. En consecuencia, cualquier actualización de código, se ve reflejada en cuestión de segundos de forma automática.

- **Flutter**: Flutter es un kit de desarrollo de software de interfaz de usuario de código abierto creado por Google. Cuenta con multitud de librerías para extender su funcionalidad. Su principal característica es que permite compilar el mismo código para distintas plataformas. Permite desarrollar aplicaciones para Android, iOS, Linux, Mac, Windows, Google Fuchsia, y la web. Flutter se ha utilizado para desarrollar la aplicación móvil.

2.2. Ciclo de Trabajo

La metodología empleada durante el desarrollo del producto de este trabajo se basa en una metodología incremental. Esta metodología divide el desarrollo del proyecto en varios fragmentos que se denominan *incrementos*, que no son otra cosa que iteraciones que van añadiendo funcionalidad o mejoras al producto final.

En cada iteración se busca poder entregar un producto completamente funcional, hasta llegar a la solución completa que se pretende obtener. En la primera iteración se desarrolla una arquitectura completa del sistema. Una vez decidida la arquitectura se tienen que desarrollar

tres grandes subproyectos. Por tanto, primero se va a desarrollar el sistema automático de descarga, limpieza y estructura de ficheros Excel e inserción en la base de datos. A continuación se implementará el servicio REST y por último la aplicación móvil. En cada uno de los proyectos se utilizará una metodología incremental y se podrán modificar los proyectos ya finalizados para añadir nuevas funcionalidades según se vaya requiriendo.

Cada incremento requerirá de una serie de etapas (se desarrollarán incrementos hasta obtener un producto final satisfactorio):

- **Análisis:** Se desarrolla la especificación de las principales características. En cada incremento sucesivo se añadirán o cambiarán requisitos en base a las necesidades del proyecto.
- **Diseño:** Una vez las funcionalidades han sido analizadas, se procede al diseño de los componentes especificados en el análisis.
- **Implementación:** Se ha de implementar un prototipo funcional que cumpla las especificaciones y sea completamente funcional.
- **Pruebas:** Se procede a la elaboración de los tests necesarios que comprueben el correcto funcionamiento del sistema.
- **Documentación:** Por último, se documenta todo el proceso desarrollado durante el incremento.

Durante el desarrollo se han utilizado algunas de las técnicas y herramientas habituales en la industria. En particular se ha utilizado un sistema de workflow y branding llamado gitflow workflow, consistente en crear commits de Git en una rama de desarrollo llamada *develop* y en hacer un merge de esa rama hacia *master* con cada nueva versión o release. Gracias a este flujo, se pueden automatizar procesos con facilidad.

En cuanto a las pruebas y el despliegue automático, se ha utilizado *GitHub Actions*. Con cada *commit* en el repositorio, se ejecutan una serie de pruebas, que de ser exitosas permiten que los cambios implementados se actualicen automáticamente en el servidor mediante el protocolo ssh y el cifrado de clave pública-privada. Esto significa que los cambios que se hacen en local, pasan una serie de tests y en caso de ser exitosos se envían al servidor de manera automática. Estos procesos automatizados, aunque costosos de implementar al principio, permiten que el desarrollo de software sea rápido y efectivo.

2.3. Planificación temporal

El proyecto se inicia a principios de Julio de 2020 y se estima finalice a principios de Junio de 2021. Esto es prácticamente un año natural. Este lento desarrollo se debe a múltiples factores. El

presente proyecto son realmente tres trabajos en uno, lo que multiplica el tiempo de desarrollo, pues no solo hay que programar los tres sistemas, sino también implementar la interacción entre los mismos. Por otra parte, se planifica tener que realizar una parada de aproximadamente siete semanas para preparar el examen de una asignatura del grado en matemáticas y desde Enero de 2021 este proyecto se ha compaginado con un trabajo a tiempo completo. Además, muchas de las tecnologías utilizadas eran desconocidas, por lo que se ha tenido que invertir un tiempo extra en estudiarlas y aprender a manejarlas.

Este proyecto tiene 5 grandes incrementos. El primer incremento consiste en la elaboración del sistema automático de descarga de ficheros Excel, limpieza y estructurado de datos para posteriormente guardarlos en una base de datos. El segundo incremento consiste en la elaboración del sistema REST y el tercero en la aplicación móvil. El cuarto incremento es una revisión general de todo lo anterior. Se prentede aquí dedicar tiempo comprobar que la integración de los sistema es correcta y que el diseño y análisis han sido correctos. El quinto y último incremento es el desarrollo de los modelos de predicción. Estos incrementos pueden visualizarse en la figura 2.1

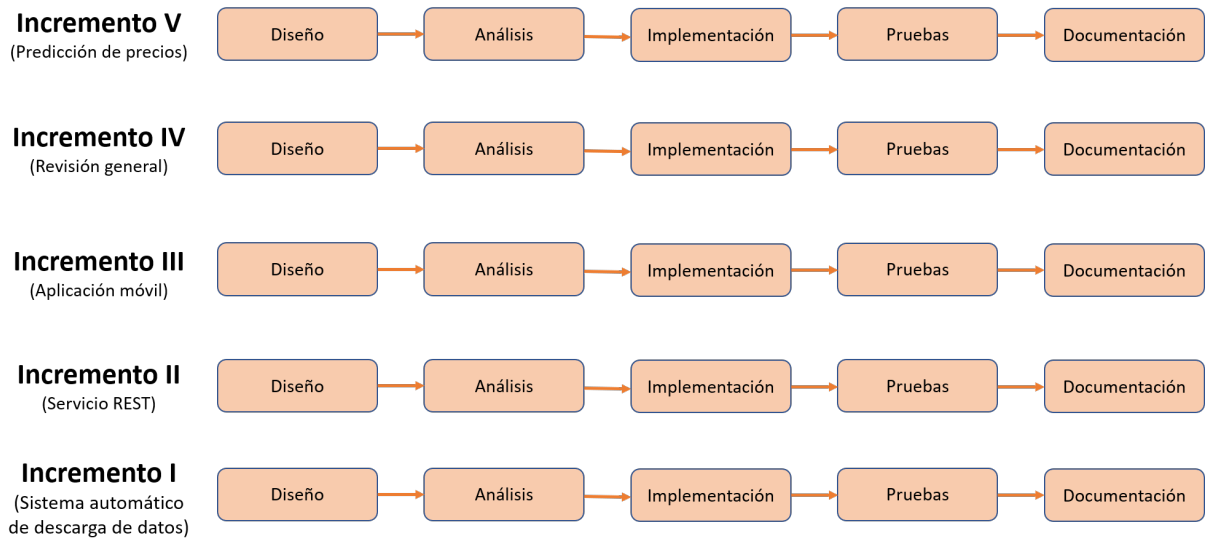


Figura 2.1: Diagrama de incrementos y fases

Cada uno de los incrementos incluye no solo desarrollar el propio incremento, sino integrarlo en el sistema para que se comunique con el resto de servicios. Por supuesto hay un último incremento que no se incluye en el gráfico por ser más liviano y que no sigue este patrón: el de hacer una revisión general del sistema final una vez que todos los servicios funcionan de forma coordinada.

En la figura 2.2 se muestra la planificación dividida en iteraciones y fases en cada iteración.

El tiempo total es de 269 días y se prevé finalizar el día 31 de mayo de 2021. En la figura 2.3 se muestra esta misma información en un diagrama de Gantt.

	N° de días	Inicio	Final
ITERACIÓN I	**33**	**08/07/2020**	**09/08/2020**
Análisis	3	08/07/2020	10/07/2020
Diseño	3	11/07/2020	13/07/2020
Implementación	22	14/07/2020	04/08/2020
Pruebas	3	05/08/2020	07/08/2020
Documentación	2	08/08/2020	09/08/2020
ITERACIÓN II	**47**	**10/08/2020**	**25/09/2020**
Análisis	3	10/08/2020	12/08/2020
Diseño	3	13/08/2020	15/08/2020
Implementación	36	16/08/2020	20/09/2020
Pruebas	3	21/09/2020	23/09/2020
Documentación	2	24/09/2020	25/09/2020
ITERACIÓN III	**39**	**17/11/2020**	**25/12/2020**
Análisis	2	17/11/2020	18/11/2020
Diseño	2	19/11/2020	20/11/2020
Implementación	26	21/11/2020	16/12/2020
Pruebas	6	17/12/2020	22/12/2020
Documentación	3	23/12/2020	25/12/2020
ITERACIÓN IV	**6**	**26/12/2020**	**31/12/2020**
Análisis	1	26/12/2020	26/12/2020
Diseño	1	27/12/2020	27/12/2020
Implementación	1	28/12/2020	28/12/2020
Pruebas	1	29/12/2020	29/12/2020
Documentación	2	30/12/2020	31/12/2020
ITERACIÓN V	**144**	**08/12/2021**	**31/05/2021**
Análisis	5	08/12/2021	12/01/2021
Diseño	4	13/01/2021	16/01/2021
Implementación	124	17/01/2021	20/05/2021
Pruebas	5	21/05/2021	25/05/2021
Documentación	6	26/05/2021	31/05/2021

FIGURA 2.2: DETALLES DE PLANIFICACIÓN DE ETAPAS

En el diagrama de Gantt (Figura 2.3) se puede observar como desde el 26 de Septiembre al 16 de Noviembre de 2020 no se avanzó en el proyecto. Como ya se ha indicado, esto se debe a que el desarrollo se tiene que suspender temporalmente para afrontar el último examen del grado en matemáticas. También se prevé hacer un descanso a principio de año en 2021.

2.4. Estimación del esfuerzo

Existen multitud de técnicas para la estimación del coste temporal de un proyecto de software. En los últimos años se ha demostrado que las estimaciones tradicionales no se adecuan a las técnicas modernas de desarrollo de software, y es por ello que han surgido novedosas técnicas de medición de costes, muchas de ellas siguiendo los principios del manifiesto ágil [1]. Es también interesante mencionar el movimiento *#NoEstimates* que se enfoca en medir la entrega real de valor sin necesidad de hacer estimaciones.

[1] https://agilemanifesto.org/iso/es/manifesto.html

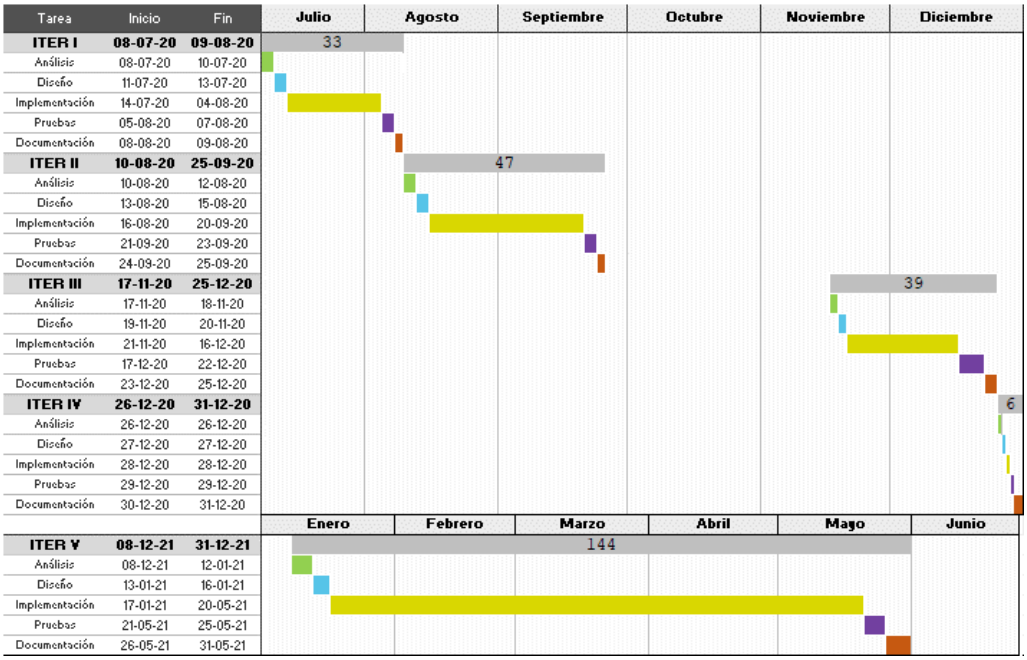


Figura 2.3: Diagrama de Gantt para la planificación

En este proyecto se va a hacer uso de unas de las técnicas de estimación estudiadas en el grado: técnica de Albrecht usando puntos de función. Cuando se estima coste temporal, el esfuerzo se expresa habitualmente en horas-persona u otra unidad, mientras que para la estimación de costo se utiliza la moneda local o el dolar.

2.4.1. Estimación por Puntos de Función

La estimación por puntos de función es una técnica de estimación de software desarrollada por Allan Albrecht en 1979 mientras trabajaba para IBM. Esta técnica permite medir la funcionalidad de un determinado software, independientemente de la tecnología usada para construirlo y las fases del ciclo de vida utilizado.

El problema de estimar mediante Líneas de Código es que puede no reflejar realmente la complejidad de la aplicación. Estimar por líneas de código depende en gran medida del programador ya que se pueden desarrollar algoritmos muy complejos en pocas líneas de código o viceversa. Es por ello que se ha decidido estimar según las funcionalidades del software mediante el método de Albretch. De esta forma, la estimación será independiente del lenguaje de programación y del tamaño en líneas de código.

A continuación se definen los flujos de información y su complejidad. Se indicará con una A si la complejidad es alta, con M si es media y con B si la complejidad es baja.

FLUJOS DE INFORMACIÓN							
Entradas Usuario		Salidas Usuario		Consultas		Ficheros lógicos externos	
Seleccionar carburante	B	Ver opciones	B	Precios	A	Base de Datos	A
Buscar gasolinera	A	Ver gasolineras	A	Ubicaciones	A		
Seleccionar gasolinera	M	Ver precios	A	Compañías	A		
Crear ruta	M	Ver compañía	A	Google Maps	M		
Modificar zoom	B						
Ir a la ubicación	B						

TABLA 2.1: FLUJOS DE INFORMACIÓN

Nótese que no aparece la opción de ficheros lógicos internos debido a que el sistema no hace ninguna consulta a este tipo de documentos. A continuación, una vez identificados estos flujos y otorgada su complejidad, se obtienen los puntos de función sin ajustar o *PFSA*, que se pueden observar en la tabla 2.2.

Puntos de Función Sin Ajustar (PFSA)			
Elemento	Complejidad	Número	Total
Entradas	Baja (×3)	3	9
	Media (×4)	2	8
	Alta (×6)	1	6
Salidas	Baja (×4)	1	4
	Media (×5)	0	0
	Alta (×7)	2	14
Consultas	Baja (×3)	0	0
	Media (×4)	1	4
	Alta (×6)	3	18
Ficheros externos	Baja (×5)	0	0
	Media (×7)	0	0
	Alta (×10)	1	10
TOTAL			73

TABLA 2.2: FLUJOS DE INFORMACIÓN

Una vez obtenidos los PFSA, se deben ajustar mediante el Factor de Ajuste (FA). El cálculo del FA está basado en 14 características generales de los sistemas que miden la funcionalidad general y complejidad/influencia de la aplicación. A cada característica se le atribuye un peso de 0 a 5 donde 0 es un factor sin influencia/complejidad y 5 es la máxima influencia/complejidad posible.

Factores de Ajuste	
Factor	Valor
Comunicación de datos	4
Procesamiento distribuido	5
Rendimiento	5
Desempeño crítico	4
Frecuencia de transacciones	5
Entrada de datos en línea	5
Transacción de entrada sobre pantallas múltiples	5
Actualización en línea	5
Procesos complejos	5
Facilidad de mantenimiento	4
Código diseñado para reutilización	4
Conversión / instalación en diseño	3
Instalaciones múltiples	4
Facilidad de cambios	4

Tabla 2.3: Flujos de información

Ahora se va a calcular el factor de ajuste (FA) apartir de la suma de los 14 factores de compejidad haciendo uso de la siguiente ecuación:

$$FA = 0{,}6 + 0{,}1 \cdot 0{,}01 \cdot \sum_{i=1}^{14} \text{complejidad}_i = 0{,}6 + 0{,}01 \cdot 62 = 1{,}22$$

Se calculan ahora los Puntos de Función Ajustados (PFA) como el producto de los Puntos de Función Sin Ajustar y el Factor de Ajuste:

$$\text{PFA} = \text{PFSA} \cdot FA = 73 \cdot 1{,}22 = 89{,}06$$

Asignando ahora un valor de un mes de esfuerzo por cada 8,5 puntos de función tenemos que el esfuerzo final medido en persona-mes es:

$$\text{Esfuerzo} = \frac{89{,}06}{8{,}5} = 10{,}47 \text{ persona-mes}$$

El esfuerzo dedicado al proyecto será, según la estimación siguiendo el método de Albretch, de 10,47 persona-mes. La estimación es acorde a la planificación realizada.

2.5. Presupuesto económico

Antes de comenzar el proyecto es necesario estimar los costes de desarrollo. Para la estimación del presupuesto se tendrán en cuenta las herramientas hardware y software utilizadas, atendiendo a los factores de impacto correspondientes a la duración del proyecto, junto con los gastos derivados del personal, atendiendo para ello a la estimación de horas llevada a cabo en la planificación y al tipo de rol correspondiente a cada tarea.

El coste del proyecto se desglosa en cuatro categorías: hardware (Tabla 2.4), servicios (Tabla 2.5), licencias de software (Tabla 2.6) y costes de personal (Tabla 2.7).

	Coste (€)	**Uso (%)**	**Total (€)**
Ordenador portátil	1200,00	20	240,00
Monitor	200,00	20	40,00

Tabla 2.4: Costes de hardware

Para el desarrollo del proyecto se hará uso de una máquina virtual del Departamento de Informática de 8GB de RAM, un procesador dedicado de dos núcleos y 50GB de almacenamiento HDD. Para estimar su coste se utilizarán los precios de uso de una máquina virtual de Digital Ocean con unas especificaciones similares[2].

	Coste	**Uso (%)**	**Duración**	**Total (€)**
Conexión a Internet	45,00 €/mes	10	10 meses	45,0
Máquina Virtual	20 €/mes	100	10 meses	200,00

Tabla 2.5: Costes de servicios

[2]Véase https://www.digitalocean.com/pricing/

	Licencia	Coste (€)	Uso (%)	Total (€)
PyCharm	GPLv2a	0	100	0
FileZilla	GPLv2	0	100	0
Texmaker	GPLv2	0	100	0
HeidiSQL	GPLv2	0	100	0
PuTTy	GPLv2	0	100	0
MariaDB	GPLv2	0	100	0
WinSCP	GPLv2	0	100	0
Notion	GPLv2	0	100	0

Tabla 2.6: Costes de licencias software

Para la estimación del coste de personal se deberá desglosar el coste en función del rol de la tarea. Por ello se establece un salario bruto de 25.000€ para el rol de desarrollador backend junior, 20.000€ para el rol de desarrollador de aplicaciones móviles junior y 28.000€ para el rol de analista. Estos salarios son anuales (equivalente a 1.735 horas de trabajo).

El presente proyecto tiene una planificación de 300 horas (equivalente a un trabajo de fin de grado). No es complicado calcular el coste en términos de recursos humanos que va a tener el proyecto (véase el desglose de la nómina en la Tabla 2.7)

	Coste (€/hora)	Duración (horas)	Total (€)
Desarrollador backend junior	14,40	150	2.160,00
Desarrollador móvil junior	11,53	100	1.152,50
Analista	16,14	50	807,00

Tabla 2.7: Costes estimados de personal

A partir del estudio previo de costes estimados se deduce que el presupuesto final del proyecto es de *4.664,50 €*, de los cuales el 88,71 % (*4.119,5 €*) se destina al pago de nóminas (véase la Tabla 2.8). Esto ocurre en la mayoría de desarrollos de software que se desarrollan en la actualidad donde el mayor coste de los proyectos es atribuible al pago de nóminas.

	Total (€)	**% del total**
Hardware	280,00	6,02
Servicios	245,00	5,27
Licencias software	0,00	0,00
Nóminas	4.119,5	88,71
	4.664,50	100

TABLA 2.8: DESGLOSE TOTAL DE COSTES ESTIMADOS

2.6. Balance final del proyecto

En este apartado se va a comentar lo que ha supuesto realmente, una vez finalizado, el esfuerzo de desarrollo del proyecto.

La planificación original suponía que la cuarta iteración acabaría el 31 de Diciembre. Sin embargo esta iteración se retrasó **28 días** y finalizó el 28 de Enero. Además, la quinta iteración no se ha llevado a cabo y el proyecto no presenta ningún apartado de predicción. También hizo falta realizar una segunda revisión general en el mes de Mayo, que supuso 30 horas de trabajo no planificado.

Para obtener el coste económico real que ha supuesto el desarrollo del proyecto, se ha de multiplicar por 1,1 el presupuesto planificado en salarios, pues las 30 horas de esfuerzo no planificado equivalen al 10 % del tiempo planificado. Esto supone que el presupuesto real del proyecto sea de 5.130,95 euros.

En la figura 2.4 se incluye una tabla con los tiempos reales empleados en cada iteración, donde la quinta iteración no consiste en el desarrollo de un apartado de predicción sino en una segunda revisión consistente en incluir las mejoras sugeridas, portar los sistemas a máquinas virtuales externas a la universidad, etc.

	Nº de días	Inicio	Final
ITERACIÓN I	**33**	**08/07/2020**	**09/08/2020**
Análisis	3	08/07/2020	10/07/2020
Diseño	3	11/07/2020	13/07/2020
Implementación	22	14/07/2020	04/08/2020
Pruebas	3	05/08/2020	07/08/2020
Documentación	2	08/08/2020	09/08/2020
ITERACIÓN II	**47**	**10/08/2020**	**25/09/2020**
Análisis	3	10/08/2020	12/08/2020
Diseño	3	13/08/2020	15/08/2020
Implementación	36	16/08/2020	20/09/2020
Pruebas	3	21/09/2020	23/09/2020
Documentación	2	24/09/2020	25/09/2020
ITERACIÓN III	**39**	**17/11/2020**	**25/12/2020**
Análisis	2	17/11/2020	18/11/2020
Diseño	2	19/11/2020	20/11/2020
Implementación	26	21/11/2020	16/12/2020
Pruebas	6	17/12/2020	22/12/2020
Documentación	3	23/12/2020	25/12/2020
ITERACIÓN IV	**27**	**26/12/2020**	**28/01/2021**
Análisis	6	26/12/2020	31/12/2020
Diseño	5	08/01/1900	12/01/2021
Implementación	5	13/01/2021	17/01/2021
Pruebas	3	18/01/2021	20/01/2021
Documentación	8	21/01/2021	28/01/2021
ITERACIÓN V	**30**	**01/05/2021**	**07/05/2021**
Análisis	7	08/12/2021	12/01/2021
Diseño	8	13/01/2021	16/01/2021
Implementación	4	17/01/2021	20/05/2021
Pruebas	5	21/05/2021	25/05/2021
Documentación	6	26/05/2021	31/05/2021

Figura 2.4: Tiempo dedicado al proyecto

Capítulo 3

Estado del arte

Existen en la actualidad tres aplicaciones móviles líderes en el nicho de mercado que es el ahorro de combustible en vehículos particulares. En este apartado se estudiarán las tres y se hará un análisis comparativo entre ellas.

- Gasolineras España [1] (disponible para iOS y Android)
- Precio de Gasolina al instante [2] (disponible para Android)
- GasAll [3] (diponible para iOS y Android)

Existen, al menos, otras 10 aplicaciones que intentan resolver este mismo problema, pero con un éxito muy limitado debido a diversos factores: interfaz poco intuitiva, precios desactualizados, lentitud al cargar la aplicación y exceso de publicidad entre otros. Es por ello que este estado del arte se limita a estudiar las aplicaciones líderes, y cuyo funcionamiento es adecuado, con el objetivo de comparar mi aplicación con aquellas que son exitosas.

Para la plataforma Android, hay tres grandes líderes, que son las aplicaciones que se han mencionado. Por otro lado, para la plataforma iOS hay una única aplicación líder, es es GasAll y que cuenta con versión para Android e iOS. Esta cuenta con más de 8.500 valoraciones, muy lejos de la siguiente app, que cuenta con 252 y una valoración media muy inferior.

Todas las aplicaciones descritas están programadas en lenguajes nativos. Esto significa que las bases de código son completamente distintas para una plataforma que para otra en caso de tener versión tanto en iOS como en Android, lo cual supone un esfuerzo extra para el equipo de desarrollo que con Flutter se elimina.

---

[1] Véase https://play.google.com/store/apps/details?id=com.mobialia.gas.spain

[2] https://play.google.com/store/apps/details?id=de.mwwebwork.benzinpreisblitz

[3] https://play.google.com/store/apps/details?id=com.gasall

Tras analizar las aplicaciones disponibles para iOS, se ha detectado que hay una aplicación líder de forma indiscutible: *GasAll*. Esta cuenta con más de 8.500 (ocho mil quinientas) valoraciones, mientras que la siguiente más valorada cuenta con 252.

3.1. Gasolineras España

Gasolineras España, consta de más de medio millón de descargas. Actualiza los precios cada hora y tiene información sobre todas las gasolineras de España. Contiene anuncios y está traducida al Catalán y Gallego. Tiene más de 9.000 comentarios o reseñas. Está desarrollada por una *startup* española llamada *mobialia* [4] fundada en 2010 que se dedica al desarrollo de aplicaciones móviles desde 2010 que tiene, a día de hoy, otras 6 aplicaciones en el play store, todas ellas exitosas. También cuenta con versión en iOS, con idénticas funcionalidades e interfaces.

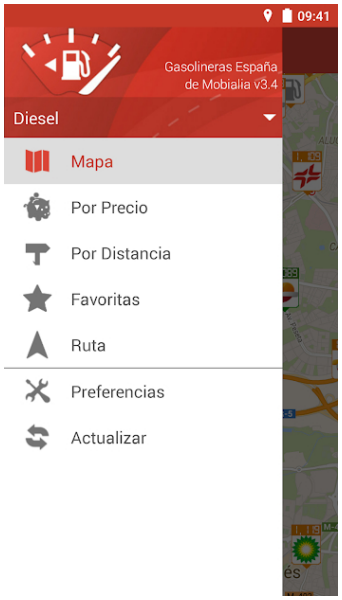


FIGURA 3.1: MENÚ DE OPCIONES

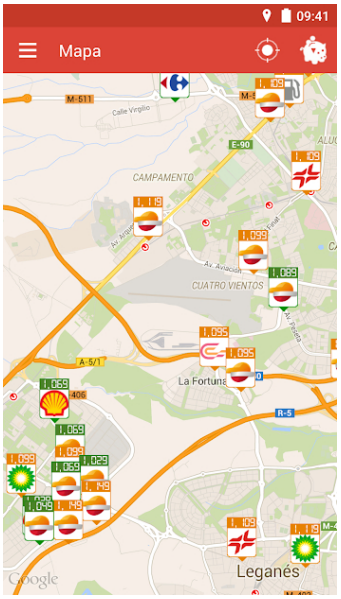


FIGURA 3.2: MAPA CON PRECIOS DE GASOLINERAS

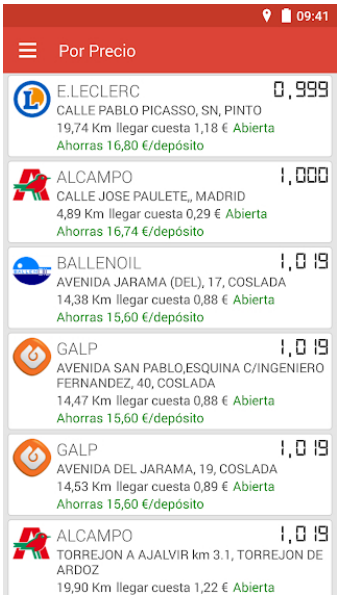


FIGURA 3.3: GASOLINERAS ORDENADAS POR PRECIO

En la figura 3.1 se puede observar el menú de opciones del que dispone la aplicación. Permite seleccionar la opción de visualizar las gasolineras en un mapa, en forma de lista ordenadas por precio y por distancia. También da la opción de ver tus gasolineras favoritas y hacer una ruta desde tu ubicación a alguna de las gasolineras. Por último tiene una opción para buscar nuevas actualizaciones y para modificar las preferencias. En la figura 3.2 se puede visualizar un mapa

[4] https://www.mobialia.com/

con los iconos y precios de las gasolineras en una zona de Madrid. Por último, en la figura 3.3 se puede ver un listado de gasolineras en un radio de 5km ordenadas por precio de menor a mayor.

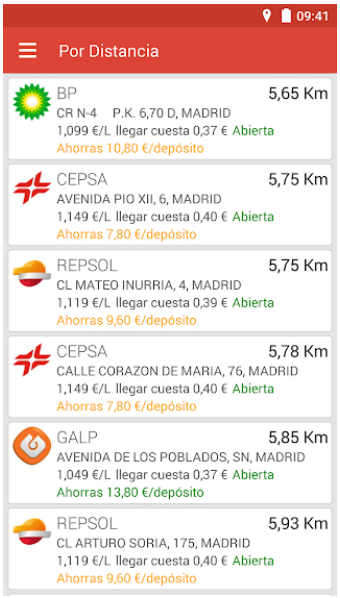


Figura 3.4: Gasolineras ordenados por distancia

Figura 3.5: Información sobre una gasolinera

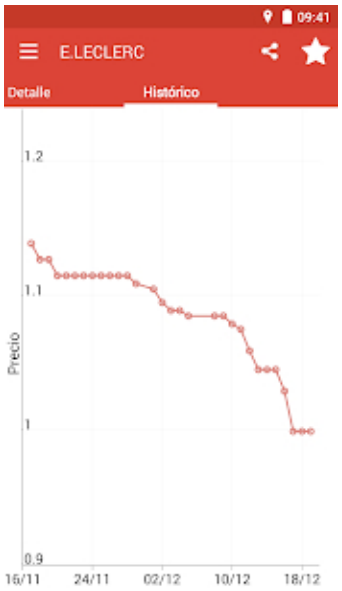


Figura 3.6: Histórico de precios de una gasolinera

En la figura 3.4 se puede ver un listado de gasolineras ordenadas por distancia. En la figura 3.4 se observa la pantalla que aparece al seleccionar una gasolinera en la cual aparecen los horarios, la dirección y el precio. También da la opción de navegar hasta esa ubicación usando google maps. Por último en la figura 3.6 se puede ver el histórico de los precios de los últimos días de una gasolinera.

3.2. Precio de Gasolina al instante

Precio de Gasolina al instante cuenta con una funcionalidad similar a la primera, pero además incluye gasolineras de Alemania, Austria y Francia y cuenta con más de un millón de descargas. Al igual que en España, estos países tienen un portal donde exponen esta información. Esta aplicación está desarrollada por la empresa alemana *MW WebWork* [5], especialistas en desarrollo móvil y web.

[5] https://www.mw-webwork.de/

FIGURA 3.7: GASOLINERAS ORDENADOS POR PRECIO

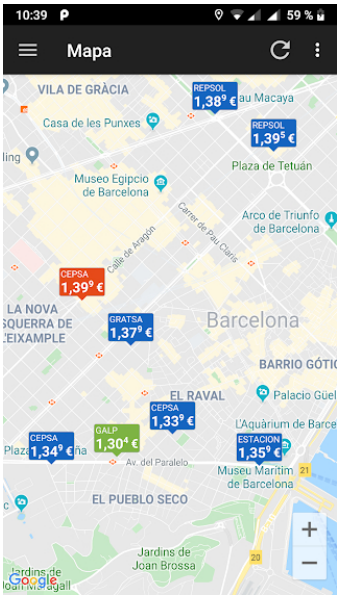


FIGURA 3.8: MAPA CON PRECIOS DE GASOLINERAS

FIGURA 3.9: DETALLES DE UNA GASOLINERA

En las figuras 3.7, 3.8 y 3.9 se pueden ver un listado de gasolineras en un radio de 5km, un mapa con las gasolineras y por último una pantalla mostrando los detalles de una gasolinera incluyendo horarios, dirección y precios. Esta última pantalla permite guardar una gasolinera en favoritos o generar una ruta hacia esa gasolinera usando google maps.

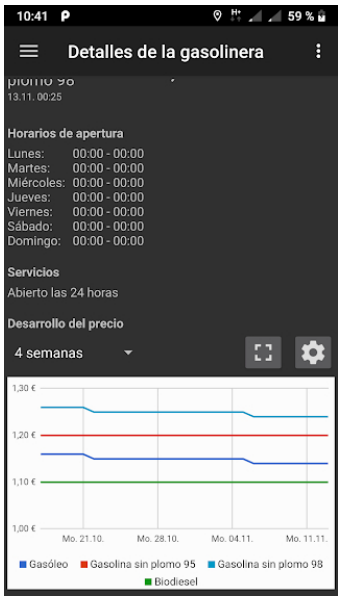


FIGURA 3.10: HISTÓRICO DE PRECIOS

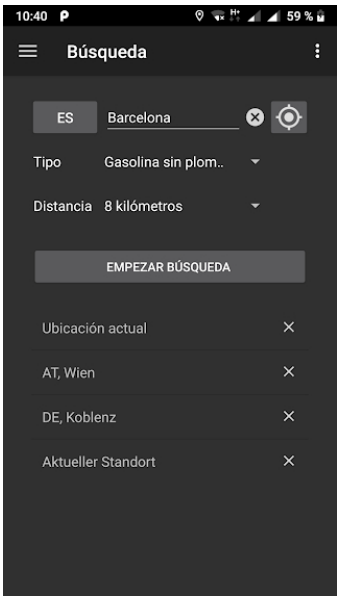


FIGURA 3.11: MENÚ DE OPCIONES DE BÚSQUEDA

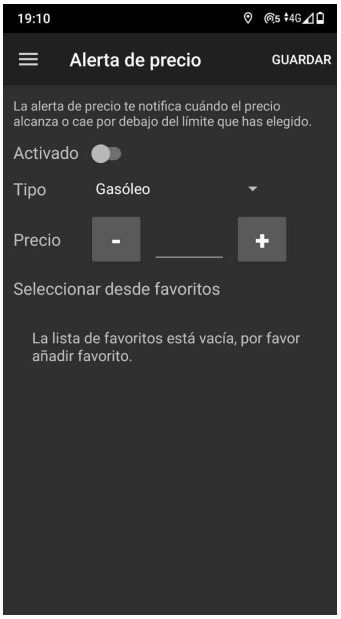


FIGURA 3.12: MENÚ DE CONFIGURACIÓN ALERTAS

3.3. GasAll

GasAll es una aplicación móvil con más de 100.000 descargas y 4.000 comentarios o reseñas. Está desarrollado por la empresa española *auroralabs* [6] con oficinas en Santander, Barcelona y Sevilla. Esta empresa se dedica al desarrollo de aplicaciones móviles y ha desarrollado aplicaciones para *Decathlon* y *Mapfre* entre otros.

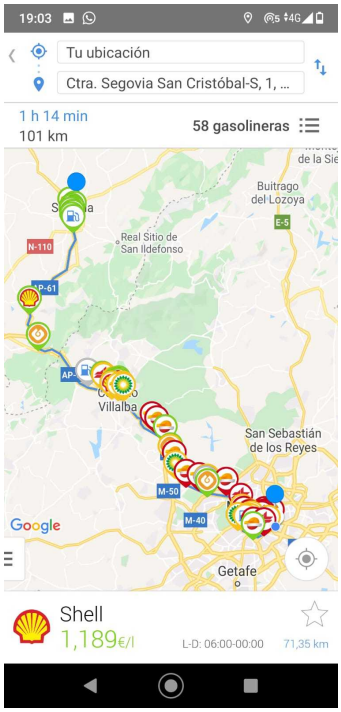


Figura 3.13: Gasolineras a lo largo de una ruta

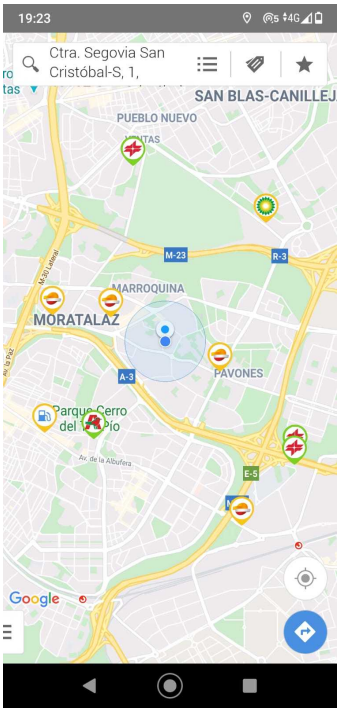


Figura 3.14: Gasolineras cercanas a tu ubicación

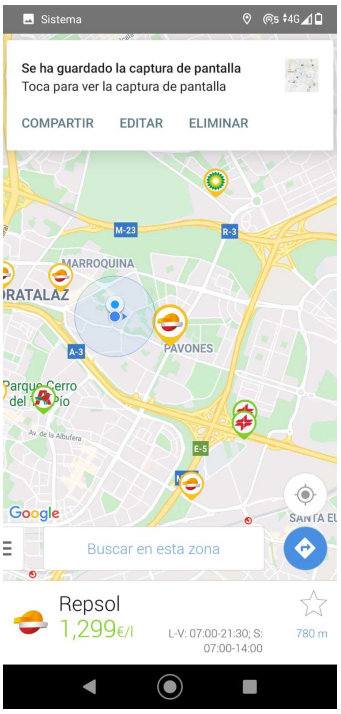


Figura 3.15: Información tras seleccionar una gasolinera

En la figura 3.13 se puede ver una ruta generada desde la ubicación del usuario hasta la gasolinera final. A diferencia del resto de aplicaciones, GasAll genera la ruta en la propia aplicación sin hacer uso de google maps como plataforma externa. En la figura 3.14 se muestra la ubicación del usuario en un mapa y las gasolineras de su alrededor. En la imagen 3.15 se muestra la misma información anterior añadiendo la información de una gasolinera que ha sido seleccionada. Al seleccionar la gasolinera se muestra el horario de la misma, el precio y la opción de guardar esa gasolinera en favoritos.

[6] https://www.auroralabs.es/

FIGURA 3.16: INFORMACIÓN UNA VEZ SELECCIONADA LA GASOLINERA

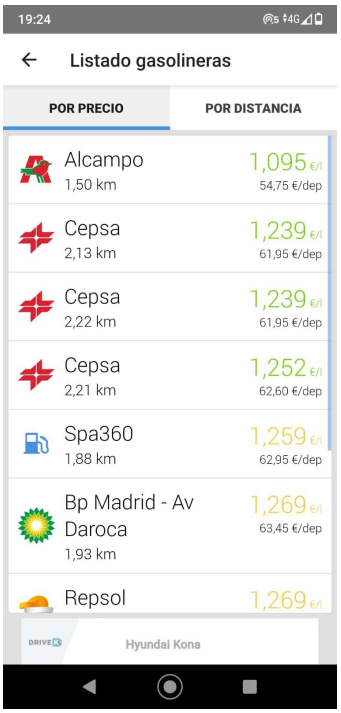


FIGURA 3.17: GASOLINERAS ORDENADAS POR PRECIO

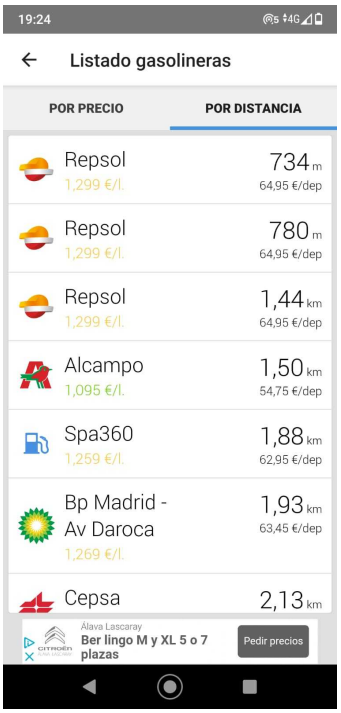


FIGURA 3.18: GASOLINERAS ORDENADAS POR DISTANCIA

En la figura 3.16 aparece una imagen similar a la anterior pero con información extra y que además da la opción de ver el histórico de precios. En las imágenes 3.17 y 3.18 se pueden observar un listado de gasolineras en un radio predeterminado ordenadas por precio y por distancia respectivamente.

Esta aplicación dispone de versiones tanto para la plataforma Android como para la plataforma iOS. Aquí se han mostrado solo las interfaces Android porque las interfaces iOS son muy similares. GasAll es aplicación más exitosa del ecosistema iOS en este sector, aunque la tercera del ecosistema Android. Es una aplicación que primero apareció en Android y un año más tarde para iOS. Si bien es cierto que no es la más popular en todos los ecosistemas, sí que tiene una interfaz muy lograda y intuitiva.

3.4. Comparativa

En este apartado se va a mostrar una tabla comparativa con distintas características. Como la aplicación GasAll tiene las mismas características tanto para Android como iOS solo se muestra una única columna para la misma.

Característica	Gasolineras España	Precio de gasolina al instante	GasAll
Lanzamiento Android	26/10/2010	16/08/2014	01/07/2014
S.O.	Android/iOS	Android	Android/iOS
Tamaño Android	9,8MB	19,7MB	21,4 MB
Tamaño iOS	33,9MB	N.A	15,2MB
Valoración Android	3,7/5	4,7/5	4,6/5
Valoración iOS	4,7/5	SÍ	4,5/5
Nº Descargas Android	500.000	1.000.000	100.000
Versión gratuita	SÍ	SÍ	SÍ
Anuncios	SÍ	SÍ	SÍ
Precio versión PRO	N.A.	1,49€/año	0,99€
Ordenar por precio	SÍ	SÍ	SÍ
Ordenar por cercanía	SÍ	SÍ	SÍ
Información de horarios	SÍ	SÍ	SÍ
Gasolineras en un ruta	NO	NO	SÍ
Admite clustering	NO	NO	NO
Requiere permiso GPS	SÍ	SÍ	SÍ
Muestra tu ubicación	SÍ	SÍ	SÍ
Genera ruta a destino	SÍ	SÍ	SÍ
Guardar favoritos	SÍ	SÍ	SÍ
Filtrar por marcas	SÍ	NO	NO

Tabla 3.1: Tabla comparativa

Desde mi punto de vista particular la aplicación más lograda es GasAll. Tiene una interfaz muy cuidada y cuenta con más opciones que el resto. Aunque gasolineras España es la que más descargas tiene, tiene una valoración media considerablemente más baja que su competencia, debido a que su interfaz se ha estancado y no ha recibido actualizaciones en los últimos años. Sin embargo es cierto que cuenta con información de gasolineras de más países además de España, al contrario que su competencia, cuyo alcance se circunscribe únicamente a España.

Capítulo 4

Fuentes de datos

En este capítulo se hace un estudio de la estructura de los datos proporcionados por el Ministerio para la Transición Ecológica y el Reto demográfico desde el portal geogasolineras.

4.1. Origen de los datos

Los datos con los que se trabaja en este proyecto se obtienen desde desde la url proporcionada por el portal geogasolineras para la descarga de ficheros[1]. Este punto de descarga proporciona un fichero Excel con extensión xls que es actualizado cada hora. Todas las estaciones de servicio en España están obligadas a informar con al menos 12 horas de antelación los cambios de precio, por lo que los datos proporcionados están siempre actualizados.

El fichero Excel proporcionado siempre tiene el mismo nombre: *preciosEESS_es.xls*, y su estructura apenas varía con el tiempo. Además se actualiza cada hora con nueva información. Se han analizado más de 3.000 ficheros desde 2010 hasta ahora y la estructura de estos apenas ha sufrido modificaciones. Durante este tiempo la url desde la que se descarga el fichero también se ha mantenido constante. Por otro lado, cada cierto tiempo el portal que proporciona los datos genera una nueva columna con información extra, pero sin modificar la estructura de datos ya existente. Así en 2019 apareció una nueva columna que especificaba los horarios en los cuales las gasolineras tenían autoservicio. Debido a que existen gasolineras que dependiendo del día y la hora ofrecen atención o no, el portal incluyó esta columna para aclarar esta información. No sería de extrañar que en el futuro aparezcan nuevas columnas.

[1] http://geoportalgasolineras.es/resources/files/preciosEESS_es.xls

4.2. Estructura de los datos

En esta sección se especifica cómo es la estructura de los datos que se descargan a través de tablas e imágenes. Las tres primeras filas del fichero descargado proporcionan información sobre la fecha en la que se generó el fichero, una descripción del fichero y el significado de las siglas que se van a utilizar. En la cuarta fila hay 30 columnas, cada una de ellas con un título. A partir de la quinta fila se encuentran los datos y cada línea corresponde a una gasolinera distinta.

En la tabla 4.1 se muestra la información presente en las tres primeras filas de estos documentos.

Fichero Excel - cabecera	
Fecha	DD-MM-YYYY hh:mm
Descripción	Este archivo se genera una vez cada hora, con los precios en ese momento, y sobrescribe al de la hora anterior, que no se conserva.
Siglas	**Margen** - D: Derecho, I: Izquierdo, N: No aplica **Tipo venta** - P: Venta al público en general, R: Venta restringida a socios o cooperativistas **Rem.** - OM: Datos procedentes del operador mayorista, dm: Datos procedentes del distribuidor minorista **Tipo servicio** - P: Servicio asistido, A: Autoservicio con presencia de personal en la instalación, D: Autoservicio sin presencia de personal en la instalación

TABLA 4.1: ESTRUCTURA DE LA CABECERA DEL FICHERO EXCEL

En la figura 4.1 se muestra la estructura y apariencia del fichero Excel que se descarga. En esta figura se han eliminado trece columnas correspondientes a varios tipos de carburante para facilitar la visualización por parte del lector.

Más adelante, en la tabla 4.2 se listan las columnas que tiene el fichero Excel desde la cuarta fila en adelante. Se especifican también los posibles valores que pueden tomar esas columnas y se muestra un ejemplo con datos reales.

Municipio	Localidad	Código postal	Dirección	Margen	Longitud	Latitud	Precio gasolina 95 E5	Precio gasolina 98 E5	Precio gasóleo A	Precio gasóleo Premium	Precio gases licuados del petróleo	Rótulo	Tipo venta	Rem.	Horario	Tipo servicio
ABENGIBRE	ABENGIBRE	02250	AVENIDA CASTILLA LA MANCHA, 26	D	-1,539167	39,211417	1,269		1,139			Nº 10.935	P	dm	L-D: 07:00-22:00	L-D: 07:00-22:00 (A)
ALATOZ	ALATOZ	02152	CR CM-332, 46,4	I	-1,346083	39,100389	1,309	1,320	1,200	1,250		REPSOL	P	dm	L-D: 7:00-23:00	L-D: 7:00-23:00 (A)
ALBACETE	ALBACETE	02001	CALLE PRINCIPE DE ASTURIAS (POLÍGONO DE ROMICA), 5	I	-1,832000	39,054694	1,339	1,481	1,239	1,329		BP ROMICA	P	OM	L-D: 06:00-21:30	L-D: 06:00-21:30 (A)
ALBACETE	ALBACETE	02001	CALLE FEDERICO GARCIA LORCA, 1	D	-1,849833	39,000861	1,209		1,079			PLENOIL	P	dm	L-D: 24H	L-D: 24H (D)
ALBACETE	ALBACETE	02001	AVENIDA 1º DE MAYO, S/N	N	-1,868500	38,985667	1,319	1,399	1,199	1,229		CARREFOUR	P	dm	L-S: 08:00-22:00; D: 09:00-21:00	L-S: 08:00-22:00 (A); D: 09:00-21:00 (A)
ALBACETE	ALBACETE	02005	PASEO CUBA (LA), 36	N	-1,859917	39,005083	1,369	1,469	1,239	1,299	0,819	CEPSA	P	dm	L-D: 06:00-23:00	L-D: 06:00-23:00 (P)
ALBACETE	ALBACETE	02005	CL PASEO DE LA CUBA, 15	D	-1,854556	38,999722	1,349	1,459	1,239	1,299	0,799	REPSOL	P	OM	L-D: 06:00-22:00	L-D: 06:00-22:00 (A)
ALBACETE	ALBACETE	02005	AVENIDA MENÉNDEZ PIDAL, 58	N	-1,864917	39,003333	1,359	1,469	1,229	1,299		TAMOS	P	dm	L-D: 24H	L-D: 24H (A)
ALBACETE	ALBACETE	02005	AVENIDA ESCRITOR RODRIGO RUBIO, 3	D	-1,885361	39,006889	1,209		1,099	1,159		A&A	P	dm	L-S: 09:00-21:30	Sin datos
ALBACETE	ALBACETE	02002	CALLE HERMANOS FALCÓ, 2	D	-1,849028	38,989250	1,359	1,469	1,229	1,309		CEPSA	P	dm	L-D: 06:30-22:30	Sin datos
ALBACETE	ALBACETE	02006	CALLE ALCALDE CONANGLA (C.C. EROSKI), S/N	N	-1,847361	38,988972	1,209	1,289	1,079	1,099		FAMILY ENERGY	P	dm	L-S: 07:00-23:00; D: 08:00-23:00	L-S: 07:00-23:00 (A); D: 08:00-23:00 (A)
ALBACETE	ALBACETE	02006	CARRETERA JAEN KM. 2	D	-1,871722	38,994667	1,369	1,469	1,239	1,299		REPSOL	P	dm	L-D: 24H	L-D: 24H (A)
ALBACETE	ALBACETE	02006	CR N-322, 337,8	I	-2,031861	38,925833	1,359	1,479	1,239	1,309		REPSOL	P	dm	L-D: 06:30-22:00	L-D: 06:30-22:00 (A)
ALBACETE	ALBACETE	02006	AVDA. DE ESPAÑA (PRÓX. A C/ SAN JUAN), S/N	N	-1,853306	38,982167	1,269		1,139			INPEALSA	P	dm	L-D: 07:00-23:00	L-D: 07:00-23:00 (P)
ALBACETE	ALBACETE	02006	AVDA. DE LOS TOREROS, S/N	N	-1,869889	38,998222	1,269		1,139	1,189		INPEALSA	P	dm	L-D: 07:00-23:00	L-D: 07:00-23:00 (P)
ALBACETE	ALBACETE	02006	CARRETERA NACIONAL 322 PTO KM 347 KM. 347	D	-1,935500	38,965639			1,129			P347	P	dm	L-S: 07:00-22:00	L-S: 07:00-22:00 (A)
ALBACETE	ALBACETE	02006	AVENIDA 1º DE MAYO, S/N	N	-1,866806	38,985194			1,149			PETROCAMP	P	dm	L-D: 07:00-23:00	L-D: 07:00-23:00 (A)
ALBACETE	ALBACETE	02006	CALLE CTRA. DE JAEN, 79	I	-1,883694	38,988111	1,369	1,489	1,239	1,309		REPSOL	P	dm	L-D: 06:00-23:00	L-D: 06:00-23:00 (A)
ALBACETE	ALBACETE	02006	CR N-301, 244	I	-1,892972	39,027778	1,339	1,449	1,239	1,289		REPSOL	P	OM	L-D: 24H	L-D: 24H (A)
ALBACETE	ALBACETE	02006	CARRETERA N-322 KM. 359	D	-1,845083	39,033139	1,319	1,459	1,229	1,279		REPSOL	P	OM	L-V: 07:00-21:00; S-D: 09:00-14:00	Sin datos
ALBACETE	ALBACETE	02006	PG CAMPOLLANO-P. 2-CT, S.N.	D	-1,874972	39,012583	1,339	1,449	1,239	1,289		REPSOL	P	OM	L-V: 06:00-22:00; S: 07:30-15:30; D: 09:00-15:00	L-V: 06:00-22:00 (A); S: 07:30-15:30 (A); D:
ALBACETE	ALBACETE	02006	CALLE ALMANSA, 2	I	-1,847083	38,984139	1,339	1,471	1,229	1,295		CEPSA	P	dm	L-D: 06:00-23:00	L-D: 06:00-23:00 (A)
ALBACETE	ALBACETE	02099	CARRETERA NACIONAL 301 KM. 256	D	-1,831639	38,929389	1,279		1,129			INPEALSA	P	dm	L-V: 06:00-22:00; S: 06:00-14:00; D: 09:00-21:00	L-V: 06:00-22:00 (P); S: 06:00-14:00 (P); D:
ALBACETE	ALBACETE	02004	CARRETERA N-322 KM. 349	D	-1,919778	38,970167	1,339	1,409	1,249	1,319		CEPSA	P	dm	L-D: 24H	L-D: 24H (A)
ALBACETE	SANTA ANA	02328	AVENIDA PRINCIPE, 2	D	-1,994028	38,900944	1,309	1,419	1,199	1,249		REPSOL	P	OM	L-D: 08:00-16:00	L-D: 08:00-16:00 (A)
ALBACETE	SALOBRAL (EL)	02140	CM. 3203 km 14,1	D	-1,920500	38,867556	1,285		1,160	1,200		LOZANO	P	dm	L-S: 07:00-21:30; D: 09:00-14:00	L-S: 07:00-21:30 (P); D: 09:00-14:00 (P)
ALBACETE	ALBACETE	02007	CALLE C/ AUTOVIA C/V AVD. SEXTA, S/N	D	-1,888083	39,029583						NATURGY ES GNV	P	dm	L-D: 24H	L-D: 24H (A)
ALBACETE	ALBACETE	02007	AVENIDA POLIGONO CAMPOLLANO AVENIDA 0, 67	N	-1,886556	39,021583	1,269		1,129			VALERO 24H	P	dm	L-D: 24H	L-D: 24H (A)

Figura 4.1: Imagen Fichero Excel

Columna	Posibles valores	Ejemplo
Provincia	Provincias de España	ALBACETE
Municipio	Municipios de España	ABENGIBRE
Localidad	Localidades de España	ABENGIBRE
Código postal	Número de 5 dígitos	02250
Dirección	Cualquier dirección	AVENIDA LA MANCHA, 26
Margen	D,I,N	D
Longitud	Longitud	-1,539167
Latitud	Latitud	39,211417
Precio gasolina 95 E5	Número con tres decimales	1,329
Precio gasolina 95 E10	Número con tres decimales	1,400
Precio gasolina 95 E5 Premium	Número con tres decimales	1,375
Precio gasolina 98 E5	Número con tres decimales	1,298
Precio gasolina 98 E10	Número con tres decimales	1,278
Precio gasóleo A	Número con tres decimales	1,190
Precio gasóleo Premium	Número con tres decimales	1,208
Precio gasóleo B	Número con tres decimales	1,147
Precio gasóleo C	Número con tres decimales	1,357
Precio bioetanol	Número con tres decimales	1,243
% bioalcohol	Número con dos decimales	4,56
Precio biodiésel	Número con tres decimales	0,978
% éster metílico	Número con dos decimales	12,50
Precio gases licuados del petróleo	Número con tres decimales	0,879
Precio gas natural comprimido	Número con tres decimales	0,765
Precio gas natural licuado	Número con tres decimales	0,822
Precio hidrógeno	Número con tres decimales	0,913
Rótulo	Cualquier nombre	REPSOL
Tipo venta	P,R	P
Rem.	O.M,d.m	dm
Horario	Cualquier horario	L-D: 7:00-23:00
Tipo servicio	P,A,D	L-D: 7:00-23:00 (A))

TABLA 4.2: ESTRUCTURA CONTENIDO DEL FICHERO EXCEL

En el capítulo de implementación (capítulo 7) se especifican las transformaciones de datos realizadas para poder ingestar estos datos de forma estructurada en la base de datos.

Capítulo 5

Análisis

Este capítulo contiene toda la información recogida y especificada durante la fase de análisis, fase centrada en obtener un conocimiento más preciso de los requerimientos del cliente. A continuación se especifican las características del sistema, los actores que interactúan con la aplicación empresarial y las tareas que desempeñará cada uno con el sistema (requisitos de usuario). Además, se procede a derivar los requisitos funcionales a partir de los de usuario y se detallan los requisitos no funcionales. Por último se documentan los datos con los que operará el sistema (requisitos de información).

5.1. Características del sistema

Las características de un sistema son las funcionalidades básicas y principales con las que éste debe contar para poder cumplir con los objetivos del proyecto. A continuación se enumeran las características que ha de cumplir este proyecto:

- **CAR-1.0 - Descarga de ficheros:** descarga de los ficheros Excel desde el portal de ministerio cada hora.
- **CAR-1.1 - Limpieza de datos:** asegura de que no haya inconsistencia en los datos, añade datos que puedan faltar o elimina datos incorrectos.
- **CAR-1.2 - Inserción de datos:** inserta la información en una base de datos.

- **CAR-2.0 - Obtención de precios en tiempo real:** devuelve precios de gasolineras en tiempo real.
- **CAR-2.1 - Obtención histórico de precios:** devuelve un histórico de precios de un conjunto de gasolineras.

- **CAR-2.2 - Obtención histórico de gasolineras:** permite visualizar la información histórica de una
- **CAR-2.3 - Descarga de app móvil:** permite descargarse el apk Android de la app móvil.
- **CAR-2.4 - Visualización miembros proyecto:** permite visualizar a los miembros del proyecto.

- **CAR-3.0 - Visualización de gasolineras:** muestra un mapa con todas las gasolineras disponibles.
- **CAR-3.1 - Visualización de precios:** muestra los precios de todas las gasolineras disponibles
- **CAR-3.1 - Visualizar información de una gasolinera:** muestra la información de una gasolinera.

5.2. Árbol de características

En la figura 5.1 se incluye un diagrama con todas las características del proyecto que permite visualizar el alcance del sistema.

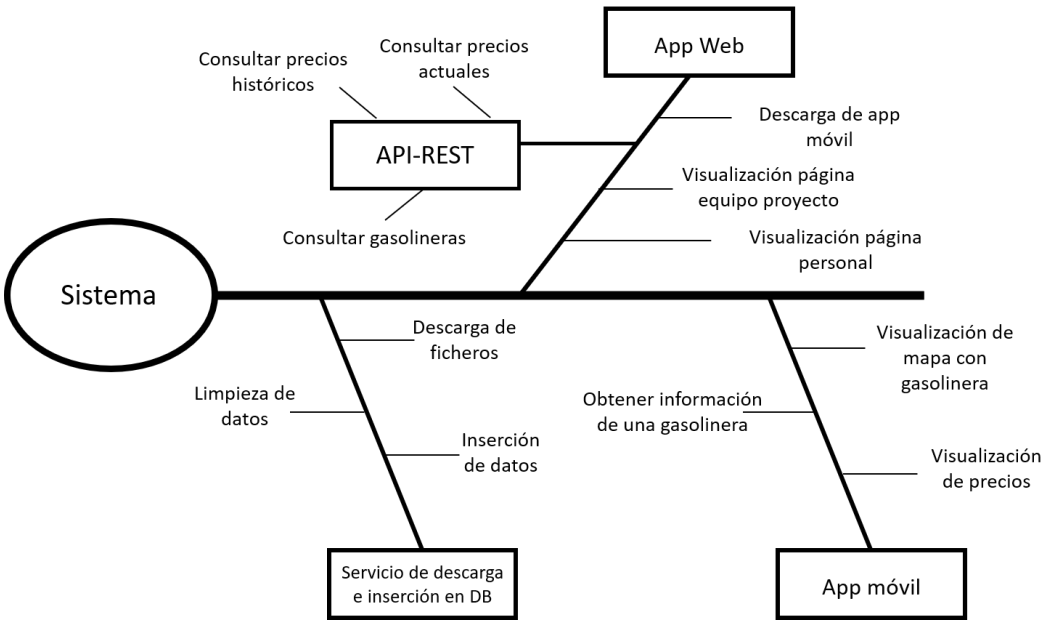


FIGURA 5.1: ÁRBOL DE CARACTERÍSTICAS

5.3. Identificación y descripción de los actores

En esta sección se describirán todos los actores (personas u otros sistemas externos) que interactúan con el sistema. La Figura 5.2 muestra un diagrama de actores con las relaciones existentes entre ellos.

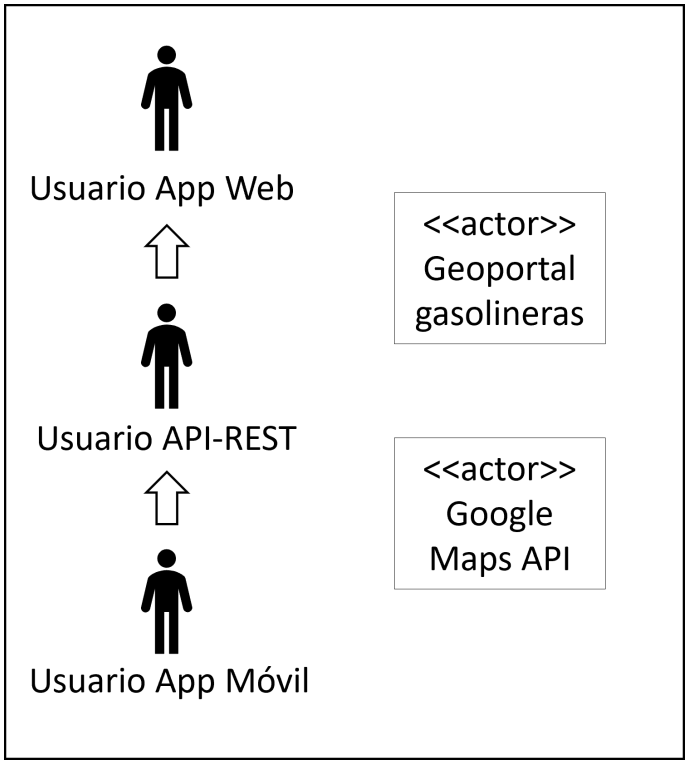


Figura 5.2: Diagrama de relación entre actores

La herencia entre usuarios se explica debido a que la aplicación móvil hace uso de la API-REST para obtener la información y por tanto los usuarios de la aplicación móvil están usando dicho servicio REST, aunque sea de forma transparente para ellos. Al mismo tiempo, el servicio REST es solo una parte de la aplicación web desarrollada, por lo que se puede distinguir entre usuarios de dicho servicio, y los usuarios de la aplicación web, que es un tipo de usuario más genérico. En todos los casos son usuarios sin identificar, ya que ninguno de los sistemas requiere de identificación para ser utilizado. Para los actores secundarios del sistema se ha utilizado la notación de Craig Larman [5].

En la tabla 5.1 se describen con detalle a los usuarios de los distintos sistemas.

ACTORES		
ID	**Nombre**	**Descripción**
A1	Usuario App Móvil	Este usuario describe al conjunto de usuarios que hace uso de la aplicación móvil. Estos usuarios podrán visualizar un mapa con las gasolineras y sus precios.
A2	Usuario API-REST	Este usuario describe al conjunto de usuarios que hace uso de la API-REST. Estos usuarios podrán consultar datos de precios y gasolineras a través del sistema.
A3	Usuario App Web	Este usuario describe al conjunto de usuarios que hace uso de la aplicación web, donde el servicio REST se aloja.
A4	Geportal gasolineras	Portal web del Ministerio de Transición Ecológica y el Reto Demográfico que pone a disposición del público los datos.
A5	Google Maps API	Servicio externo proporcionado por Google, utilizado para obtener un mapa sobre el que mostrar las gasolineras.

TABLA 5.1: ACTORES

A continuación en la Figura 5.3 se muestra el Diagrama de Contexto de la aplicación móvil, incluyendo los flujos de datos entrantes y salientes de la aplicación móvil.

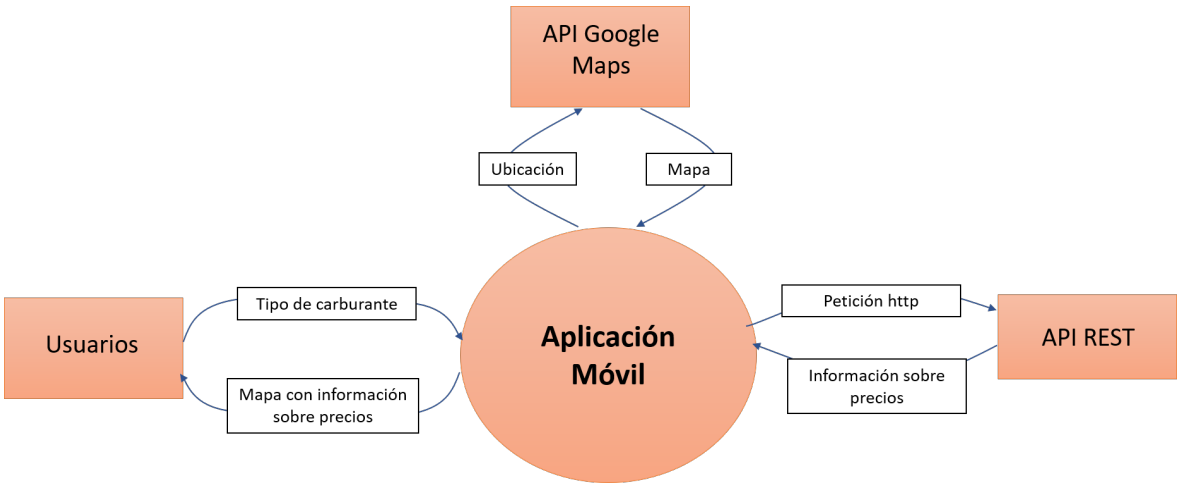


FIGURA 5.3: DIAGRAMA DE CONTEXTO APP MÓVIL

5.4. Requisitos de Usuario

Los Requisitos de Usuario son tareas que una clase de usuario podrá realizar con el sistema. Describen los requisitos que los usuarios esperan obtener mediante la aplicación.

En este apartado se procede a identificar y listar todos los requisitos de usuario que presentan los distintos sistemas. Para la especificación de los requisitos de usuario se utilizarán los Casos de Uso puesto que permiten determinar la interacción que acontece entre un actor externo y el sistema. También se mostrará el Diagrama de Casos de Uso que determina la relación entre los Requisitos de Usuario y la propia relación entre Casos de Uso. Por último se detallarán las especificaciones de Casos de Uso con el fin de aclarar y fijar las interacciones entre actores y los sistemas de cara a su posterior implementación.

Requisitos de Usuario	
Usuario App Móvil (A1)	
RU-A1-01	Buscar gasolineras
RU-A1-02	Mover mapa
RU-A1-03	Seleccionar gasolinera
RU-A1-04	Crear ruta
RU-A1-05	Ir a ubicación actual
Usuario API REST (A2)	
RU-A2-01	Consultar datos
Usuario App Web (A3)	
RU-A3-01	Descargar app móvil
RU-A3-02	Ver página de equipo
RU-A3-03	Acceder página personal
Sistema (A0)	
RU-A0-01	Descargar datos
RU-A0-02	Procesar datos
RU-A0-02	Guardar datos

Tabla 5.2: Requisitos de Usuario

Los requisitos de usuario asociados al actor A0 son aquellos requisitos que son iniciados por el propio sistema. Los procesos de descarga, procesado y guardado de datos no se asocian a ningún actor externo que los inicialice pues es el propio sistema quien se encarga de ejecutar estas funcionalidades de manera automática.

La Figura 5.4 muestra el Diagrama de Casos de Uso.

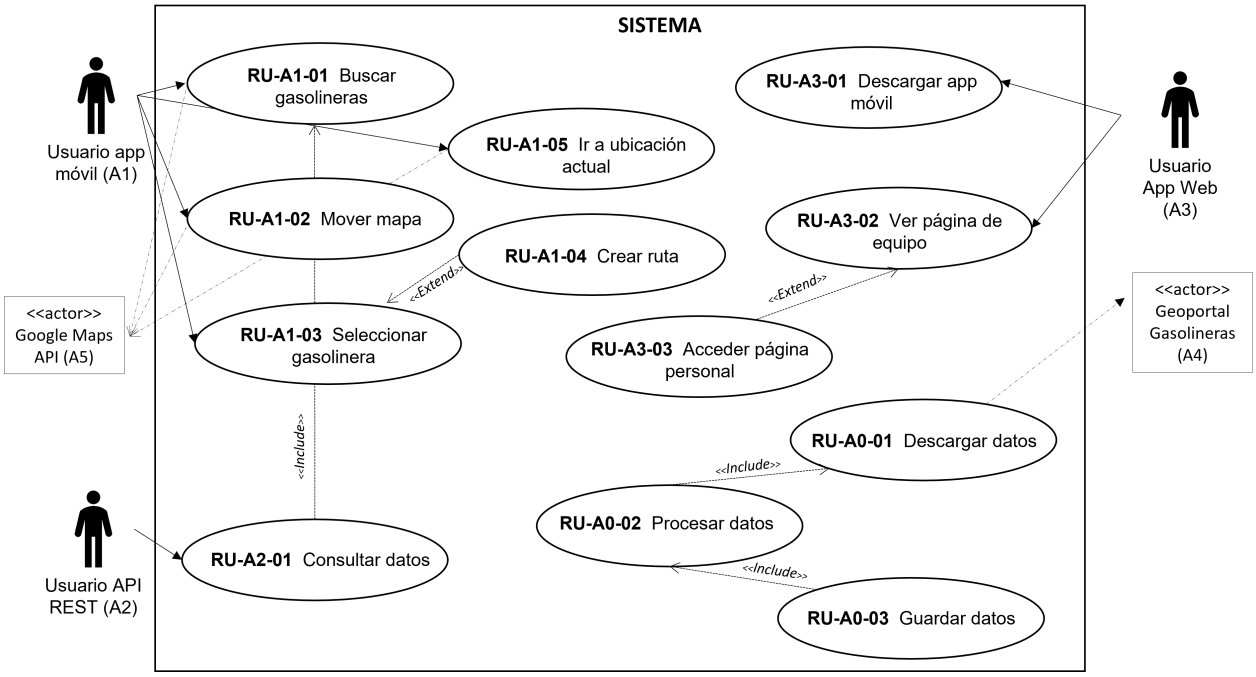


FIGURA 5.4: DIAGRAMA DE CASOS DE USO USUARIOS

La relación de dependencia *«include»* se utiliza para especificar que un caso de uso incluye a otro, y este no se podría ejecutar sin que lo haga el anterior. Por otro lado, la relación de dependencia *«extends»* se utiliza para especificar que un caso de uso extiende la funcionalidad de otro. En este último caso no es indispensable que la extensión se ejecute.

Se han utilizado líneas discontinuas para especificar aquellos casos en los que los actores secundarios A5 (Google Maps API) o A4 (Geoportal gasolineras) intervienen en la realización o ejecución de alguno de los casos de uso.

La forma de especificar los requisitos de usuario será mediante casos de uso. Un caso de uso es una secuencia de acciones que un sistema lleva a cabo y que da lugar a un resultado de valor observable para un actor o conjunto de actores particulares (alguien o algo fuera del sistema que interactúa con el sistema). En las tablas que se muestran a continuación se especifican los casos de uso presentados en el diagrama anterior.

Id y Nombre	RU-A1-01 Buscar gasolineras
Actor principal	Usuario App Móvil(A1)
Actores secundarios	Google Maps API (A5)
Descripción	El usuario de la app móvil desea buscar gasolineras que vendan un carburante en particular y visualizar sus precios.
Precondiciones	El usuario A1 ha seleccionado un carburante.
Postcondiciones	El usuario A1 visualiza un mapa con gasolineras y sus precios.
Flujo normal	1. El usuario pulsa el botón *buscar*. 2. La aplicación muestra una pantalla de espera. 3. La aplicación hace una llamada a la API-REST con el tipo de carburante previamente seleccionado. 4. La aplicación hace una llamada a la API de Google Maps. 5. La aplicación recibe una respuesta de la API de Google Maps (A5). 6. La aplicación recibe una respuesta de la API-REST 7. La aplicación muestra una nueva pantalla con un mapa posicionado sobre la ubicación del usuario y con iconos de gasolineras y sus precios donde corresponda.
Flujo alternativo	5. El sistema no recibe una respuesta de la API-REST 6. El sistema no recibe una respuesta de la API de google maps (A5).
Excepciones	3. Si no hay conexión a internet, las llamadas a las APIs no se llevarán a cabo y el sistema mostrará, hasta que se reanude la conexión, una pantalla de espera.
Prioridad	Alta

Tabla 5.3: RU-A1-01 Buscar gasolineras

Id y Nombre	RU-A1-02 Mover mapa
Actor principal	Usuario App Móvil (A1)
Actores secundarios	Google Maps API (A5)
Descripción	El usuario desea moverse por el mapa para ver el precio de las gasolineras en otras zonas.
Precondiciones	La aplicación ha cargado correctamente el mapa
Postcondiciones	N/A
Flujo normal	1. El usuario A1 utiliza la interfaz para desplazarse por el mapa. 2. La aplicación hace una llamada a la API de Google Maps (A5). 3. La aplicación recibe una respuesta de la API de Google Maps (A5). 4. La aplicación dibuja la nueva zona del mapa.
Flujo alternativo	3. El sistema no recibe una respuesta de la API de Google Maps (A5).
Excepciones	3. Si no hay conexión a internet, las llamadas a las APIs no se llevarán a cabo y el sistema mostrará, hasta que se reanude la conexión, una pantalla de espera.
Prioridad	Media

Tabla 5.4: RU-A1-02 Mover mapa

Id y Nombre	RU-A1-03 Seleccionar gasolinera
Actor principal	Usuario App Móvil (A1)
Actores secundarios	N/A
Descripción	El usuario A1 selecciona una gasolinera del mapa.
Precondiciones	La aplicación ha cargado correctamente el mapa
Postcondiciones	Se muestra la marca de la gasolinera y la opción de crear ruta hacia la misma.
Flujo normal	1. El usuario (A1) selecciona el icono de una gasolinera. 2. La aplicación muestra la marca de la gasolinera y muestra la opción de crear una ruta hasta esa gasolinera
Flujo alternativo	N/A
Excepciones	N/A
Prioridad	Media

Tabla 5.5: RU-A1-03 Seleccionar gasolinera

Id y Nombre	RU-A1-04 Crear ruta
Actor principal	Usuario App Móvil (A1)
Actores secundarios	Google Maps API (A5)
Descripción	El usuario selecciona generar una ruta a una gasolinera.
Precondiciones	Se ha seleccionado una gasolinera
Postcondiciones	Se ha abierto la aplicación de Google Maps con una ruta
Flujo normal	1. El usuario selecciona una gasolinera del mapa. 2. La aplicación muestra la opción de generar ruta hacia esa gasolinera. 3. El usuario selecciona la opción de generar ruta hacia esa gasolinera. 4. Se abre la aplicación de Google Maps. 5. Google Maps muestra las opciones habituales para desplazarse hasta una ubicación.
Flujo alternativo	4. Si no hay conexión se muestra un mensaje de error
Prioridad	Baja

Tabla 5.6: RU-A1-04 Crear ruta

Id y Nombre	RU-A1-05 Ir a la ubicación actual
Actor principal	Usuario App Móvil (A1)
Actores secundarios	Google Maps API (A5)
Descripción	El usuario A1 selecciona la opción de ir a la ubicación actual.
Precondiciones	Se ha cargado el mapa con la información de gasolineras
Postcondiciones	El mapa se desplaza a la ubicación del usuario
Flujo normal	1. El usuario selecciona la opción de ir a su ubicación actual. 2. La aplicación obtiene la ubicación del usuario. 3. La aplicación hace una llamada a Google Maps 4. La aplicación recibe la información de Google Maps. 5. La aplicación dibuja la zona del mapa en la que está situado el usuario.
Flujo alternativo	N/A
Prioridad	Alta

Tabla 5.7: RU-A1-05 Ir a la ubicación actual

Id y Nombre	RU-A2-01 Consultar datos
Actor principal	Usuario API-REST (A2)
Actores secundarios	N/A
Descripción	El usuario utiliza el servicio REST para consultar datos.
Precondiciones	N/A
Postcondiciones	Los datos se descargan correctamente
Flujo normal	1. El usuario hace una llamada al servicio REST. 2. El servicio REST devuelve información en formato JSON.
Flujo alternativo	2. El servicio REST da una respuesta 404 error si los filtros introducidos no son válidos.
Prioridad	Alta

Tabla 5.8: RU-A2-01 Obtener gasolineras

Id y Nombre	RU-A3-01 Descargar app móvil
Actor principal	Usuario app web (A3)
Actores secundarios	N/A
Descripción	El usuario A3 se descarga la app móvil.
Precondiciones	N/A
Postcondiciones	La aplicación se descarga correctamente
Flujo normal	1. El usuario A3 accede a la url `gasolineras.pablompg.com`. 2. El usuario A3 accede a la página Android. 3. El usuario selecciona la opción de descargar la aplicación. 4. El usuario se instala la aplicación.
Flujo alternativo	1. El servidor está caído y no puede acceder a la url de destino.
Prioridad	Alta

Tabla 5.9: RU-A3-01 Descargar app móvil

Id y Nombre	RU-A3-02 Ver página de equipo
Actor principal	Usuario app web (A3)
Actores secundarios	N/A
Descripción	El usuario A3 selecciona visualizar la página de equipo.
Precondiciones	N/A
Postcondiciones	El usuario A3 accede a la página de equipo correctamente.
Flujo normal	1. El usuario A3 accede a la url `gasolineras.pablompg.com`. 2. El usuario A3 accede a la página Equipo. 3. El usuario A3 selecciona a uno de los miembros del equipo. 4. El sistema redirecciona al usuario A3 a la página del miembro seleccionado.
Flujo alternativo	1. El servidor está caído y no puede acceder a la url de destino.
Prioridad	Alta

Tabla 5.10: RU-A3-02 Ver página de equipo

Id y Nombre	RU-A3-03 Acceder página personal
Actor principal	Usuario app web (A3)
Actores secundarios	N/A
Descripción	El usuario A3 accede a la página personal de un miembro del equipo.
Precondiciones	El usuario A3 ha accedido a la página de equipo (RU-A3-02).
Postcondiciones	El usuario A3 accede a la página personal correctamente.
Flujo normal	1. El usuario A3 selecciona la imagen de uno de los miembros del equipo. 2. La aplicación web redirecciona al usuario A3 a la página personal seleccionada.
Flujo alternativo	N/A
Prioridad	Baja

Tabla 5.11: RU-A3-03 Acceder página personal

Id y Nombre	RU-A0-01 Descargar datos
Actor principal	Sistema (A0)
Actores secundarios	Geoportal gasolineras (A4)
Descripción	El sistema (A0) descarga los datos del portal del Ministerio.
Precondiciones	N/A
Postcondiciones	Los datos se han descargado correctamente
Flujo normal	1. La plataforma (A0) hace una petición http al geoportal gasolineras (A4) solicitando el fichero actualizado de precios. 2. El portal del ministerio (A4) recibe la petición y devuelve el fichero solicitado. 3. La plataforma (A0) recibe el fichero Excel solicitado y lo guarda.
Flujo alternativo	2. El portal del ministerio (A5) devuelve un error 404 si está en proceso de mantenimiento.
Prioridad	Alta

Tabla 5.12: RU-A0-01 Descargar datos

Id y Nombre	RU-A0-02 Procesar datos
Actor principal	Sistema (A0))
Actores secundarios	Geoportal gasolineras (A4)
Descripción	La plataforma procesa los datos descargados.
Precondiciones	Los datos se han descargado (RU-A0-01)
Postcondiciones	Se ha guardado un csv con los nuevos datos generados tras el procesado.
Flujo normal	1. Se leen los datos del fichero Excel descargado. 2. Los datos del fichero Excel se cargan en memoria. 3. Se transforman los datos para eliminar datos erróneos, mal estructurados, incompletos, etc. 4. Los datos transformados se guardan en un csv.
Flujo alternativo	3. El proceso de limpieza de datos falla. 4. La plataforma guarda el error en un fichero log.
Prioridad	Alta

Tabla 5.13: RU-A0-02 Procesar datos

Id y Nombre	RU-A0-03 Guardar datos
Actor principal	Sistema (A0)
Actores secundarios	Geoportal Gasolineras (A4)
Descripción	La plataforma guarda los datos procesados y estructurados en una base de datos.
Precondiciones	Los datos se han descargado y procesado (RU-A0-02)
Postcondiciones	Los datos se han guardado en la base de datos y están disponibles para el usuario de la API-REST (A2)
Flujo normal	1. Se leen los datos del fichero csv generado con anterioridad. 2. Los datos se guardan en la base de datos. 3. La base de datos informa de que la inserción ha sido correcta.
Flujo alternativo	3. La base de datos informa de que ha habido algún error. 4. El mensaje de error se guarda en un fichero log.
Prioridad	Alta

Tabla 5.14: RU-A0-03 Guardar datos

5.5. Requisitos funcionales

A continuación se listan los Requisitos Funcionales, los cuales describen el comportamiento que debe mostrar el sistema bajo condiciones específicas.

ID	Requisitos funcionales
RF1	La aplicación habilitará la opción de seleccionar carburante.
RF2	La aplicación consultará en la API-REST los precios.
RF3	La aplicación te posicionará en tu ubicación.
RF4	La aplicación mostrará los precios de cada gasolinera.
RF5	Al seleccionar una gasolinera, la aplicación te dará la opción de ir hasta esa ubicación.
RF6	La aplicación habilitará la opción de seleccionar carburante
RF7	La API-REST permitirá filtrar gasolineras por id, provincia, margen de la carretera, tipo de venta, nombre de compañía y tipo de carburante.
RF8	La API-REST permitirá consultar precios en tiempo real filtrando por id, provincia, tipo de carburante, margen de la carretera al que está la gasolinera, tipo de venta y nombre de compañía.
RF9	La API-REST permitirá obtener series temporales de precios históricos asociados a gasolineras filtrando por id, provincia, tipo de carburante, margen de la carretera, tipo de venta y nombre de compañía.
RF10	La plataforma ejecutará el proceso de descarga, limpieza e inserción de datos cada hora.

TABLA 5.15: REQUISITOS FUNCIONALES

5.6. Requisitos no funcionales

Los requisitos no funcionales son aquellas restricciones del sistema que afectan a su desarrollo o comportamiento. Aquí se detallan 8 tipos distintos de requisitos no funcionales.

5.6.1. Implementación y arquitectura

Los requisitos de implementación describen las tecnologías que se utilizarán para la implementación de la aplicación móvil y la API-REST.

ID	Requisitos de implementación
RNF-IM1	La aplicación móvil se implementará usando el framework Flutter.
RNF-IM2	La API-REST debe ser independiente del resto de sistemas.
RNF-IM3	El proceso de actualización de los datos debe ser independiente.

Tabla 5.16: Requisitos de implementación

5.6.2. Interfaces externas

Describen la tecnología usada para desarrollar las comunicaciones entre distintos sistemas.

ID	Requisitos de interfaces externas
RNF-IE1	La comunicación con la API-REST se realizará mediante el protocolo HTTP.
RNF-IE2	La trasmisión de datos entre sistemas se realizará en formato Json.

Tabla 5.17: Requisitos de interfaces externas

5.6.3. Disponibilidad

La disponibilidad mide el tiempo en el que los servicios del sistema están disponibles.

ID	Requisitos de disponibilidad
RNF-DI1	El servidor que aloja al servicio REST tendrá un SLA de tres nueves (99,9 %).

Tabla 5.18: Requisitos de disponibilidad

5.6.4. Seguridad

La seguridad asegura la protección del sistema frente a ataques maliciosos.

ID	Requisitos de seguridad
RNF-SE1	Los sistemas sanetizarán todas las entradas proporcionadas por el usuario.
RNF-SE2	Los datos sensibles de configuración se guardarán como variables de entorno.
RNF-SE3	Se accederá al servicio REST como usuario con permisos solo de lectura.

Tabla 5.19: Requisitos de seguridad

5.6.5. Rendimiento

El rendimiento mide la respuesta del sistema ante las solicitudes y acciones de los usuarios.

ID	Requisitos de rendimiento
RNF-RE1	El tiempo de respuesta del servicio REST no será superior a un segundo para conexiones de 10 Mbps.
RNF-RE1	La base de datos soportará al menos 50 conexiones simultaneas.

Tabla 5.20: Requisitos de rendimiento

5.6.6. Escalabilidad

La escalabilidad determina la capacidad del sistema para gestionar un número creciente de usuarios, datos, transacciones, servidores, etc.

ID	Requisitos de escalabilidad
RNF-ES1	El sistema debe soportar la concurrencia de 50 usuarios concurrentes.
RNF-ES2	El sistema se debe adaptar fácilmente ante un incremento de demanda a corto plazo del 200 %.
RNF-ES3	El sistema deberá soportar modificaciones en su estructura física, sin costes elevados, para poder escalar el sistema ante un aumento de demanda de recursos.

Tabla 5.21: Requisitos de escalabilidad

5.6.7. Robustez

La robustez mide la capacidad del sistema para funcionar correctamente en situaciones anómalas.

ID	Requisitos de robustez
RNF-RO1	El sistema deberá presentar una gran robustez frente la ocurrencia de situaciones anómalas.

Tabla 5.22: Requisitos de robustez

5.6.8. Usabilidad

La usabilidad comprende todos aquellos factores que hacen que la aplicación sea atractiva o fácil de usar.

ID	Requisitos de usabilidad
RNF-US2	Tras leer el manual, un usuario debe de ser capaz de utilizar cualquier funcionalidad del sistema en menos de 20 minutos.

Tabla 5.23: Requisitos de usabilidad

5.7. Requisitos de información

En esta sección se describen los datos con los que operará el sistema. Se utilizará el modelo Entidad-Relación como modelo conceptual de datos específico para describir las características de información gestionada por el sistema.

5.7.1. Modelo Conceptual (E/R)

Para el modelado conceptual de datos se ha realizado una descripción de alto nivel de la estructura de la base de datos, independientemente del sistema gestor de base de datos (SGBD) que se implemente.

El modelo Entidad-Relación utilizado (Figura 5.5) describe:

- Los distintos tipos de entidades que modelan los diferentes objetos de datos para el sistema.
- Los atributos que caracterizan a cada uno de estos tipos de entidades.
- Los tipos de relaciones que describen las asociaciones existentes entre entidades.

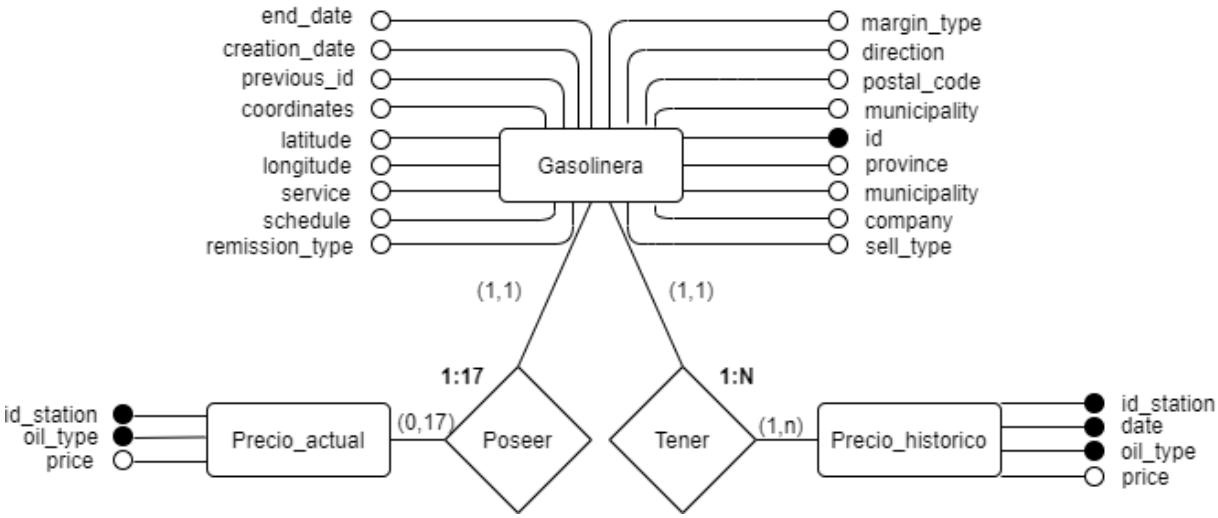


FIGURA 5.5: DIAGRAMA DE ENTIDAD-RELACIÓN

Se ha procurado diseñar un modelo conceptual lo más sencillo posible. El diagrama E/R tiene tres entidades:

- **Gasolineras** almacena todos los datos relativos a una gasolinera y permite mantener un histórico de gasolineras haciendo uso de los atributos *previous_id*, *creation_date* y *end_date*. El atributo *id* es su clave primaria.
- **Precio_histórico** almacena un histórico de precios que se actualiza una vez al día. La clave primaria está compuesta por *id_station*, *date* y *oil_type*. Por cada gasolinera habrá como máximo 17 instancias de precio actual, tantas como tipos de carburantes.
- **Precio_actual** es una entidad que almacena únicamente el último precio por cada tipo de carburante y gasolinera. La clave primaria está compuesta por *id_station* y *oil_type*

La entidad *Gasolinera* se relaciona con *Precio_historico* y *Precio_actual* mediante el atributo *id*, que tiene una correspondencia directa con *id_station* en las dos últimas entidades.

Este diseño permite tener una tabla de precios que almacena los datos históricos y otra tabla que solo almacena los precios actualizados. Este diseño permite realizar consultas a la tabla de precios actuales de manera muy rápida. Recordemos que la entidad *Precio_histórico* tiene más de 100 millones de registros, por lo cual es importante hacer un diseño eficiente que permita satisfacer los requisitos de rendimiento.

Capítulo 6

Diseño

A lo largo de este apartado se describen todos aquellos pasos llevados a cabo durante la etapa de diseño del proyecto, junto con los artefactos generados. Es importante señalar que se tienen tres sistemas que funcionan de forma independiente, pero que son capaces de interactuar entre ellos. Cada sistema contiene las integraciones de datos y código que se necesitan para llevar a cabo una función empresarial completa y diferenciada.

- **Sistema 1**: Se encarga de descargar un fichero Excel cada hora, para a continuación analizar y corregir errores del fichero y convertirlo en un csv. Finalmente inserta la información en una base de datos.
- **Sistema 2**: Es una aplicación web cuya principal característica es la API-REST que expone los datos de precios y gasolineras en formato json para ser consumidos por terceros. También tendrá la opción de descargar la aplicación móvil y de visualizar a los miembros del equipo de desarrollo.
- **Sistema 3**: Es la aplicación móvil que consume los datos de la API-REST y es utilizada por los usuarios finales.

El sistema 1 es totalmente independiente de los sistemas 2 y 3, puesto que no los necesita para llevar a cabo su función. Por otro lado, la API-REST expondrá los datos que previamente se hayan introducido en la base de datos por el sistema 1. Si el sistema 1 falla, los datos que exponga la API-REST no estarán actualizados. Por último, el sistema 3 (app móvil) consume los datos de la API-REST, por lo que necesita que esta esté operativa para poder proporcionar datos a los usuarios. La aplicación móvil también depende del sistema 1 en cuanto a que si este falla, los datos que verán los usuarios no estarán lo suficientemente actualizados.

Dentro de los próximos apartados, se explicarán las arquitecturas elaboradas para cada uno de los sistemas.

6.1. Arquitectura Lógica

La arquitectura lógica describe los componentes lógicos que componen el sistema y cómo se relacionan entre sí. Los servicios 2 y 3, siguen un diseño de tres capas estándar, mientras que el servicio 1 no cuenta con capa de presentación:

1. **Acceso a datos**: Esta capa nos permite acceder a los datos que utiliza el sistema.
2. **Lógica de negocio**: Es la capa encargada del control del flujo de datos del sistema y las operaciones que afectan a estos.
3. **Presentación**: Se trata de la capa encargada de gestionar la presentación de la información a los usuarios

Tras analizar con detalle la funcionalidad del sistema y todas las restricciones de implementación, se obtiene un primer diagrama con la Arquitectura Lógica de alto nivel del sistema (figura 6.1).

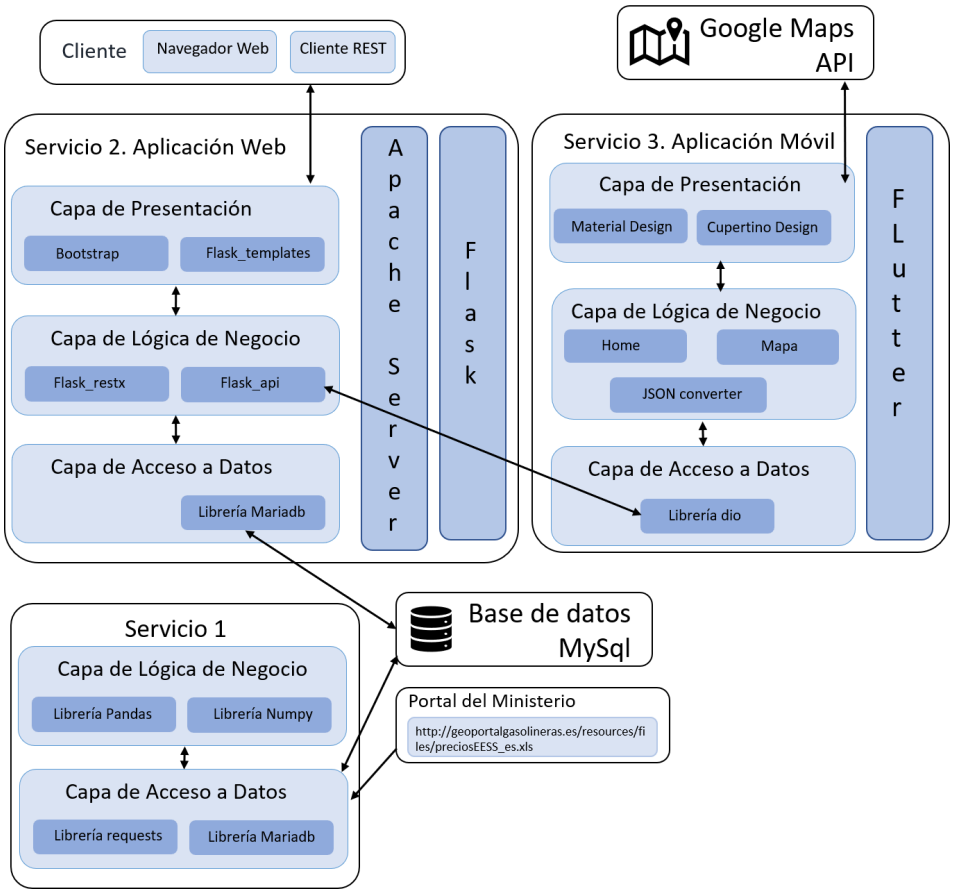


FIGURA 6.1: ARQUITECTURA LÓGICA

6.2. Arquitectura Física

En esta sección se va a detallar la Arquitectura Física a utilizar. Para elaborar esta arquitectura se han de tener en cuenta los requisitos no funcionales detallados en la fase de Análisis. Más concretamente, son imprescindibles los requisitos no funcionales de Seguridad, Disponibilidad y Escalabilidad.

Es importante señalar que la universidad solo provee de una única máquina virtual para la elaboración del proyecto. Es por ello que en la práctica el servidor de aplicaciones y la base de datos están en la misma máquina virtual. Sin embargo, lo ideal sería poder hacer un diseño por capas en el que se diferencien los siguientes apartados:

- **Cliente**: Dispositivo móvil del usuario que utiliza la aplicación
- **Servidor de aplicaciones**: Servidor que gestiona las peticiones y ejecuta la lógica de negocio
- **Base de datos**: Servidor en el que se ejecuta el motor de la base de datos y en el que está almacenada la información.

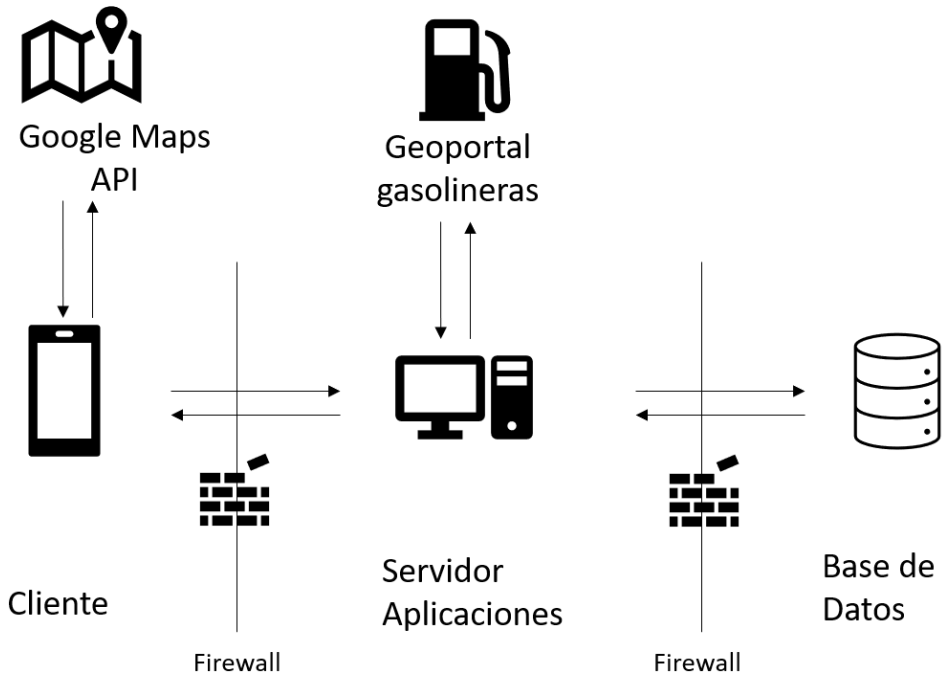


Figura 6.2: Arquitectura física

Con esta arquitectura y haciendo un buen aprovisionamiento de recursos en el servidor de aplicaciones y en el servidor de base de datos, se podría cumplir con los requisitos de disponibilidad y rendimiento. Si se quisiera hacer una aplicación más robusta y resiliente, hay varias aspectos de mejora posibles:

- En cuanto a los servidores de aplicaciones, se pueden utilizar dos servidores de balanceo de carga, uno activo que se encargue de atender las peticiones desde el exterior del sistema, y otro pasivo de respaldo, utilizado en aquellos casos en los que el servidor de balanceo de carga activo falle, asegurando de esta forma una alta disponibilidad del sistema. Del mismo modo se podrían utilizar dos servidores de aplicaciones idénticos que nos aseguren una alta disponibilidad del sistema.

- Para el acceso a los datos se podría seguir una estructura similar con balanceadores de carga. En este caso, se podría disponer de un balanceador principal que acepte las peticiones procedentes de los servidores de aplicaciones, y que las distribuya entre los servidores de bases de datos según la carga de trabajo de los mismos. Para poder mantener el estado en las bases de datos, se podría implementar un sistema de maestro-esclavo, donde los esclavos responden a peticiones de lectura, y el maestro responde a las peticiones de escritura, propagando a los esclavos las modificaciones una vez realizadas.

- Para el acceso a datos se podrían cachear las consultas más frecuentes para evitar hacer llamadas a la base de datos. De esta manera, cada hora se actualizaría la caché con los datos en tiempo real y las consultas desde la aplicación móvil se responderían sin necesidad de sobrecargar al motor de la base de datos.

En general, el uso de balanceadores permite distribuir las peticiones entre varios servidores en función de su carga de trabajo, mejorando así también el rendimiento de la aplicación. Esto aplica tanto a servidores de bases de datos como de aplicaciones.

Por último se muestra el diagrama de despliegue del sistema. Es él se describen los dispositivos físicos que componen la arquitectura física, y el entorno de ejecución que contiene cada uno de ellos. Además se muestran las relaciones entre entornos y los protocolos utilizados para comunicarse. Además se añaden los artefactos a desplegar en cada servidor 6.3.

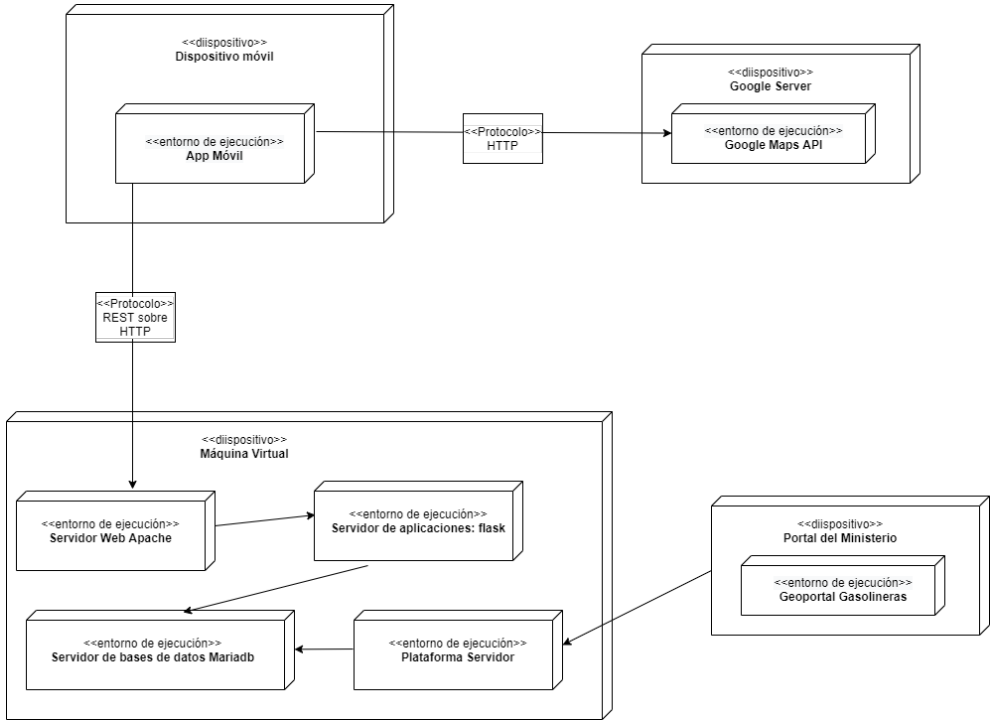


Figura 6.3: Diagrama de despliegue

6.3. Diagrama de Clases

A continuación, en la figura 6.4 se muestra el diagrama de clases elaborado en la fase de diseño. En él se modelan todos las clases que componen la aplicación junto con las relaciones entre ellas. En este modelo quedan excluidos todos los componentes que componen las vistas y recursos como las hojas de estilo o los archivos javascript.

El diagrama de clases se ha agrupado en 4 grandes bloques:

- **Servicio REST:** Incluye las clases utilizadas para el sistema encargado del servicio REST
- **Servidor de aplicaciones:** Incluye las clases utilizadas para el servidor de aplicaciones (del cual el servicio REST forma parte).
- **Plataforma Servidor:** Incluye las clases utilizadas para descargar los ficheros Excel del portal del ministerio, el procesamiento de los datos y su inserción en la base de datos.
- **App móvil:** Aquí se encuentran las clases utilizadas en la aplicación móvil como clases json, mapa, etc. Aquí se incluye todo el código programado en *dart* con el framework *flutter*.

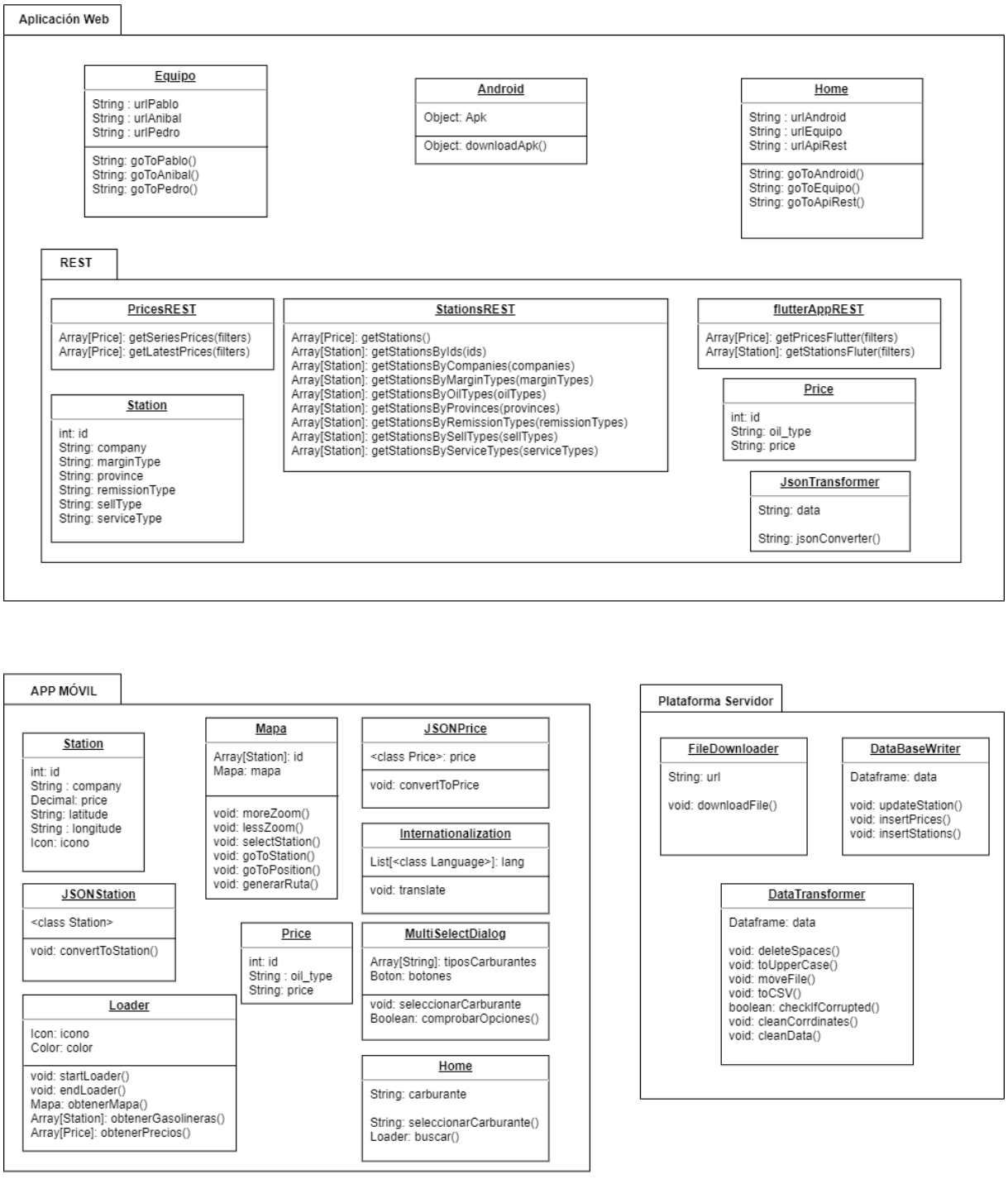


FIGURA 6.4: DIAGRAMA DE CLASES

6.4. Modelo Lógico de Datos

El modelado lógico de datos parte del resultado del concepto conceptual (Modelo Entidad-Relación) y da como resultado una descripción de la estructura de la base de datos. El diseño lógico depende del tipo de SGBD que se vaya a utilizar. Para bases de datos relacionales (como la utilizada), el diseño lógico consiste en definir las tablas que se van a utilizar, las relaciones entre ellas y normalizarlas.

En este apartado se procede al diseño de la base de datos de acuerdo con el análisis realizado previamente junto al diccionario de datos. El proyecto va a ser implementado bajo una base de datos relacional: mariadb. Esta decisión es debido a los siguientes motivos:

- Permite representar relaciones entre entidades.
- Proveen herramientas que garantizan evitar la duplicidad de registros.
- Garantizan la integridad referencial.
- Es sencillo de utilizar.
- Es escalable.

Para representar el diseño de una base de datos relacional, se usa el Modelo Relacional que puede verse a continuación en la Figura 6.5.

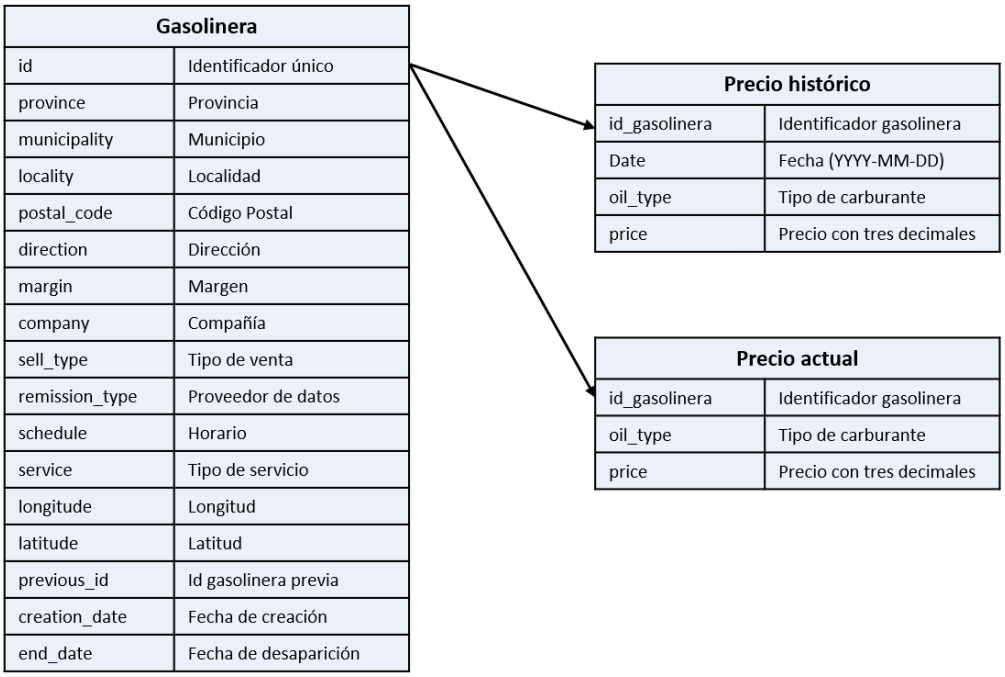


Figura 6.5: Modelo relacional

6.4.1. Diccionario de datos

El diccionario de datos especifica de forma detallada los requisitos de información. Describe las entidades de datos utilizadas en el sistema y sus relaciones, todas ellas modeladas en el Diagrama Entidad-Relación. Esta información se transformará en el esquema de la base de datos, tablas, atributos y, finalmente, en las variables que se utilizarán en el sistema. Cada requisito de información se corresponde con un atributo y estos están organizados por entidades.

RI-E1 GASOLINERA				
Definición	Representa de forma unívoca a cada una de las gasolineras			
ATRIBUTOS				
Id	**Nombre**	**Descripción**	**Dominio**	**Reestricción**
RI-E1.01	id	Identificador único	Integer	PK Autoincrement
RI-E1.02	province	Provincia	Enum	
RI-E1.03	municipality	Municipio	Varchar(150)	
RI-E1.04	locality	Localidad	Varchar(150)	
RI-E1.05	postal_code	Código Postal	Varchar(15)	
RI-E1.06	direction	Dirección	Varchar(150)	
RI-E1.07	margin_type	Margen de la carretera	Enum	
RI-E1.08	company	Compañía	Varchar(150)	
RI-E1.09	sell_type	Tipo de venta	Enum	
RI-E1.10	remission_type	Proveedor de datos	Enum	
RI-E1.11	schedule	Horario	Varchar(1000)	
RI-E1.12	service	Tipo de servicio	Varchar(1000)	
RI-E1.13	longitude	Longitud	Decimal(8,6)	
RI-E1.14	latitude	Latitud	Decimal(9,6)	
RI-E1.15	coordinates	Coordenadas	Point	
RI-E1.16	previous_id	Id gasolinera previa	Integer	Default NULL
RI-E1.17	creation_date	Fecha de creación	Date	Default NULL
RI-E1.18	endn_date	Fecha de desaparición	Date	Default NULL

TABLA 6.1: RI-E1 GASOLINERA

El atributo *RI-E1.02 province* es un enumerado, cuyos valores pueden ser: *VI, AB, A, AL, O, AV, BA, PM, B, BU, CC, CA, S, CS, CEUTA, CR, CO, C, CU, GI, GR, GU, SS, H, HU, J, LE, L, LU, M, MA, MELILLA, MU, NA, OR, P, GC, PO, LO, SA, TF, SG, SE, SO, T,*

TE, TO, V, VA, BI, ZA, Z.

El atributo *RI-E1.07 margin_ type* es un enumerado, cuyos valores pueden ser: *D, I, N* y que hacen referencia a si la gasolinera está al margen derecho de la carretera, al izquierdo o es indiferente.

El atributo *RI-E1.09 sell_ type* es un enumerado, cuyos valores pueden ser: *P, R, A* donde la primera opción hace referencia a que la gasolinera vende a particulares, la segunda opción que vende a un público restringido como puedan ser cooperativas y la última opción permite vender a todo tipo de público.

El atributo *RI-E1.10 remission_ type* es un enumerado, cuyos valores pueden ser: *OM, dm* donde la primera opción hace referencia a que la información del precio ha sido proporcionada por el operador mayorista, mientras que la segunda indica que ha sido proporcionada por un operador minorista.

El atributo *RI-E1.11 service* indica en qué horario una gasolinera es autoservicio y cuando dispone de personal.

RI-E2 PRECIO_HISTÓRICO				
Definición	Almacena precios de carburantes por día y gasolinera			
ATRIBUTOS				
Id	**Nombre**	**Descripción**	**Dominio**	**Reestricción**
RI-E2.01	id_station	Identificador de la gasolinera	Integer	PK Compuesta
RI-E2.02	date	Fecha	Date	PK Compuesta
RI-E2.03	oil_type	Tipo de carburante	Enum	PK Compuesta
RI-E2.04	price	Precio	Decimal(6,3)	

Tabla 6.2: RI-E2 Precios históricos

El atributo *R1-E2.03 oil_ type* es un enumerado, cuyos valores pueden ser: *GSLN95_ E5, GSLN95_ E10, GSLN95_ E5_ P, GSLN98_ E5, GSLN98_ E10, GSL_ A, GSL_ P, GSL_ B, GSL_ C, BIOETANOL, BIOALCOHOL, BIODIESEL, ESTER_ METILICO, GLP, GNC, GNL, H.*

Este requisito de información permite almacenar un histórico diario sobre la evolución de los precios, pero no permite trabajar en tiempo real, puesto que almacenar en una misma entidad ambos valores haría que la tabla creciera muy rápido, impidiendo hacer consultas lo suficientemente rápidas como para cumplir los requisitos de rendimiento. Es por ello que se ha creado otra entidad muy similar, que almacena únicamente los precios actuales. Esto es, almacena un único registro por cada gasolinera y tipo de carburante, correspondiente al precio más actual.

La tabla 6.3 muestra la entidad *PRECIO_ ACTUAL* que se actualiza cada hora con los precios de los carburantes. Esta tabla se borra semanalmente, de tal forma que solo se tiene registro de la evolución de los precios con precisión horaria de la última semana. Esto se hace así para hacer que las consultas sean eficientes.

RI-E3 PRECIO_ACTUAL				
Definición	Almacena precio de carburantes por hora y gasolinera.			
ATRIBUTOS				
Id	**Nombre**	**Descripción**	**Dominio**	**Reestricción**
RI-E2.01	id_station	Identificador de la gasolinera	Integer	PK Compuesta
RI-E2.02	oil_type	Tipo de carburante	Enum	PK Compuesta
RI-E2.03	price	Precio	Decimal(6,3)	

TABLA 6.3: RI-E3 PRECIO ACTUAL

En la tabla 6.4 se muestra la relación *TENER*. Una gasolinera ha de tener al menos un precio, pero lo habitual es que tenga más. Tendrá tantos precios como días lleve en activo multiplicado por el número carburantes distintos que comercialice esa gasolinera.

RI-R1 TENER				
Definición	Asociación entre gasolinera y precio histórico.			
ATRIBUTOS				
Id	**Nombre**	**Descripción**	**Dominio**	**Reestricción**
RI-R1.01	id_station	Identificador único de la gasolinera	Integer	PK Compuesta
RI-R1.02	date	Fecha	Date	PK Compuesta
RI-R1.03	oil_type	Tipo de carburante	Enum	PK Compuesta

TABLA 6.4: RI-R1 TENER

RI-R2 POSEER				
Definición	Asociación entre gasolinera y precio actual.			
ATRIBUTOS				
Id	**Nombre**	**Descripción**	**Dominio**	**Reestricción**
RI-R2.01	id_station	Identificador único de la gasolinera	Integer	PK Compuesta
RI-R2.02	oil_type	Tipo de carburante	Enum	PK Compuesta

Tabla 6.5: RI-R2 Poseer

6.5. Diagramas de Secuencia

Los diagramas de secuencia describen las interacciones entre los componentes del sistema a lo largo del tiempo para realizar una determinada operación. Son usados para clarificar la interpretación de los casos de uso.

A continuación se muestran los diagramas de Secuencia asociados a los casos de uso más complejos: Crear ruta (RU-A1-04), Consultar datos (RU-A2-01), Descargar app móvil (RU-A3-01), Acceder página personal (RU-A3-03) y Guardar datos (RU-A4-03).

6.5.1. Crear Ruta (RU-A1-04)

Este diagrama describe la interacción que se realiza con la aplicación externa Google Maps para poder generar una ruta desde la ubicación del usuario hasta la gasolinera de destino. Para ello primero el usuario deberá selccionar un carburante y esperar a que la aplicación móvil cargue el mapa con la información recibida de la API-REST y Google Maps.

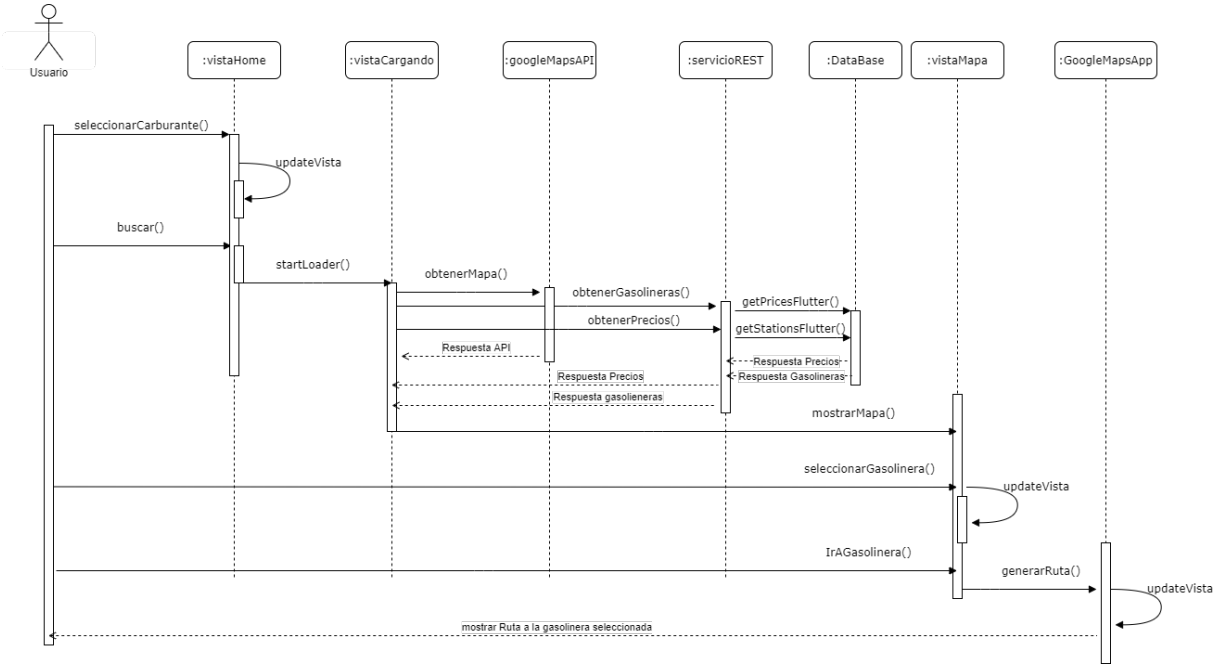


FIGURA 6.6: DIAGRAMA DE SECUENCIA PARA RU-A1-04

6.5.2. Consultar datos (RU-A2-01)

Este diagrama describe la interacción que se realiza entre el usuario A2 y la API-REST cuando se desea obtener datos. El usuario realizará una petición http a la API aplicando los filtros deseados y recibirá la información en formato json.

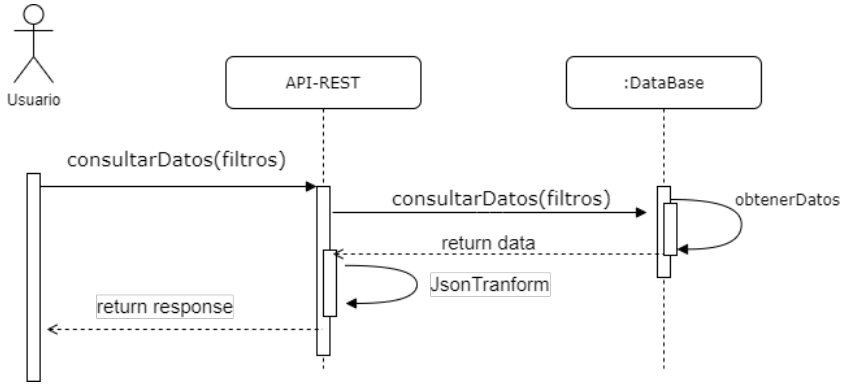


Figura 6.7: diagrama de secuencia para RU-A2-01

6.5.3. Descargar app móvil (RU-A3-01)

Este diagrama describe la interacción que realiza el usuario de la aplicación web (A3) para descargarse la aplicación móvil.

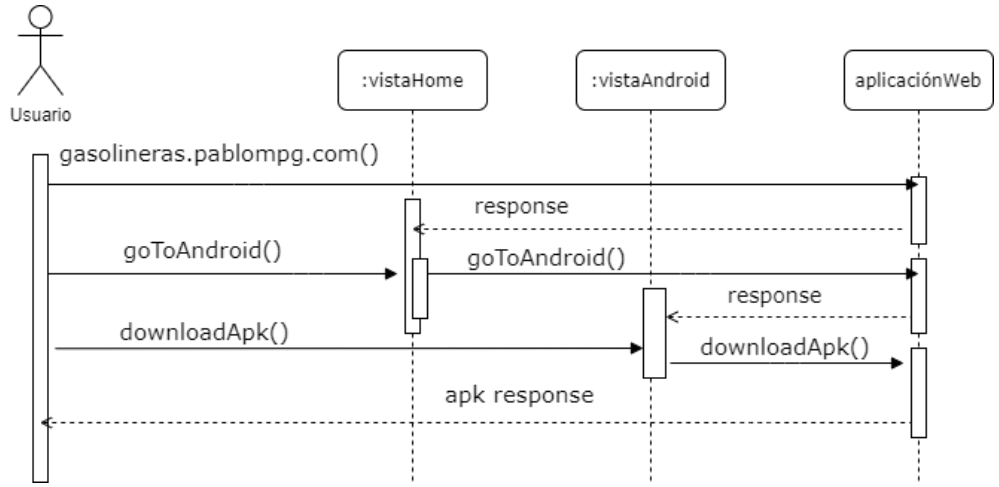


Figura 6.8: diagrama de secuencia para RU-A3-01

6.5.4. Acceder página personal (RU-A3-03)

Este diagrama describe la interacción que realiza el usuario de la aplicación web (A3) para acceder a la página personal de alguno de los miembros del equipo.

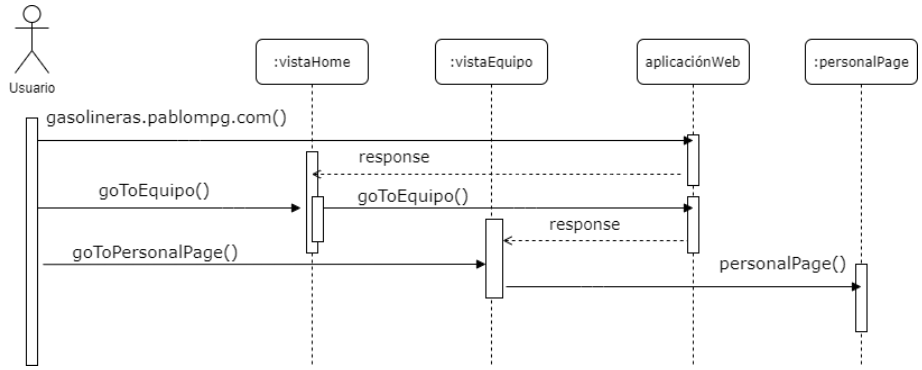


FIGURA 6.9: DIAGRAMA DE SECUENCIA PARA RU-A3-03

6.5.5. Guardar datos (RU-A0-03)

Este diagrama de secuencia describe las interacciones que realiza el sistema de manera automatizada para descargarse los datos desde el portal geogasolineras, procesar los datos y guardarlos en la base de datos.

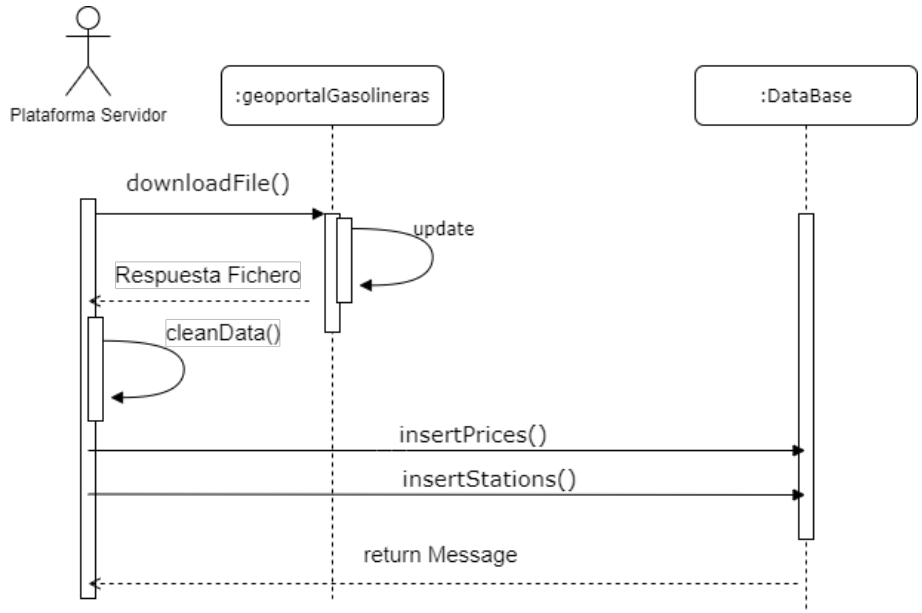


FIGURA 6.10: DIAGRAMA DE SECUENCIA PARA RU-A4-03

6.6. Diseño de interfaces

En este proyecto hay dos tipos de interfaces:

- **Interfaces de la API-REST:** Se corresponden a la aplicación web en la que está alojada la API REST y que además incluye un enlace para descargar la aplicación móvil y un apartado en el que se muestran los miembros del equipo. Se han diseñado dos interfaces:
  - Página de inicio
  - Página de equipo
- **Interfaces de la aplicación móvil:** Estas interfaces son con las que interactuará el usuario final. En este apartado se van a desarrollar tres interfaces:
  - Pantalla de inicio
  - Pantalla del mapa
  - Pantalla del mapa con gasolinera seleccionada

6.6.1. Interfaces de la Aplicación Web

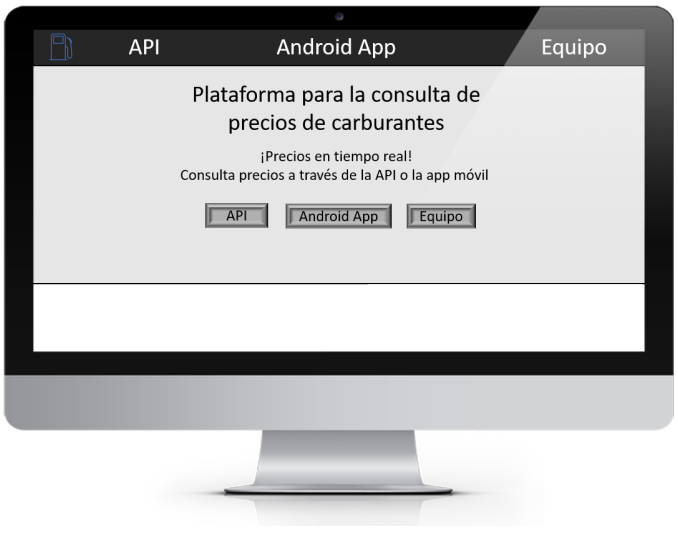
Id y Nombre	**IN-01 Ventana de Inicio de la aplicación web**
Descripción	Es la ventana inicial de la aplicación web.
Activación	Accediendo a https://gasolineras.pablompg.com.
Boceto	
Eventos	Acceder a la API, acceder a la página de equipo y descargar aplicación Android.

Tabla 6.6: Aplicación web ventana de inicio

La interfaz de la tabla 6.6 es un boceto correspondiente a cómo deberá verse la página principal de la aplicación web disponible en `gasolineras.pablompg.com`. La página principal contará con tres opciones:

- **API**: Dará acceso al servicio rest.
- **Android App**: Servirá para descargarse la aplicación web.
- **Equipo**: Dará acceso a la página de equipo.

Por otro lado, en la interfaz de la tabla 6.7 se puede visualizar cómo será la página de equipo de la aplicación web.

Id y Nombre	**IN-02 Ventana de Equipo de la aplicación web**
Descripción	Es la ventana de equipo de la aplicación web.
Activación	Accediendo a https://gasolineras.pablompg.com/equipo.
Boceto	
Eventos	Acceso a las páginas personales de los tutores y alumno.

TABLA 6.7: APLICACIÓN WEB VENTANA DE EQUIPO

6.6.2. Interfaces de la API-REST

En la tabla 6.8 se puede observar cómo será la interfaz de la API-REST. Esta interfaz permite visualizar de qué métodos y filtros dispone el servicio. La interfaz de usuario no es la forma más habitual de utilizar el servicio, ya que las llamadas se suelen hacer de manera automatizada desde una máquina a través de peticiones http.

Es importante señalar que esta interfaz sigue el estándar Swagger [6] y ha sido generada de manera automática a través de la librería Flask-RESTful[7].

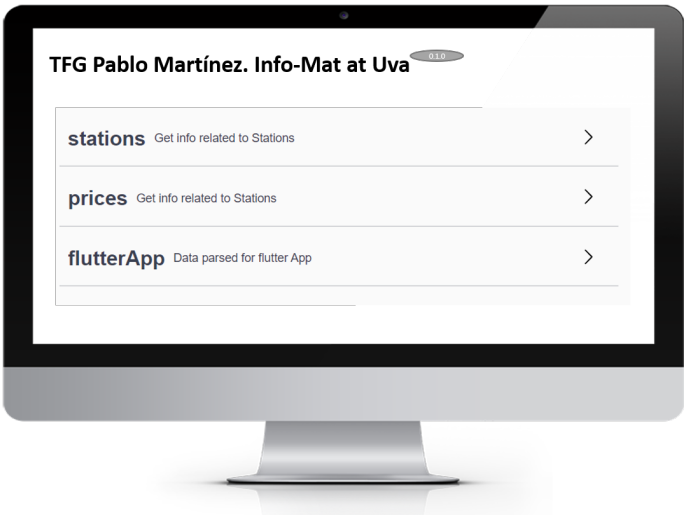
Id y Nombre	**IN-04 Ventana principal de la API-REST**
Descripción	Es la ventana de acceso a la API-REST.
Activación	Accediendo a https://gasolineras.pablompg.com/api/v0.
Boceto	
Eventos	Acceso a los métodos de la API - REST.

Tabla 6.8: API REST ventana principal

6.6.3. Interfaces de la aplicación móvil

En esta interfaz (tabla 6.9) se muestra la pantalla principal de la aplicación móvil. Aquí el usuario deberá seleccionar el carburante que desea y a continuación deberá seleccionar el botón de buscar.

Id y Nombre	**IN-05 Ventana inicial de la aplicación móvil**
Descripción	Pantalla Inicial de la aplicación móvil.
Activación	Iniciando la aplicación desde el móvil.
Boceto	
Eventos	Seleccionar carburante y buscar gasolineras.

TABLA 6.9: PANALLA DE INICIO APLICACIÓN MÓVIL

En la tabla 6.11 se muestra el boceto de la interfaz de usuario una vez se ha cargado el mapa con gasolineras. Deberá aparecer un mapa sobre el cual aparecen gasolineras en distintos colores, indicando si el precio de la gasolinera es caro o barato en comparación con el resto. Esta interfaz también tendrá la opción de pulsar el botón de *Mi Posición* para moverse en el mapa hasta la posición del usuario.

Id y Nombre	**IN-06 Ventana con mapa y gasolineras**
Descripción	Pantalla con mapa y gasolineras.
Activación	Seleccionar buscar en la pantalla de inicio.
Boceto	
Eventos	Seleccionar gasolinera.

Tabla 6.10: Vista principal con gasolineras

Por último en la tabla 6.11 se muestra un boceto de como deberá ser la interfaz que visualizará el usuario una vez seleccione una gasolinera. Deberá mostrarse el mismo mapa que en la interfaz IN-06 pero con información extra de la gasolinera seleccionada así como la opción de generar una ruta hasta dicha gasolinera.

Id y Nombre	**IN-07 Ventana con mapa y gasolinera seleccionada**
Descripción	Pantalla cuando se selecciona una gasolinera.
Activación	Se ha seleccionado una gasolinera en el mapa.
Boceto	
Eventos	Generar ruta hasta gasolinera.

TABLA 6.11: VISTA PRINCIPAL CON GASOLINERAS

Capítulo 7

Implementación

En este apartado se detallarán algunos de los aspectos claves de la implementación llevada a cabo a nivel técnico en el proyecto. Este capítulo se dividirá en tres subsecciones.

- **Descarga de datos**: Este apartado describirá la implementación llevada a cabo en el sistema encargado de descargar, transformar y guardar los datos.
- **Aplicación web**: En este apartado se aborda la implementación de la aplicación web y en particular del servicio REST desarrollado.
- **Aplicación móvil**: En este último apartado se describen las tecnologías usadas y la implementación requerida para desarrollar con éxito la aplicación móvil en Flutter.

7.1. Descarga, transformación y guardado de datos

7.1.1. Descarga de datos

Para la descarga de datos se ha utilizado un sencillo script de python que se encarga de descargar un fichero y lo guarda en una carpeta específica. Este script se ejecuta cada hora con el objetivo de tener datos lo más actualizados posibles. Para programar la ejecución se ha utilizado *crontab*, un planificador de trabajos embebido en los sistemas operativos de tipo Unix.

Como estos scripts se ejecutan en la misma máquina que el resto de sistemas, se hace imprescindible hacer uso de entornos virtuales de python que permitan aislar dependencias entre proyectos. Para poder lanzar una ejecución desde crontab haciendo uso de un entorno virtual se ha utilizado el siguiente código:

```
0 * * * * echo 'source
    /home/pablo/python_projects/tfg_intodb/venv/bin/activate; python3
    /home/pablo/python_projects/tfg_intodb/downloadxls_tocsvtodb_hourly.py' |
    /bin/bash
```

Los ficheros que se descargan cada hora se eliminan tras guardar la información en la base de datos. Esto se hace para prevenir quedarnos sin almacenamiento en la máquina virtual. Sin embargo sí que se guarda un fichero diario de precios. Este fichero se descarga a las 9:00 de la mañana, y solo se vuelve a descargar si el proceso falla. Para ejecutar este script se ha utilizado la siguiente entrada en crontab:

```
0 9-21 * * * echo 'source
    /home/pablo/python_projects/tfg_intodb/bin/venv/activate; python3
    /home/pablo/python_projects/tfg_intodb/downloadxls_tocsvtodb.py' |
    /bin/bash
```

Cuando el proceso de descarga finaliza hace una llamada al proceso de transformación de datos para que este comience. La implementación de este proceso es sencilla pues como hemos visto basta con crearse un entorno de python virtual, programar un script de descarga de datos y programar una entrada en crontab.

7.2. Transformación de los datos

Los datos que proporciona el portal geogasolineras necesitan ser transformados y estandarizados para poder trabajar con ellos. Una buena práctica a la hora de trabajar con datos externos sobre los cuales no tienes información de cómo se han generado es sanetizarlos. Es por tanto esa la primera tarea que se realiza con los datos. El proceso de sanetización consta de:

- Convertir todo el texto a mayúsculas.
- Eliminar saltos de línea en el texto de una misma celda.
- Eliminar espacios iniciales, finales y dobles de una misma celda.
- Eliminar vectores de ataque e inyección SQL utilizando las funciones estándar de python.

A lo largo de los años se ha observado que el nombre de las provincias variaba en los ficheros descargados. Esto se debe a que en ocasiones se ha utilizado el nombre de provincia en castellano, otras veces su abreviatura y en la actualidad el nombre de la provincia en la lengua cooficial si procede. Para prevenir este tipo de modificaciones se realiza un mapeo de provincias a códigos de provincia según el estándar ISO 3166-2:ES [8].

Una vez los datos se han transformado, se guardan en un fichero csv y se llama al proceso de guardado de datos que guardará los datos en la base de datos. Para la transformación de datos se han utilizado las librerías pandas y numpy.

7.2.1. Guardado de datos

Como base de datos se ha utilizado mariadb. Mariadb es una de las bases de datos más ligeras que podemos encontrar. Esto hace que sea una tecnología muy viable para poder implementar el proyecto en una máquina virtual con recursos limitados. Además es una tecnología que lleva años en el mercado y es utilizada en producción por centenares de empresas.

El problema al que nos enfrentamos en este proyecto es un problema común a muchos casos de uso. Se tienen que realizar grandes ingestas de datos de forma puntual, sin que estos momentos puntuales alteren el rendimiento del sistema.

La forma habitual para permitir hacer consultas de datos rápidas es hacer uso de índices. La contrapartida de usar índices es que cuando se hace una inserción, el árbol asociado al índice se tiene que volver a construir, lo cual hace que las inserciones de datos sean más lentas que cuando se inserta en una tabla sin índices. Esto hace que nuestra base de datos tenga una sobrecarga considerable en los momentos puntuales en los que se insertan datos.

La tabla de precios históricos cuenta con tres índices distintos, por lo que por cada inserción se tienen que reconstruir tres árboles distintos. Esto es incompatible con los requisitos de rendimiento que tiene nuestro sistemas. Se valoraron varias alternativas para afrontar este problema entre las cuales estaba el utilizar una base de datos no relacional. Finalmente se optó por una solución más práctica que permitía utilizar la arquitectura ya existente sin necesidad de añadir más complejidad: crear una tabla que almacene el último precio para cada gasolinera y tipo de carburante. De esta manera se tienen dos tablas para almacenar precios, una tabla para precios históricos y que se actualiza cada día y es incremental, y otra tabla para precios actuales, la cual se actualiza cada hora pero no es incremental puesto que lo que hace es actualizar el valor del precio en lugar de añadir una nueva fila.

Las consultas desde la aplicación móvil se realizan contra la tabla de precios actuales, reduciendo drásticamente el número de consultas a realizar sobre la tabla de precios históricos, la cual cuenta con cientos de millones de filas y tarda algo más de un minuto en insertar los datos diarios. De esta manera nuestro sistema sigue cumpliendo con los requisitos de rendimiento cuando se hacen las inserciones de datos diarios.

7.3. Aplicación web

Son múltiples las tecnologías y herramientas que se pueden utilizar a la hora de implementar una aplicación web. Yo me decanté por utilizar algún framework en python ya que es el lenguaje con el que más experiencia tengo y uno de los más demandados en el mercado laboral. Python cuenta principalmente con dos frameworks para desarrollar aplicaciones web:

- **Django**: Framework open source pesado pero muy potente. Es utilizado por compañias como Facebook (en instagram) y Mozilla.
- **Flask**: Framework muy ligero y modularizable utilizado por compañías como Netflix o Reddit.

Ambos frameworks tienen importantes compañías utilizándolos y grandes comunidades que los dan soporte. Cualquiera de las dos soluciones hubiera sido válida, aunque finalmente se utilizó Flask por ser más ligero.

Para la parte de interfaz de usuario de la aplicación web se ha utilizado una plantilla de bootstrap a la cual se le ha dado forma para acoplarse al proyecto. Se ha utilizado bootstrap por ser un framework sencillo de usar aunque limitado. Como la parte de interfaz de usuario de la aplicación web se esperaba que fuera sencilla, este framework era ideal.

7.3.1. API-REST

Para el desarrollo del servicio REST se ha utilizado un módulo de flask llamado Flask-restful [9]. Este módulo permite construir puntos de acceso de manera muy sencilla utilizando anotaciones como las que se muestran a continuación:

```
1 api = Namespace('stations', description='Get info related to Stations')
2
3 station_parser = reqparse.RequestParser(bundle_errors=True)
4 station_parser.add_argument('id', type=int, required=False, action='split')
5 station_parser.add_argument('province', type=str, required=False,
      action='split')
6 station_parser.add_argument('margin_type', type=str, required=False,
      action='split')
7 station_parser.add_argument('sell_type', type=str, required=False,
      action='split')
8 station_parser.add_argument('company', type=str, required=False,
      action='split')
9 station_parser.add_argument('oil_type', type=str, required=False,
      action='split')
```

```
10
11 @api.route('', doc={"description": "Get all the info related to stations"})
12 class Provinces(Resource):
13     @api.expect(station_parser, validate=True)
14     def get(self):
15
16         args = station_parser.parse_args()
17
18         given_args = {
19             'id': args['id'],
20             'province': args['province'],
21             'margin_type': args['margin_type'],
22             'sell_type': args['sell_type'],
23             'company': args['company'],
24             'oil_type': args['oil_type'],
25         }
```

Además, este módulo cuenta con la ventaja de que construye de manera automática una interfaz de usuario siguiendo el estándar de swagger [6] para interactuar con el servicio REST. Esta técnica es habitual en la industria, donde es muy importante no solo tener un servicio REST, sino una interfaz que permita al usuario comprender de manera sencilla qué métodos de filtrado admite el servicio y qué puntos de acceso tiene. El servicio REST desarrollado tiene tres puntos de acceso:

- **Stations**: Este punto de acceso permite obtener información de gasolineras (coordenadas, provincia, código postal, etc.).
- **Prices**: Este punto de acceso permite consultar datos de precios tanto históricos como en tiempo real. Permite multitud de filtros como marca, provincia, tipo de carburante, etc.
- **flutterApp**: Este punto de acceso está especialmente diseñado para optimizar la transferencia de datos entre la API-REST y la aplicación móvil. Permite obtener datos de gasolineras y de precios actuales.

Para enviar el mínimo de información posible por la red, el punto de acceso **flutterApp** no sigue el estándar de la industria sino que envía la información organizada de manera ligeramente distinta. Mientras que el punto de acceso **Stations** genera un objeto json por cada gasolinera, el punto de acceso utilizado por la aplicación móvil agrupa la información en arrays, evitando la duplicidad de las claves que se genera al utilizar arrays de objetos.

Esta es una respuesta devuelta por el punto de accceso Stations tras filtrar por los ids 1 y 2.

```
{
  "description": "Station info given its id",
  "length": 2,
  "data": [
    {
      "id": 1,
      "province": "AB",
      "municipality": "ABENGIBRE",
      "locality": "ABENGIBRE",
      "postal_code": "02250",
      "direction": "AVENIDA CASTILLA LA MANCHA, 26",
      "margin_type": "D",
      "company": "CEPSA",
      "sell_type": "P",
      "remission_type": "dm",
      "schedule": "L-D: 07:00-22:00",
      "service": "L-D: 07:00-22:00 (A)",
      "longitude": "-1.539167",
      "latitude": "39.211417"
    },
    {
      "id": 2,
      "province": "AB",
      "municipality": "ALATOZ",
      "locality": "ALATOZ",
      "postal_code": "02152",
      "direction": "CR CM-332, 46,4",
      "margin_type": "I",
      "company": "REPSOL",
      "sell_type": "P",
      "remission_type": "dm",
      "schedule": "L-D: 7:00-23:00",
      "service": "L-D: 7:00-23:00 (A)",
      "longitude": "-1.346083",
      "latitude": "39.100389"
    }
  ]
}
```


Se puede observar como cada objecto json se corresponde a una gasolinera y las claves de los objetos se tienen que escribir tantas veces como objetos haya. Por otro lado, el punto de acceso flutterApp/stations devuelve la siguiente respuesta cuando se filtra por los ids 1 y 2.


```
{
  "description": "prices today (2021-06-08 at 9:00) classified by oil_type",
```

```
  "length": 4,
  "data": {
    "ids": [
      1,
      2
    ],
    "latitude": [
      "39.211417",
      "39.100389"
    ],
    "longitude": [
      "-1.539167",
      "-1.346083"
    ],
    "company": [
      "CEPSA",
      "REPSOL"
    ]
  }
}
```


Esta segunda respuesta evita tener que escribir las claves de los objetos tantas veces como objetos haya, sino que las escribe una única vez y agrupa la información en arrays de strings. A la hora de procesar la información es imprescindible tener en cuenta la posición que una gasolinera ocupa en el array para relacionar la información de distintos arrays. Esto añade complejidad al procesado de datos en la aplicación móvil, pero ahorra bancho de anda y disminuye el tiempo de espera a respuestas de la API.

7.4. Aplicación móvil

La aplicación móvil se ha desarrollado utilizando Flutter, un framework de código abierto desarrollado por Google para crear aplicaciones nativas de forma rápida y sencilla. Su principal ventaja es que ahorra esfuerzo de desarrollo, pues permite desarrollar simultáneamente aplicaciones nativas en Windows, Mac, iOS y Android. Para escribir aplicaciones en Flutter es necesario saber Dart, un lenguaje de programación también desarrollado por Google y escrito en C++.

Algunas de las características de Dart, el lenguaje de programación que utiliza Flutter, son:

- El código en dart se compila con una técnica conocida como *Ahead of Time (AOT)* o compilado anticipado tanto para arquitecturas X86 como ARM. Además Dart también permite ser compilado con la técnica *Just in Time (JIT)* o en tiempo real. Esto es útil durante la fase de desarrollo ya que permite recompilar en tiempo de ejecución solo la parte

del código que ha cambiado, lo cual es significativamente más rápido que la alternativa de Android, donde para cada cambio que se hace hay que volver a compilar todo el código.

- Dart es un lenguaje orientado a objetos, cuenta con recolector de basura, es un lenguaje tipado y además permite la inferencia de tipos.
- Dart es uni hilo y no permite la Multitarea apropiativa, lo cual hace que la recolección de basura sea extremadamenta rápida y no se requiera de bloqueos de memoria.

Flutter utiliza un motor (Flutter Engine) que compila el código y renderiza pixel a pixel una interfaz de usuario para la plataforma de destino. Esto es lo que permite que el mismo código se pueda compilar a distintas plataformas sin necesidad de hacer cambios. La desventaja de esta aproximación es que es más lenta que las tecnologías nativas de cada plataforma, por lo que aplicaciones que requieran de mucho uso de cpu o gpu como algunos videojuegos no es aconsejable desarrollarlos usando Flutter.

Flutter proporciona a los desarrolladores un conjunto de widgets (artilugios) configurables que en función de la plataforma en la que se estén ejecutando se mostrarán con diferente apariencia para adecuarse a las interfaces a las que está acostumbrado el usuario. Por ejemplo, los mensajes de error tienen distinta apariencia en iOS que en Android, aunque el contenido de los mismos sea el mismo. Flutter se encarga de modificar la apariencia del widget que contiene el mensaje de error de forma que el desarrollador obtiene funcionalidades sin esfuerzo adicional.

En Flutter, todo son artilugios o widgets. Podemos considerar a los widgets como bloques de código reusables que describen la apariencia de la interfaz de usuario (UI). Un widget puede ser un botón, un texto, etc. Flutter construye un árbol de widgets donde organiza las dependencias entre widgets. Existe un widget raíz que contiene al resto. Existen dos tipos de widgets:

- **Widgets sin estado**: Se trata de widgets que no guardan ningún tipo de estado, es decir, no tienen valores que puedan cambiar. Por ejemplo, un widget tipo Image que solo muestra una imagen en pantalla.
- **Widgets con estado**: Se trata de widgets que realizan el seguimiento de los cambios que se producen y pueden modifcar la interfaz en función de estos cambios. Por ejemplo, un botón que cambia su apariencia al ser pulsado es un ejemplo de Stateful widget.

La dependencia de widgets en la aplicación móvil desarrollada puede verse en la figura 7.1.

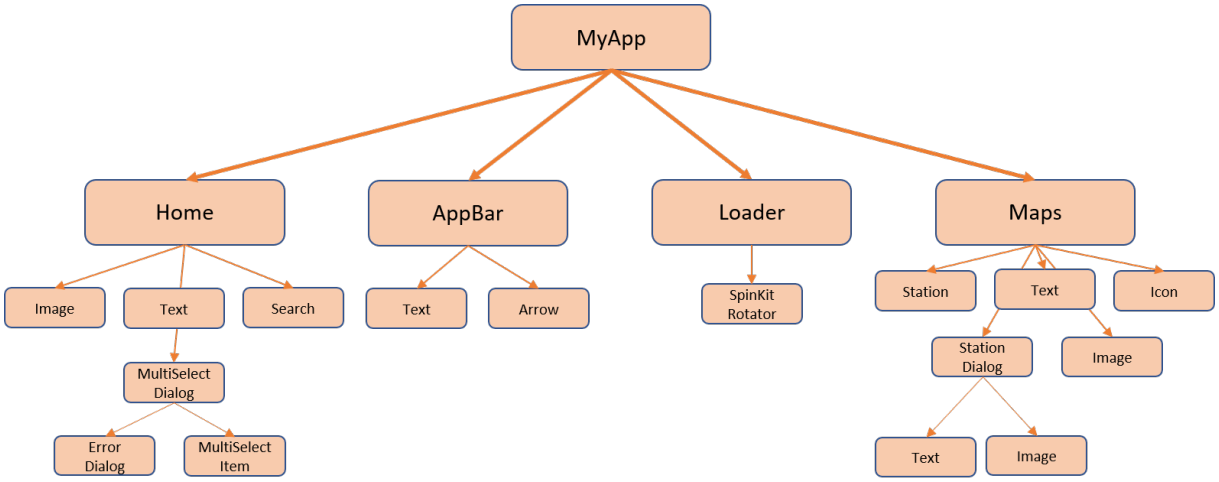


Figura 7.1: Árbol de dependencias de widgets

En la figura se observa que del nodo raíz cuelgan 4 widgets, correspondientes a las 3 pantallas principales de la aplicación y una barra de tareas común. Cada uno de estos widgets a su vez tiene nodos hijos que son los widgets que se utilizan para implementar cada una de las pantallas.

Capítulo 8

Pruebas

En todo proyecto de software de envergadura, la implementación de pruebas es imprescindible y es por ello que en los últimos años, la industria de la ingeniería de software ha ido cambiando sus hábitos hacia un desarrollo más centrado en las pruebas y en su automatización.

Con la gran complejidad que pueden alcanzar las aplicaciones actuales no tiene sentido realizar pruebas de forma manual pues, con el tiempo, se acaban convirtiendo en una carga y se acaban abandonando. Debido a han surgido multitud de *frameworks* y librerías que facilitan y automatizan el proceso de pruebas o *testing*. En particular, en este proyecto se han implementado pruebas utilizando la librería *pytest*[1] para la aplicación web y la librería *test*[2] de Dart para las pruebas de la aplicación móvil.

Tradicionalmente se distingue entre pruebas de caja blanca y pruebas de caja negra. Las primeras estudian el comportamiento de una entrada y sus transformaciones hasta que el sistema devuelve una salida, mientras que las pruebas de caja negra solo validan la salida obtenida para una entrada determinada, sin tener en cuenta el funcionamiento interno del sistema. Habitualmente las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas para comprobar que la integración entre estos subsistemas es correcta.

A continuación se presentan las pruebas de caja negra que se han utilizado para comprobar el buen funcionamiento del sistema.

[1] https://docs.pytest.org/en/stable/

[2] https://pub.dev/packages/test

PR-01	Descarga de ficheros Excel
Descripción	Cuando un fichero no se descarga correctamente, inmediatamente se vuelve a reintentar la descarga hasta que se realice correctamente
Prerrequisito	Hay conexión a internet
Datos de entrada	Dirección url del archivo Excel a descargar
Resultado previsto	El fichero que se ha descargado con errores es eliminado y sustituido por uno sin errores
Resultado real	Correcto

Tabla 8.1: PR-01 Descarga de ficheros Excel

PR-02	Eliminación de caracteres malintencionados
Descripción	Si el fichero Excel incluye texto que pueda contener inyección sql se debe detectar y eliminar
Prerrequisito	Se ha descargado un fichero Excel correctamente
Datos de entrada	Contenido del fichero Excel
Resultado previsto	Eliminación de caracteres que puedan suponer inyección sql
Resultado real	Correcto

Tabla 8.2: PR-02 Eliminación de caracteres malintencionados

PR-03	Actualización de la base de datos cada hora
Descripción	La información en la base de datos debe actualizarse cada hora con cada nuevo fichero que se descarga
Prerrequisito	Se ha descargado y procesado un fichero Excel correctamente
Datos de entrada	Contenido del fichero Excel ya procesado y sin errores
Resultado previsto	La información en la base de datos se ha actualizado
Resultado real	Correcto

Tabla 8.3: PR-03 Actualización de la base de datos cada hora

PR-04	Correcto funcionamiento de la API REST
Descripción	La API REST devuelve la información de precios y gasolineras actualizadas cuando recibe peticiones
Prerrequisito	Se ha descargado y procesado un fichero Excel correctamente
Datos de entrada	Contenido del fichero Excel ya procesado y sin errores
Resultado previsto	La información devuelta en la llamada API REST tiene el formato correcto y está actualizada.
Resultado real	Correcto

Tabla 8.4: PR-04 Correcto funcionamiento de la API REST

PR-05	Renderización página de inicio app móvil
Descripción	Comprobar la correcta renderización de la página de inicio
Prerrequisito	Se ha abierto la aplicación
Datos de entrada	Ninguno
Resultado previsto	La página de inicio se ha renderizado correctamente y en un tiempo menor a dos segundos
Resultado real	Correcto

Tabla 8.5: PR-05 Renderización página de inicio app móvil

PR-06	Validación de formulario de búsqueda
Descripción	Validar el comportamiento del formulario de búsqueda. Comprueba que solo una opción se puede seleccionar simultáneamente
Prerrequisito	La pantalla de inicio se ha renderizado
Datos de entrada	Selección de las opciones del formulario
Resultado previsto	Si se ha seleccionado más de una opción se muestra un mensaje de error
Resultado real	Correcto

Tabla 8.6: PR-06 Validación de formulario de búsqueda

PR-07	Renderización correcta del mapa con gasolineras
Descripción	Comprobar la correcta renderización del mapa sobre el cual se sobreponen los iconos de gasolineras en distintos colores
Prerrequisito	Se ha seleccionado un carburante y la opción de buscar
Datos de entrada	Carburante seleccionado
Resultado previsto	La pantalla se ha renderizado correctamente y las gasolineras se pueden visualizar de manera adecuada
Resultado real	Correcto

Tabla 8.7: PR-07 Renderización correcta del mapa con gasolineras

PR-08	Precio de carburantes
Descripción	Validar que el precio de carburantes es correcto y está actualizado
Prerrequisito	El mapa se ha renderizado correctamente
Datos de entrada	Gasolinera seleccionada
Resultado previsto	El precio del carburante está actualizado y se muestra de forma adecuada al usuario desde la aplicación móvil
Resultado real	Correcto

Tabla 8.8: PR-08 Precio de carburantes

PR-09	Vinculación correcta con google maps para buscar rutas
Descripción	Validar el comportamiento del sistema cuando se selecciona ir a una gasolinera
Prerrequisito	El mapa con gasolineras se ha renderizado correctamente
Datos de entrada	Gasolinera seleccionada (incluye sus coordenadas)
Resultado previsto	La aplicación *Google Maps* se abre con la opción de ir hasta la ubicación seleccionada cuando el usuario lo solicita
Resultado real	Correcto

Tabla 8.9: PR-09 Vinculación correcta con google maps

Capítulo 9

Documentación

9.1. Manual de usuario

La interfaz gráfica de la aplicación móvil se ha diseñado para poder ser manejada sin necesidad de consultar ningún manual ni de requerir ninguna formación previa. De cualquier forma, en este apartado se incluyen los conceptos básicos necesarios para utilizar la aplicación y la API-REST.

9.1.1. API-REST

La API-REST sigue el estándar de open-api [1] y la interfaz se ha generado de manera automática siguiendo dicho estándar con la ayuda de las librería *flask-restful*, de manera que cualquier manual que explique la especificación es válido para usar la interfaz. Por tanto se recomienda leer el manual citado para aprender a utilizar el servicio. Los *endpoints* del servicio son:

- **/stations:** El punto de acceso *stations* se divide en los siguientes :
  - / : Devuelve toda la información de un conjunto de gasolineras. Permite filtrar por los campos siguientes:
    - id - opcional - admite números enteros
    - province - opcional - admite cadenas de texto
    - margin_type - opcional - admite cadenas de texto
    - sell_type - opcional - admite cadenas de texto
    - company - opcional - admite cadenas de texto
    - oil_type - opcional - admite cadenas de texto

[1] https://swagger.io/specification/

- **/companies** : Devuelve un listado de compañías. Los elementos de esta lista se usan para filtrar en otras consultas. Permite filtrar pos los campos siguientes:
  - province - opcional - admite cadenas de texto
  - oil_type - opcional - admite cadenas de texto
  - sell_type - opcional - admite cadenas de texto
  - margin_type - opcional - admite cadenas de texto
- **/margintypes** : Devuelve una lista con los tipos de márgenes en los que puede estar una gasolinera (derecha, izquierda o indeterminado). Los elementos de esta lista se usan para filtrar en otras consultas.
- **/oiltypes** : Devuelve un listado con los tipos de carburantes que que se comercializan en España.
- **/provinces** : Devuelve un listado con las siglas de todas las provincias de España. Los elementos de esta lista se usan para filtrar en otras consultas.
- **/remissiontypes** : Devuelve un listado con los tipos de proveedores con los que trabajan las gasolineras (Operador Mayorista y Operador Minorista). Los elementos de esta lista se usan para filtrar en otras consultas.
- **/selltypes** : Devuelve un listado con el tipo de venta posible de una gasolinera (venta al público, restringido a cooperativas o *partners* o ambos). Los elementos de esta lista se usan para filtrar en otras consultas.
- **/servicetypes** : Devuelve un listado con los tipos de venta que puede tener una gasolinera (servicio asistido por personal, autoservicio con personal en la instalación y autoservicio sin personal en la instalación). Los elementos de esta lista se usan para filtrar en otras consultas.

- **prices:** El punto de acceso prices se divide en los siguientes:

  - **/now** : Devuelve precios en tiempo real. El formato de respuesta es un array de json donde cada json contiene precio, id de gasolinera y tipo de carburante. Permite filtrar por los campos siguientes:
    - id - opcional - admite números enteros
    - province - opcional - admite cadenas de texto
    - oil_type - opcional - admite cadenas de texto
    - margin_type - opcional - admite cadenas de texto
    - sell_type - opcional - admite cadenas de texto
    - company - opcional - admite cadenas de texto

- /**series** : Devuelve una serie de datos que contienen el histórico de precios de un conjunto de gasolineras. Permite filtrar por los campos siguientes:
  - id - opcional - admite números enteros
  - province - opcional - admite cadenas de texto
  - oil_type - requerido - admite cadenas de texto
  - margin_type - opcional - admite cadenas de texto
  - sell_type - opcional - admite cadenas de texto
  - company - opcional - admite cadenas de texto

- **flutterApp:** Punto de acceso usado por la aplicación móvil. Tiene los mismos campos y mismas posibilidades que /**prices** y /**stations**/ pero devuelve los datos agrupados más eficientemente en cuanto a memoria se refiere.

En todos los casos si se introduce información incorrecta el servicio devuelve un mensaje informativo sugiriendo cómo ha de introducirse la información de manera correcta. La interfaz principal con la que interactuar se observa en la figura 9.1.

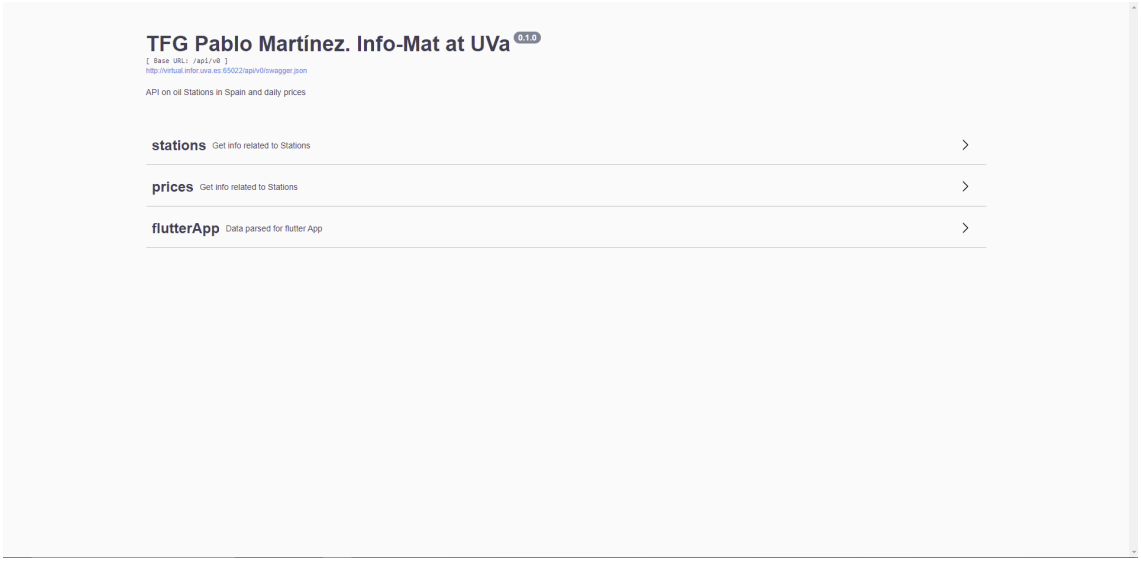


Figura 9.1: Servicio API-REST - Pantalla principal

Esta es la interfaz principal de la API con la que los usuarios pueden interactuar. Evidentemente para usar la API no hace falta usar la interfaz, pero es una buena práctica desarrollar una interfaz gráfica que permita a los usuarios aprender a utilizarla para luego hacer consultas desde un entorno de desarrollo. En las figuras 9.2 y 9.3 se pueden observar algunas interacciones con el servicio.

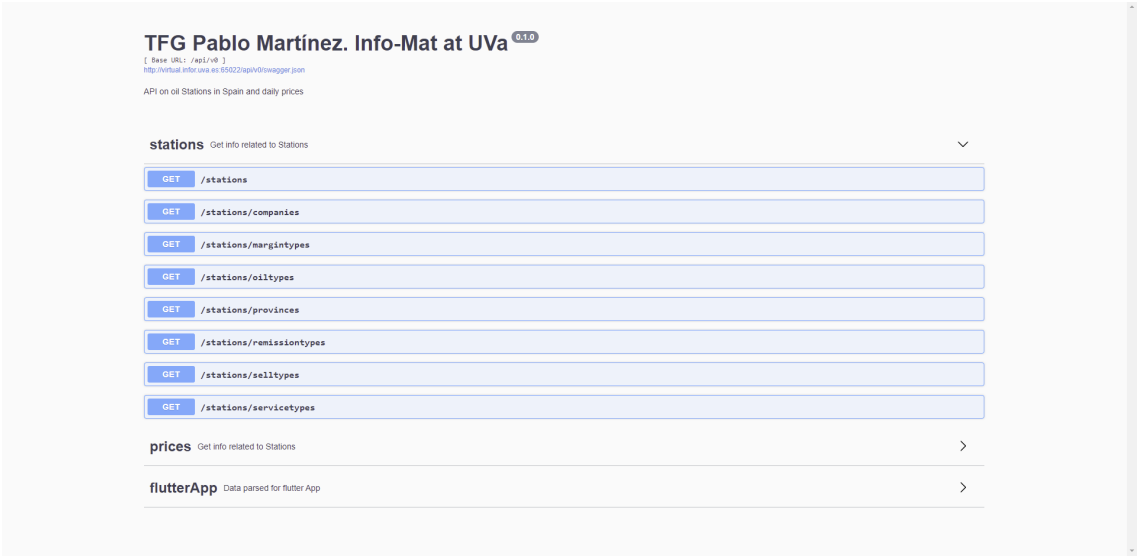


Figura 9.2: Servicio API-REST - stations

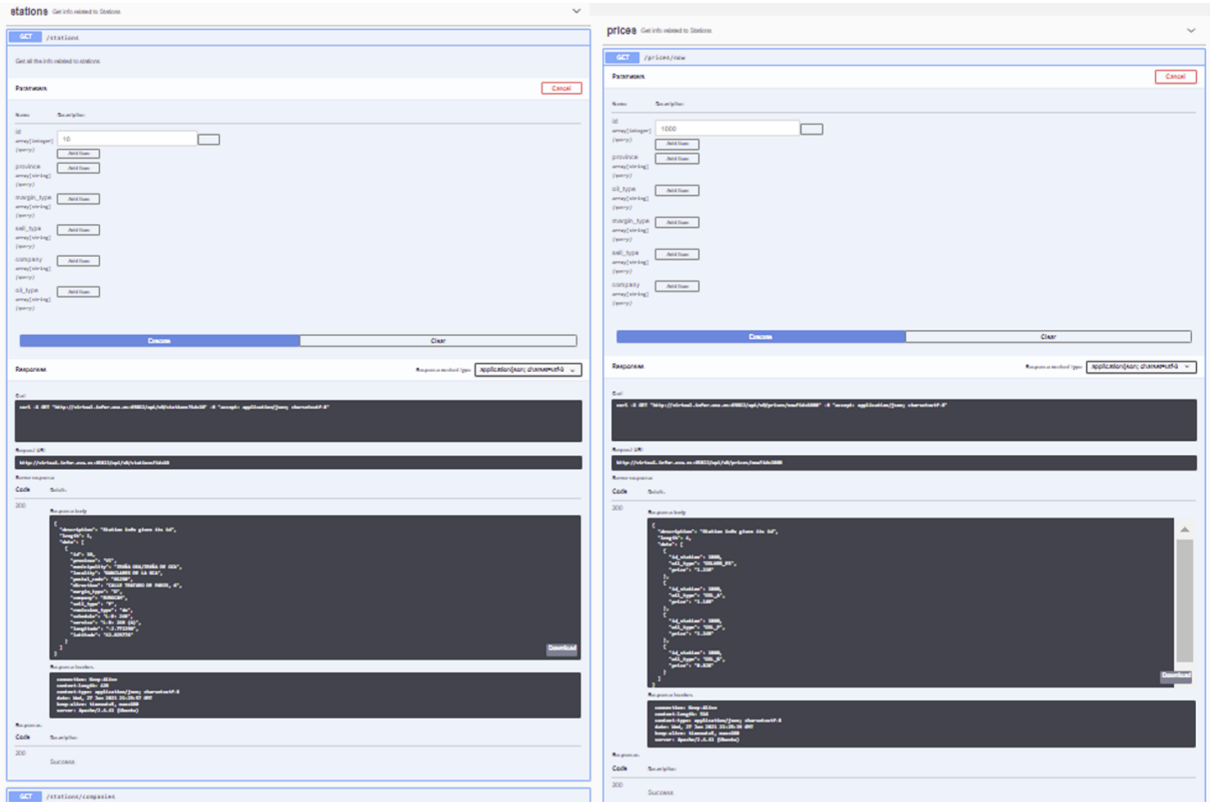


Figura 9.3: Servicio API-REST - stations and prices consulta

9.1.2. Aplicación Móvil

En cuanto a la aplicación móvil se refiere, la interacción es muy sencilla. Para buscar gasolineras se requiere en primer lugar seleccionar un carburante y seleccionar la opción **buscar**. Si se selecciona la opción de buscar sin previamente haber escogido un carburante se muestra un mensaje de error. En la figura 9.5 se muestra cómo es la interacción para este caso de uso.

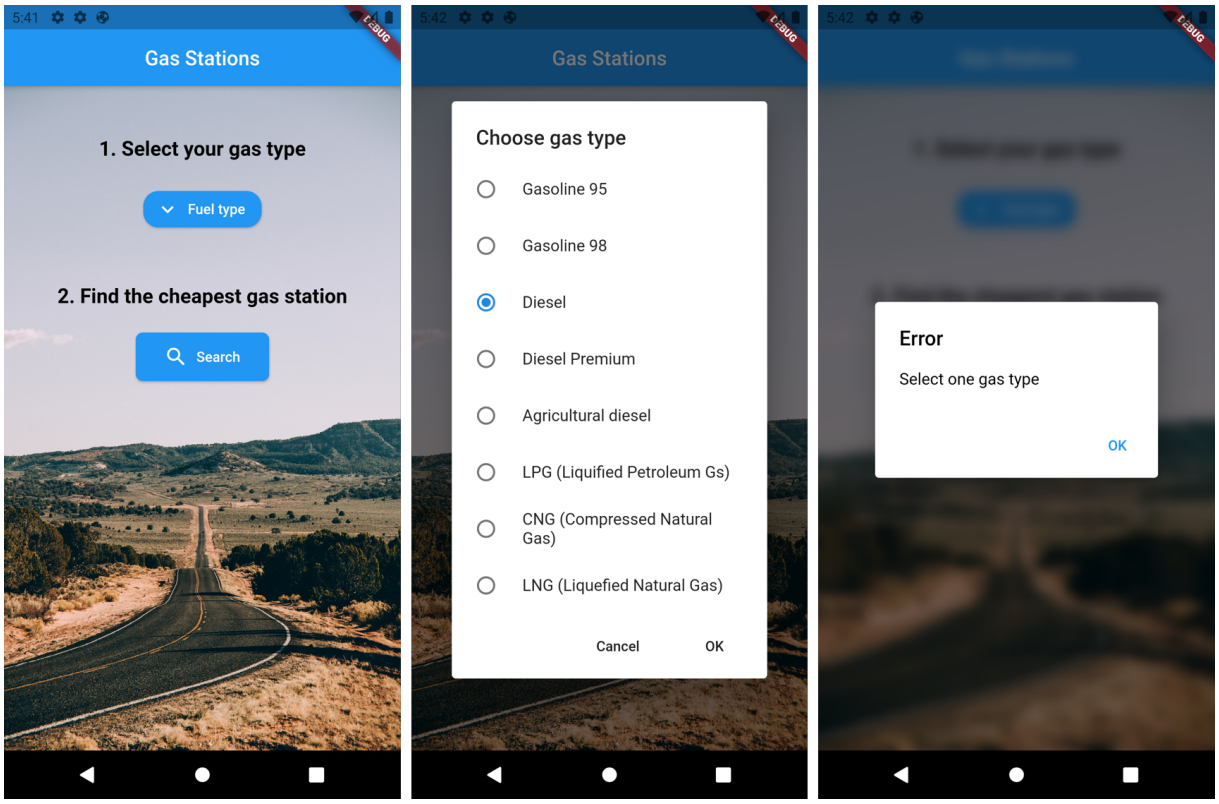


Figura 9.4: Aplicación móvil - buscar gasolineras

Una vez seleccionada la opción de **buscar** se presenta al usuario un mapa con iconos de gasolineras sobrepuestos al mapa. Estos iconos son de cuatro colores posibles:

- **Rojo** : Indica que el precio de esa gasolinera es caro en comparación con la competencia.
- **Negro** : Indica que el precio de esa gasolinera es neutro, presenta un precio medio.
- **Verde** : Indica que el precio de esa gasolinera es económico en comparación con la competencia.
- **Azul** : Una gasolinera azul es una agrupación de gasolineras. Es necesario hacer *zoom* para poder visualizar al conjunto de gasolineras agrupadas en ese conjunto.

A continuación se pueden ver imágenes de la interfaz (Figura 9.5). Las imágenes son capturas hechas con gasolineras reales en la ciudad de Valladolid.

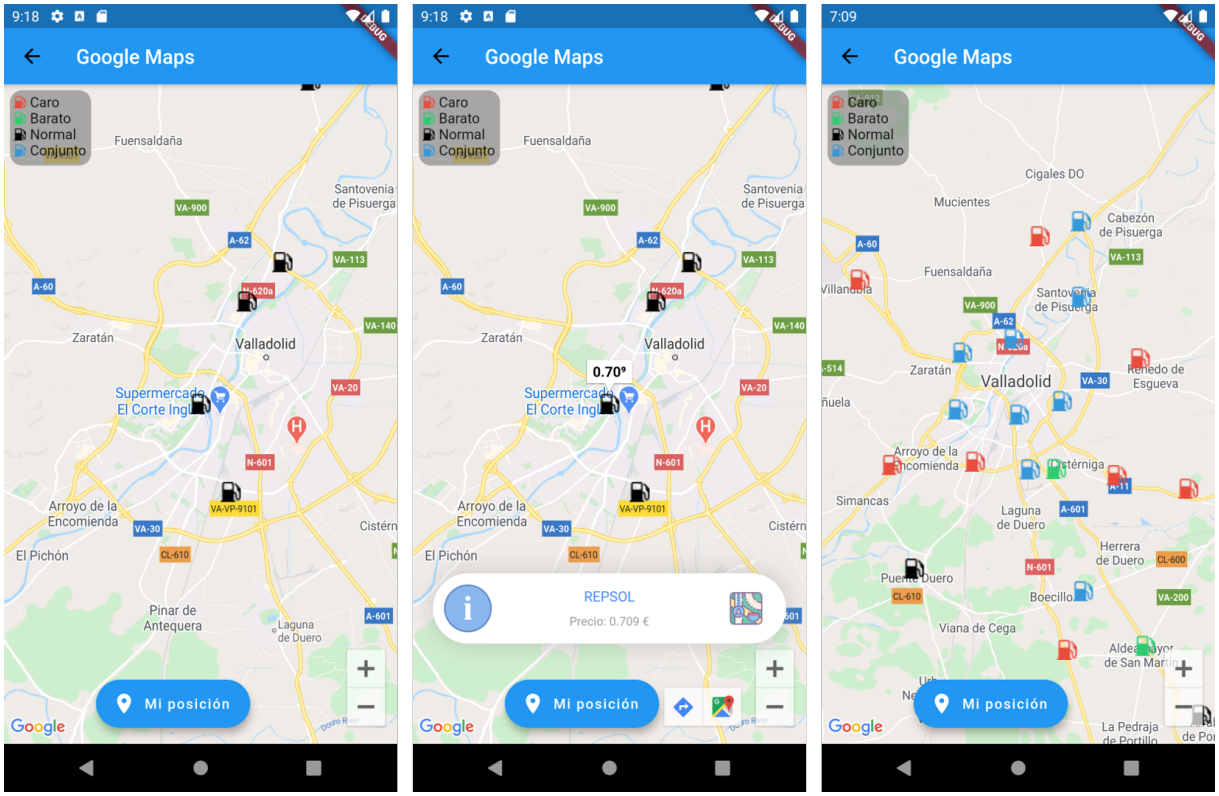


FIGURA 9.5: APLICACIÓN MÓVIL - GASOLINERAS EN VALLADOLID

9.2. Manual de despliegue

Para lanzar este proyecto se va a suponer que se tiene acceso a una máquina virtual Ubuntu con la versión 20.04 LTS o 18.04.5 LTS. Idealmente esta máquina deberá contar con al menos un núcleo y 2GB de RAM. Una vez que se tiene acceso a la máquina virtual, las configuraciones ha realizar son:

- **Configuración inicial:** Inicialmente se han de ejecutar una serie de comandos: actualización del gestor de paquetes, instalación de pip, git, venv, apache2 y mod_wsgi (módulo que permite enrutar el tráfico desde apache2 a la aplicación flask).

```
1 sudo apt-get update
2 sudo apt -y upgrade
3
4 # Se crea un nuevo usuario
5 adduser pablo
```

```
usermod -aG sudo pablo
# Se crean las claves ssh para acceder a la máquina virtual
mkdir /home/pablo/.shh
nano /home/pablo/.shh/authorized_keys # Put public key here. Must be in one line
sudo service ssh restart
# Se cambia la zona horaria
sudo timedatectl set-timezone Europe/Madrid
# Se instala git
```

- **Base de datos:** Como base de datos se ha utilizado mariadb, la cual se puede instalar de manera sencilla. También será necesario instalar una librería en python para usar mariadb:

```
sudo apt-get install apt-transport-https
curl -LsS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
sudo apt-get install mariadb-server mariadb-client

# python client
sudo apt-get update -y
sudo apt-get install -y libmariadb-dev
sudo apt install python3-pip
pip3 install mariadb
```

Una vez instalada es necesario crear la base de datos y las tablas que se necesitan en el proyecto, para lo cual hay que acceder a la base de datos y crear una nueva base de datos a la que se llamará tfg:

```
sudo mysql -u root
CREATE DATABASE tfg CHARACTER SET UTF8MB4;
USE tfg;
```

A continuación se crea la tabla hourly_prices:

```
CREATE TABLE IF NOT EXISTS hourly_prices (
  id_station INT UNSIGNED NOT NULL,
  oil_type ENUM('GSLN95_E5', 'GSLN95_E10', 'GSLN95_E5_P', 'GSLN98_E5',
      'GSLN98_E10', 'GSL_A', 'GSL_P', 'GSL_B', 'GSL_C', 'BIOETANOL',
      'BIOALCOHOL', 'BIODIESEL', 'ESTER_METILICO', 'GLP', 'GNC', 'GNL',
      'H') NOT NULL,
  price DECIMAL(6,3) NULL
  )
CHARACTER SET = utf8mb4
COLLATE = utf8mb4_unicode_ci
ENGINE = Aria;
```

```
9
10 CREATE UNIQUE INDEX unique_index ON hourly_prices(id_station, oil_type,
       date);
11 CREATE INDEX dates_index ON hourly_prices(date);
12 CREATE INDEX prices_index ON hourly_prices(price);
```

También hay que crear la tabla stations:

```
1  # Tabla prices
2  CREATE TABLE IF NOT EXISTS prices (
3    id_station INT UNSIGNED NOT NULL,
4    date DATE NOT NULL,
5    oil_type ENUM('GSLN95_E5', 'GSLN95_E10', 'GSLN95_E5_P', 'GSLN98_E5',
         'GSLN98_E10', 'GSL_A', 'GSL_P', 'GSL_B', 'GSL_C', 'BIOETANOL',
         'BIOALCOHOL', 'BIODIESEL', 'ESTER_METILICO', 'GLP', 'GNC', 'GNL',
         'H') NOT NULL,
6    price DECIMAL(6,3) NULL
7    )
8  CHARACTER SET = utf8mb4
9  COLLATE = utf8mb4_unicode_ci
10 ENGINE = Aria;
11
12 # Tabla stations
13 CREATE TABLE IF NOT EXISTS stations (
14   id INT UNSIGNED NOT NULL AUTO_INCREMENT COMMENT 'Auto Increment
         Primary Key',
15   province
         ENUM('VI','AB','A','AL','O','AV','BA','PM','B','BU','CC','CA','S',
   'CS','CEUTA','CR','CO','C','CU','GI','GR','GU','SS','H','HU','J','LE',
17 'L','LU','M','MA','MELILLA','MU','NA','OR','P','GC','PO','LO','SA',
18 'TF','SG','SE','SO','T','TE','TO','V','VA','BI','ZA','Z') NULL COMMENT
         'Province in which the oil station is',
19   municipality VARCHAR(150) NULL COMMENT 'Municipality in which the
         oil_station is',
20   locality VARCHAR(150) NULL COMMENT 'Locality in which the oil_station
         is',
21   postal_code VARCHAR(15) NULL,
22   direction VARCHAR(150) NULL,
23   margin_type ENUM('D', 'I', 'N') NULL COMMENT '\'D\' means \'Derecha\'
         or \'right\' margin of road while \'I\' stands for \'izquierda\'
         or left. \'N\' stands for \'No Aplica\' or \'does not apply\'',
24   company VARCHAR(150) NULL COMMENT 'Big post or sign where usually the
         company\'s name which operates the oil station is written',
25   sell_type ENUM('P', 'R', 'A') NULL COMMENT 'There are two
         possibilities: \'P\' which stands for general public and \'R\'
         which stands for restrictied only for cooperatives or partners. A
```

```
        stands for ambos',
26    remission_type ENUM('OM', 'dm') NULL COMMENT '\'OM\' stands for From
        whom the information comes from. There are two possibilities.
        \'Operador Mayorista\' (wholesale operator) and \'dm\' which
        stands for \'distribuidor minorista\' (retail operator)',
27    schedule VARCHAR(1000) NULL COMMENT 'Opening hours and days.',
28    service VARCHAR(1000) NULL COMMENT 'P: Staff assisted service, A:
        Self-service by the client, with the presence of personnel at the
        facility, D: Self-service by the client, without the presence of
        personnel in the installation',
29    longitude DECIMAL(8,6) NULL COMMENT 'longitude stored as DECIMALs for
        searching purposes',
30    latitude DECIMAL(9,6) NULL COMMENT 'latitude as DECIMAL for searching
        purposes',
31    coordinates POINT NOT NULL DEFAULT POINT(0,0) COMMENT 'Physical
        coordinates in longitude and latitude in which the oil station is
        placed.',
32    previous_id INT UNSIGNED NULL DEFAULT NULL COMMENT 'Previous
        oil_Station id which was in the same coordinates before. Same
        phisical oilStation may change its managment, and for our
        purposes, that means it is a new oil Station',
33    creation_date DATE NULL COMMENT 'Creation date of oil Station',
34    end_date DATE NULL COMMENT 'Always NULL except if management changes',
35    PRIMARY KEY (`id`)
36    )
37  CHARACTER SET = utf8mb4
38  COLLATE = utf8mb4_unicode_ci
39  ENGINE=Aria;
40
41  CREATE INDEX company_index ON stations(company);
42  CREATE INDEX province_index ON stations(province);
```

Por último hay que crear la tabla prices:

```
1   CREATE TABLE IF NOT EXISTS prices (
2     id_station INT UNSIGNED NOT NULL,
3     date DATE NOT NULL,
4     oil_type ENUM('GSLN95_E5', 'GSLN95_E10', 'GSLN95_E5_P', 'GSLN98_E5',
        'GSLN98_E10', 'GSL_A', 'GSL_P', 'GSL_B', 'GSL_C', 'BIOETANOL',
        'BIOALCOHOL', 'BIODIESEL', 'ESTER_METILICO', 'GLP', 'GNC', 'GNL',
        'H') NOT NULL,
5     price DECIMAL(6,3) NULL
6     )
7   CHARACTER SET = utf8mb4
8   COLLATE = utf8mb4_unicode_ci
9   ENGINE = Aria;
```

```
10
11 CREATE UNIQUE INDEX unique_index ON prices (id_station, oil_type, date);
12 CREATE INDEX dates_index ON prices (date);
13 CREATE INDEX prices_index ON prices (price);
```

- **Apache:** El servidor de aplicaciones utilizado será Apache. Además lo vamos a habilitar tanto en el puerto 80 como en el 443:

```
1 # Install apache2
2 sudo apt install apache2
3
4 # Vemos las posibilidades de qué puertos abrir
5 sudo ufw app list
6
7 # Abrimos el puerto 80 y el 443
8 sudo ufw allow in "Apache"
9 sudo ufw allow in "Apache Secure"
```

- **Directorios:** Es necesario generar un conjunto de directorios donde se guardarán los ficheros que se descarguen desde el portal del Ministerio para la Transición Ecológica y el Reto Demográfico. Para generar la estructura basta con ejecutar los siguientes comandos:

```
1 cd
2 # Carpeta donde se guardarán los ficheros Excel descargados
3 mkdir tfg_data
4 # Carpeta donde se guardarán los scripts de python encargados de la
      descarga, limpieza e inserción de datos
5 mkdir python_projects
6
7 cd tfg_data
8 mkdir csv_insertados csv_sininsertar db_backup xls_procesados
      xls_sinprocesar
```

- **Github:** Será necesario descargarse los proyectos desde github a nuestra máquina virtual. Además vincularemos la cuenta de github con la máquina virtual para que cualquier *commit* que se realicé en local automáticamente se despliegue en producción en la máquina virtual:

```
1 # Create private-public ssh key in virtual machine
2 ssh-keygen -t rsa -b 4096 -C "pablompg@pm.me"
3
4 # Start the ssh-agent in the background.
5 eval "$(ssh-agent -s)"
6
7 # Add your SSH private key to the ssh-agent.
```

```
ssh-add ~/.ssh/id_rsa

# Show you public key and copy paste it to github
cat ~/.ssh/id_rsa.pub

# Add ssh keys to root user (in /var/www we will need sudo)
cp /home/pablo/.ssh/id_rsa /root/.ssh/id_rsa
cp /home/pablo/.ssh/id_rsa.pub /root/.ssh/id_rsa.pub

# test your local key works. You should be denied shell access by github after successfully authenticated
ssh -T git@github.com

# Using your favorite text editor, open up the file at ~/.ssh/config.
# If this file doesn't exist, you can create it by entering touch ~/.ssh/config
nano ~/.ssh/config

# Enter the following text into the file
Host 141.95.1.170
  ForwardAgent yes

# Let's download our repos
cd python_proyects
git clone git@github.com:Pablompg/tfg_intodb.git

# Add ssh keys to root user
cp /home/pablo/.ssh/id_rsa /root/.ssh/id_rsa
cp /home/pablo/.ssh/id_rsa.pub /root/.ssh/id_rsa.pub

cd /var/www
git clone git@github.com:Pablompg/tfg_flaskapp.git

# Añadir el fichero de configuración a ambos proyectos
cd python_projects/tfg_intodb
nano .env

sudo chmod g+w * in /var/www/tfg_flaskapp
```

- **Carpeta tfg_intodb:** Una vez están generados dichos directorios se copia la carpeta *tfg_intodb* dentro de */python_script*. Una vez copiada la carpeta se genera el entorno virtual y se instalan las librerías necesarias para poder ejecutar el proyecto con los siguientes comandos:

```
sudo apt-get install python3-venv
```

```
2
3  cd /var/Www/tfg_flaskapp
4  sudo pip3 install -r requirements.txt
5
6  cd /home/pablo/python_projects/tfg_intodb/
7  python3 -m venv venv
8
9  # Lo activamos
10 source venv/bin/activate
11 pip3 install -r requirements.txt
12
13 # Para generar el fichero de requirements.txt
14 pip3 freeze > requirements.txt
```

- **Mod_wsgi:** Instalámos el módulo *mod_wsgi* [10] que permite alojar y desplegar aplicaciones python en Apache:

```
1  # Instalamos mod_wsgi
2  sudo apt-get install libapache2-mod-wsgi-py3
3  sudo apt-get install apache2-dev
4
5  # Check if module is loaded
6  apache2ctl -t -D DUMP_MODULES
7
8  # If module not loaded:
9  # We enable it, we restart apache and we check again
10 sudo a2enmod wsgi
11 sudo systemctl restart apache2
```

- **Host Virtual:** Se configura un host virtual para la aplicación web:

```
1  # Creamos un nuevo archivo de configuración
2  sudo nano /etc/apache2/sites-available/tfg_flaskapp.conf
3
4  # Lo editamos y guardamos
5  #WSGIPythonHome "/var/www/tfg_flaskapp/tfg_flaskapp/venv/bin"
6  #WSGIPythonPath
       "/var/www/tfg_flaskapp/tfg_flaskapp/venv/lib/python3.8/site-packages"
7
8  <VirtualHost *:80>
9                  WSGIScriptAlias /
                       /var/www/tfg_flaskapp/tfg_flaskapp_wsgi.py
10                 <Directory /var/www/tfg_flaskapp/>
11                         Order allow,deny
12                         Allow from all
```

```
                </Directory>
                ErrorLog ${APACHE_LOG_DIR}/error.log
                LogLevel warn
                CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

# Enable the virtual host with the following command:
sudo a2ensite tfg_flaskapp
sudo systemctl reload apache2

# Para deshabilitar un virtual host
sudo a2dissite 000-default.conf
sudo a2dissite default-ssl.conf

# Para verificar configuración de apache
sudo apachectl configtest
```

- **Fichero wsgi:** Apache usa el fichero *.wsgi* para servir la aplicación de Flask:

```
cd /var/www/tfg_flaskapp
vim tfg_flaskapp_wsgi.py

# Editamos el fichero y deberá ser algo así
import sys
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0, "/var/www/tfg_flaskapp/")

from app import app as application
application.secret_key = '1234'

# Ahora reiniciamos apache y todo debería funcionar perfectamente
sudo service apache2 restart

# Give tfg_flaskapp owner -> pablo
sudo mysql -u root
GRANT ALL privileges ON tfg.* TO pablo@localhost IDENTIFIED BY 'router';
```

- **Frontab:** Utilizaremos crontab para descargarnos los ficheros Excel cada hora:

```
# crontab -e
0 9-21 * * * echo 'source
    /home/pablo/python_projects/tfg_intodb/bin/venv/activate; python3
    /home/pablo/python_projects/tfg_intodb/downloadxls_tocsvtodb.py' |
    /bin/bash
```

```
0 * * * * echo 'source
    /home/pablo/python_projects/tfg_intodb/venv/bin/activate; python3
    /home/pablo/python_projects/tfg_intodb/downloadxls_tocsvtodb_hourly.py'
    | /bin/bash
```

- **Permisos:** Garantizamos permisos de lectura a la aplicación web:

```
#Give write permissions to files
sudo chmod g+w * in /var/www/tfg_flaskapp
```

- **Certbot:** Utilizaremos certbot [11] para instalarnos un certificado ssl:

```
# install snap
sudo snap install core
sudo snap refresh core

#install certbot
sudo snap install --classic certbot

# Prepare the Certbot command
sudo ln -s /snap/bin/certbot /usr/bin/certbot

# Get a certificate and have Certbot edit your Apache configuration
    automatically to serve it, turning on HTTPS access in a single step
sudo certbot --apache
```

Capítulo 10

Conclusiones y trabajo futuro

Después de la finalización de este TFG, podemos concluir que se han alcanzado la mayor parte de los objetivos planteados al inicio del mismo. Se ha creado un sistema capaz de descargar, procesar e insertar de manera estructurada los datos de los precios de estaciones de servicio en España en una base de datos. También ha sido posible crear una aplicación móvil que permita visualizar dichos datos.

Junto a estos dos sistemas se ha desarrollado una API que sirve los datos a través de un servicio REST y es utilizada por la aplicación móvil. Esta API sigue los estándares modernos y es fácil de utilizar. Además permite añadir una capa de abstracción sobre los datos, lo cual posibilita que sean utilizados por cualquier otro sistema. Cabe señalar que no existe ninguna otra API pública que permita consultar los datos actualizados de precios en España.

El hecho de que el 21 de Enero de 2021 Google anunciara que iba a comenzar a ofrecer los precios de carburantes a través de la aplicación *Google Maps* [1] hace pensar que el tema escogido para este TFG era muy acertado y la demanda por esta información por parte de los usuarios es real.

Los conocimientos adquiridos al desarrollar este proyecto han sido muy amplios y han permitido profundizar en muchas de las áreas de la ingeniería informática en las que a lo largo del grado no había podido ahondar como el desarrollo de aplicaciones móviles o servicios REST.

Como trabajo futuro se plantea la posibilidad de hacer disponible la aplicación también para los usuarios iOS, así como añadir nuevas funcionalidades a la aplicación móvil. Algunas de estas funcionalidades podrían ser permitir filtrar por compañía o región, así como añadir un inicio de

[1] https://www.xataka.com/aplicaciones/google-maps-muestra-precio-gasolina-gasolineras-espanolas-asi-funciona (19-01-2021)

sesión obligatorio. También se podría plantear la posibilidad de añadir anuncios para monetizar el trabajo realizado o añadir funcionalidades premium que requieran de previo pago.

En cuanto a la escalabilidad, si el proyecto resulta exitoso y el tráfico se multiplica varios órdenes de magnitud, algunas de las tareas que se tendrá que hacer será desplegar la base de datos en una máquina virtual independiente a la aplicación web y cambiar el servidor *Apache* por *Nginx*, que es más avanzado tecnológicamente y permite varios órdenes de magnitud más en cuanto a conexiones simultáneas se refiere. Por otro lado se podría contratar una CDN (Content Delivery Network) que se actualizase cada hora para servir los datos desde la CDN y reducir el tráfico en la base de datos.

También sería interesante descargarse los datos de precios de otros países para así llegar a más usuarios. Como la internacionalización ya está implementada en la aplicación móvil, esta integración sería sencilla y bastaría con desarrollar otro procesos de descarga y filtrado de datos para cada proveedor de datos.

A la postre, cuando se finalice el TFG del Grado en Matemáticas, si los algoritmos de predicción desarrollados dan buenos resultados, se podría añadir a la aplicación un nuevo caso de uso que fuese visualizar los precios previstos para los próximos días. Esto permitiría a los usuarios decidir si repostar un día o esperar a los días siguientes.

Por último, la aplicación va a seguir funcionando aun cuando el TFG sea presentado, pues todos los servicios se han migrado a una máquina virtual independiente de la Universidad de Valladolid con el fin de que amigos y familiares sigan usando la aplicación en el día a día.

Referencias bibliográficas

[1] Instituto Nacional de Estadística. *Encuesta de Presupuestos Familiares 2019*. 2019. URL: https://www.ine.es/prensa/epf_2019.pdf (visitado 01-05-2021).

[2] Red Eléctrica de España. *Demanda Eléctrica y Actividad Económica*. 2019. URL: https://www.ree.es/es/datos/publicaciones/analisis-informes-demanda-electrica (visitado 01-05-2021).

[3] Corporación De Reservas Estrategicas De Productos Petrolíferos (CORES). *Consumo de productos petrolíferos 2019*. 2019. URL: https://www.cores.es/es/estadisticas (visitado 01-05-2021).

[4] Ministerio para la Transición Ecológica y el Reto Demográfico. *Geoportal*. URL: https://geoportalgasolineras.es (visitado 01-05-2021).

[5] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Financial Times Prentice Hall, 2000.

[6] SmartBear Software. *OpenAPI Specification*. 2021. URL: https://swagger.io/resources/open-api/ (visitado 10-08-2020).

[7] Ryan Horn Frank Stratton Kevin Burke Kyle Conroy y Guillaume Binet. *Flask-RESTful*. 2020. URL: https://flask-restful.readthedocs.io/en/latest/ (visitado 27-09-2020).

[8] Switzerland. International Organization for Standardization Geneva. *ISO 3166-2:ES*. 2021. URL: https://www.iso.org/obp/ui/#iso:code:3166:ES (visitado 28-09-2020).

[9] Kevin Burke y col. *Flask-RESTful: extension for Flask that adds support for quickly building REST APIs*. 2021. URL: https://flask-restful.readthedocs.io/en/latest/ (visitado 12-09-2020).

[10] Graham Dumpleton. *mod-wsgi*. 2020. URL: https://pypi.org/project/mod-wsgi/ (visitado 12-10-2020).

[11] Electronic Frontier Foundation. *certbot*. 2020. URL: https://certbot.eff.org/ (visitado 19-10-2020).

[12] flutter dev. *Flutter Framework*. 2020. URL: https://www.ine.es/prensa/epf_2019.pdf (visitado 01-05-2021).

[13] MariaDB Foundation. *MariDB python client*. 2021. URL: https://mariadb.com/resources/blog/how-to-connect-python-programs-to-mariadb/ (visitado 25-07-2020).

[14] MariaDB Foundation. *MariDB documentation*. 2021. URL: https://mariadb.org/documentation/ (visitado 02-08-2020).

[15] Tom Christie. *Flask api documentation*. 2021. URL: https://www.flaskapi.org (visitado 12-08-2020).

[16] Real Python. *Documenting Python Code: A complete guide*. 2021. URL: https://realpython.com/documenting-python-code/ (visitado 01-09-2020).

[17] flutter dev. *Google Maps for Flutter: Google Maps as a widget*. 2021. URL: https://pub.dev/packages/google_maps_flutter (visitado 15-11-2020).

[18] Pallets. *Flask web development framework: documentation*. 2021. URL: https://flask.palletsprojects.com/en/1.1.x/ (visitado 07-09-2020).

[19] Baptise Pillon. *Flutter Cluster Manager for Google Maps*. 2021. URL: https://pub.dev/packages/google_maps_cluster_manager (visitado 02-11-2020).

[20] Hyndman R.J. y Athanasopoulos G. *Forecasting: principles and practice*. 2018. URL: OTexts.com/fpp2 (visitado 15-01-2021).

[21] Daniel Peña. *Análisis de series temporales*. Alianza, 2005.

[22] Carmen Rodríguez Morilla. *Análisis de series temporales*. La muralla, 2000.

[23] Switzerland. International Organization for Standardization Geneva. *ISO 3166-2:ES*. 2021. URL: https://www.iso.org/obp/ui/#iso:code:3166:ES (visitado 28-09-2020).

[24] Bootstrap team. *Bootstrap*. 2020. URL: https://bootstrap.com/ (visitado 05-09-2020).

Apéndices

Apéndice A

Contenido adjunto

Junto a este documento, en el directorio de *OneDrive* proporcionado por la Escuela de Ingeniería Informática de Segovia para el depósito de los ficheros complementarios del proyecto se incluye:

- `tfg_intodb` Carpeta que contiene los ficheros del proyecto que descarga, procesa e inserta los ficheros Excel en una base de datos.
- `tfg_flaskapp` Carpeta que contiene los ficheros de la aplicación web y por tanto de la API-REST.
- `tfg_flutterapp` Carpeta que contiene los ficheros de la aplicación móvil.
- `Documento escrito` Carpeta que contiene este documento en versión pdf.

Apéndice B

Instalación y versiones

B.1. Versiones

Toda las implementaciones del proyecto se han llevado a cabo utilizando *Python* como lenguaje de programación en su versión 3.8 y Dart en su versión 2.10. En cuanto a las librerías requeridas, a continuación se incluyen las versiones que han sido utilizadas durante el desarrollo del proyecto para las librerías y *frameworks* principales.

- Pandas 1.1.1
- Numpy 1.18.5
- Matplotlib 3.3.1
- dio 3.0.10
- google_maps_flutter 1.0.6
- geolocator 6.1.4
- easy_localization
- google_maps_cluster_manager 0.2.0
- flutter_spinkit 4.1.2
- flutter_launcher_icons 0.8.0
- simplejson 3.17.2
- mariadb 1.0.1
- Flask 1.1.2
- Flask-RESTful 0.3.8

B.2. Instalación

Para la instalación de la aplicación basta con descargarse la aplicación desde: https://gasolineras.pablompg.com/static/android/gasolineras.apk