UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Aplicación Android para la búsqueda de precios económicos de combustible y geolocalización de E.E.S.S.

MEMORIA PRESENTADA POR:

Jorge Jaén Tormo

GRADO DE INFORMÁTICA

Convocatoria de defensa: *septiembre 2017*

Agradecimientos

En primer lugar me gustaría **agradecer su dedicación, esfuerzo y buen hacer a todos los profesores** que me han acompañado a lo largo de mi vida académica. A los de esta última etapa en la universidad, pero también a los de mi anterior paso por ella, hace ya 14 años. Por no dejarme a nadie, y porque también es justo, agradecerle al resto de profesores no universitarios, que han hecho posible que llegara hasta aquí. En mayor o menor medida, todos han contribuido a que me formara tanto académica como personalmente.

Como no podía ser de otra forma, tengo que dar las **gracias a mi familia por estar siempre ahí, y ayudarme a conseguir mis objetivos**. A **mis padres a los que les debo todo**, **y a mis hermanos** y familiares más allegados. Debo mencionar a mi hermano Oscar, ya que al compartir profesión y pasión por la informática, me ha podido ayudar con sus consejos y largas charlas sobre asignaturas y trabajos.

Y especialmente, agradecer a mi **mujer y mi hijo**, su **comprensión** y **paciencia,** sobre todo en los momentos menos buenos. Gracias por haberme apoyado y entender que **el tiempo que nos han robado estos años de estudio, ha sido una inversión para los tres**.

Sin vosotros hubiera sido imposible.

1-Índice

1-Índice

1- Introducción

En los últimos años, los dispositivos móviles han cambiado la forma en la que nos conectamos a internet. Para quienes ya lo hacíamos a diario, bien por trabajo o por ocio, han facilitado la conexión casi permanente a un innumerable catálogo de servicios (cuya utilidad y necesidad no vamos a valorar aquí). Asimismo a quienes no utilizaban Internet de forma habitual, o no lo habían hecho nunca, los nuevos dispositivos inteligentes, los servicios que ofrecen y porque no decirlo, la presión social, les han "empujado" a hacerlo.

Según datos del INE (Instituto Nacional de Estadística) de finales de 2016, que se recogen en la encuesta sobre "*Equipamiento y uso de Tecnología de la Información y la Comunicación en los hogares"***:** dos de cada tres españoles de 16 a 74 años **usa Internet a diario en España** y ocho de cada diez se ha conectado en los tres últimos meses. En concreto, el número de "usuarios intensivos" roza los **23 millones**, lo que supone el **66,8% de la población** de 16 a 74 años. Además, más de 27,7 millones de personas han utilizado Internet en los tres últimos meses, lo que equivale al 80,6% de la población.

Sin duda el responsable de estos datos, es el **smartphone**. Así lo destaca el informe de la Fundación Telefónica "*La Sociedad de la Información en España*", que en su 16 edición, corona al móvil como el **principal dispositivo** a través del cual los españoles entran en **Internet**, con un 88,3% de usuarios, superando al ordenador como puerta de acceso favorita.

Los dispositivos móviles actuales, en concreto los **teléfonos inteligentes**, son algo más que una conexión a internet. La combinación de procesamiento, dispositivos de conectividad inalámbricos y sensores, hacen de ellos una **herramienta casi indispensable** para el actual día a día. Es una realidad que los dispositivos móviles han cambiado nuestros hábitos e incluso nuestras formas de relacionarnos, y **por todo ello** se ha escogido una **aplicación móvil** como **Trabajo Fin de Grado**.

Otra realidad de nuestros días, es que la ya casi olvidada por algunos crisis económica, ha provocado la pérdida de poder adquisitivo al grueso de la población. Esto nos obliga a agudizar el ingenio, para estar más atentos al coste de los gastos ineludibles, si se quiere disfrutar de los pequeños placeres que están a nuestro alcance, y poder vivir como dirían algunos "por encima de nuestras posibilidades". Por mi situación laboral, uno de los gastos a los que debo prestar atención, es el asociado al combustible que necesito para desplazarme todos los días a mi puesto de trabajo, a unos 50km de donde resido. De esta necesidad personal, surge la idea de realizar una **aplicación para buscar la mejor opción para llenar el depósito**.

2- Objetivo

El objetivo del trabajo es muy claro, y es dar la posibilidad a cualquier persona con un dispositivo móvil Android, de **buscar** de una **forma rápida y sencilla**, la **mejor opción** para **repostar combustible**.

Para ello se pretende desarrollar una aplicación Android, que permita buscar los precios más económicos de diferentes tipos de combustible, o bien la estación de servicio más cercana que disponga el tipo de combustible deseado. Los **datos** correspondientes a todas las estaciones de servicio de España, serán **obtenidos** del **Ministerio de Industria, Energía y Turismo**.

La idea es, además de **mostrar los resultados** en forma de tabla, utilizar las posibilidades de los dispositivos Android para **representar las estaciones de servicio en un mapa**. Asimismo se ha considerado importante aprovechar la posibilidad de ubicación de los dispositivos actuales, para poder **mostrar los datos en función de la posición actual,** y poder calcular distancias y rutas hasta las estaciones de servicio seleccionadas.

3- Back-end

4.1- Obtención de los datos

Como parece obvio, el **punto crítico** y fundamental para una aplicación como la descrita son los **datos**. Si los datos de las estaciones de servicio: nombre, ubicación (latitud, Longitud y localidad) o los correspondientes a los precios de los diferentes tipos de combustibles, no son correctos o no se encuentran debidamente actualizados, la aplicación no servirá de nada.

También parece claro, que **no está a nuestro alcance recabar toda esta información**. El Ministerio de Energía, Turismo y Agenda Digital, tiene publicado un portal con la información sobre los precios de los carburantes en todas las estaciones de servicio de España. El **Geoportal de Hidrocarburos** http://geoportalgasolineras.es/ será la **fuente** de la **información** empleada en la aplicación.

Como se indica en el propio portal en el apartado de preguntas frecuentes, en relación a ***¿Cómo se recogen los precios?*:**

*Todos los operadores al por mayor y todas las personas que lleven la gestión efectiva de un punto de venta de carburantes para suministro a vehículos **están obligados** por la orden ITC/2308/2007 a **enviar los precios** practicados en ese punto de venta todos los lunes y cada vez que cambien dichos precios.*

En este mismo apartado se detallan las formas de denunciar los posibles incumplimientos, y los tipos de incumplimientos que destaca el ministerio

Incumplimientos

Las instalaciones que incumplen su obligación de mandar precios dejan a su vez de aparecer en Internet. Las denuncias por posible incumplimiento de la ITC/2308/2007 se presentan en papel ante la Comisión Nacional de los Mercados y la Competencia, en su sede de C/Alcalá, 47, 28014 Madrid…

… Si de lo que se trata es de que la instalación manda al Ministerio unos precios, que son los que aparecen en la página web del mismo, pero cobra otros…

Esta obligación junto con las posibles consecuencias por los incumplimientos, permiten que en condiciones normales, y como norma general, la **información** que pone a disposición el **ministerio**, se encuentre debidamente **actualizada y sea fidedigna.**

La información publicada por el ministerio, está organizada en diferentes formatos y es accesible de forma diferente dependiendo de la naturaleza y tipo de datos. Para el tipo de aplicación desarrollada, el **acceso a los datos** a través de los **servicios web** REST publicados, es lo más indicado.

4.2- ¿Qué es un servicio web?

Un **servicio web** (en inglés, web service o web services) es una tecnología que utiliza un **conjunto de protocolos** y estándares que sirven para i**ntercambiar datos entre aplicaciones**. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

Resumiendo se podría definir un servicio web como aquel que **atiende** las **peticiones** de los **clientes** web y les **envía los recursos solicitados**.

A continuación se listan los tres estilos de usos más comunes:

- **Remote Procedure Calls (RPC, Llamadas a Procedimientos Remotos)**: Los Servicios Web basados en RPC presentan una interfaz de llamada a procedimientos y funciones distribuidas, lo cual es familiar a muchos desarrolladores. Típicamente, la unidad básica de este tipo de servicios es

la operación WSDL (Web Services Description Language, es un formato del Extensible Markup Language (XML) que se utiliza para describir servicios web (WS)). Las primeras herramientas para Servicios Web estaban centradas en esta visión. Algunos lo llaman la primera generación de Servicios Web. Esta es la razón por la que este estilo está muy extendido. Sin embargo, ha sido algunas veces criticado por no ser débilmente acoplado, ya que suele ser implementado por medio del mapeo de servicios directamente a funciones específicas del lenguaje o llamadas a métodos. Muchos especialistas creen que este estilo debe desaparecer.

- **Arquitectura Orientada a Servicios (Service-oriented Architecture, SOA**). Los Servicios Web pueden también ser implementados siguiendo los conceptos de la arquitectura SOA, donde la unidad básica de comunicación es el mensaje, más que la operación. Esto es típicamente referenciado como servicios orientados a mensajes. Los Servicios Web basados en SOA son soportados por la mayor parte de desarrolladores de software y analistas. Al contrario que los Servicios Web basados en RPC, este estilo es débilmente acoplado, lo cual es preferible ya que se centra en el "contrato" proporcionado por el documento WSDL, más que en los detalles de implementación subyacentes.

- **REST (REpresentation State Transfer)**. Los Servicios Web basados en REST intentan emular al protocolo HTTP o protocolos similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar (por ejemplo GET, PUT,...). Por tanto, este estilo se centra más en interactuar con recursos con estado, que con mensajes y operaciones.

4.3- Servicios web del Ministerio

Como se ha indicado en puntos anteriores, el portal http://geoportalgasolineras.es/ tiene publicado un **catálogo de servicios web REST**, que nos devuelve los **datos actualizados de las Estaciones de Servicio,** los **precios de los combustibles** así como toda la información relacionada (ciudades, provincias, tipos de combustibles…). Estos servicios, tienen diferentes filtros para su consulta, y quedan resumidos en la siguiente tabla:

Servicio	URL
Estaciones de servicio	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/

Servicio	URL
Estaciones de servicio por comunidad autónoma	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/FiltroCCAA/{IDCCAA}
Estaciones de servicio por comunidad autónoma y producto	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/FiltroCCAAProducto/{IDCCAA}/{IDPRODUCTO}
Estaciones de servicio por municipio	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/FiltroMunicipio/{IDMUNICIPIO}
Estaciones de servicio por municipio y producto	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/FiltroMunicipioProducto/{IDMUNICIPIO}/{IDPRODUCTO}
Estaciones de servicio por producto	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/FiltroProducto/{IDPRODUCTO}
Estaciones de servicio por provincia	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/FiltroProvincia/{IDPROVINCIA}
Estaciones de servicio por provincia y producto	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/FiltroProvinciaProducto/{IDPROVINCIA}/{IDPRODUCTO}
Listado comunidades autónomas	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/Listados/ComunidadesAutonomas/
Listado de municipios	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/Listados/Municipios/

Servicio	URL
Listado de municipios por provincia	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/Listados/MunicipiosPorProvincia/{IDPROVINCIA}
Listado de productos petrolíferos	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/Listados/ProductosPetroliferos/
Listado de provincias	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/Listados/Provincias/
Listado de provincias por comunidad autónoma	https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/Listados/ProvinciasPorComunidad/{IDCCAA}

Todos los servicios utilizan únicamente el **método HTTP GET,** y la información necesaria para su uso es accesible desde el propio portal.

A continuación se muestra parte de la información de ayuda, para el servicio que devuelve el precio para todos los productos de todas las estaciones de servicio de España. El servicio en la tabla anterior corresponde con **Estaciones de servicio.**

Referencia para:
https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/

Url:
https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/

Método HTTP: GET

Servicio	URL	Cuerpo
Solicitar	N/D	El cuerpo Solicitar está vacío.
Respuesta	Xml	Ejemplo, Esquema
Respuesta	Json	Ejemplo

Como se puede observar la respuesta tanto de este servicio como del resto, puede obtenerse en **formatos XML y Json**.

Para comprender un poco mejor lo que implica el formato en el tipo de respuesta, se describirán de forma básica ambos formatos:

XML

XML, siglas en inglés de eXtensible Markup Language, traducido como "Lenguaje de Marcado Extensible", es un **meta-lenguaje** que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) **utilizado para almacenar datos** en forma legible. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande, con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que **permite** la compatibilidad entre sistemas para **compartir la información de una manera segura, fiable y fácil.**

JSON

JSON, acrónimo de **JavaScript Object Notation**, es un **formato de texto** ligero para el **intercambio de datos**. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

Hay muchos artículos sobre la conveniencia de uno u otro formato en el intercambio de datos. Si bien la **implementación final** de la solución, podría haber empleado cualquiera de los dos formatos, me he decantado por el **formato JSON,** ya que su soporte a arrays proporciona mayor sencillez a la estructura de datos devuelta.

Según diferentes estudios los tiempos de transferencia se reducen empleando JSON, y aunque en la arquitectura empleada no es un factor crítico, los resultados empleando JSON son satisfactorios.

A continuación se muestra un **ejemplo** concreto de **consulta y respuesta** de uno de los **servicios web**: Estaciones de servicio por municipio y producto, cuya llamada como se puede apreciar en el cuadro anterior se realizaría con la URL

https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/FiltroMunicipioProducto/IDMUNICIPIO/IDPRODUCTO

Del servicio: ***Listado de Municipios*** obtendríamos el ID correspondiente al municipio, en el ejemplo Alcoy, con una respuesta como la que sigue:

```
[
    …
    {
        "IDMunicipio": "147",
        "IDProvincia": "03",
        "IDCCAA": "10",
        "Municipio": "Alcoy/Alcoi",
        "Provincia": "ALICANTE",
        "CCAA": "Comunidad Valenciana"
    },
    …
]
```

Y del servicio: ***Listado de productos petrolíferos*** el ID del producto al que deseamos hacer la consulta que en este caso es el 5 que corresponde con Nuevo gasóleo A.

```
[
    …
    {
        "IDProducto": "5",
        "NombreProducto": "Nuevo gasóleo A",
        "NombreProductoAbreviatura": "NGO"
    },
    …
]
```

Por tanto para obtener los precios de "Nuevo Gasóleo A" en las estaciones de servicio de Alcoy realizaríamos la siguiente consulta:

https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/FiltroMunicipioProducto/147/5

Y el **resultado** obtenido, sería un objeto JSON como el que se muestra. Contiene la fecha de la consulta, una nota indicando que los precios se actualizan cada media hora, el resultado de la consulta y como elemento central, un **array de estaciones de servicio** en ListaEESSPrecio. El array contiene los **datos** de las diferentes **estaciones** que **cumplen los requisitos** de la **búsqueda** realizada.

En el ejemplo, para evitar extenderse sin aportar demasiado, se ha dejado intacta la información de dos de las cinco estaciones de servicio incluidas en la respuesta del web service. Para los tres elementos restantes se ha marcado con puntos suspensivos, para reflejar que incluirían los datos correspondientes a las estaciones restantes.

```
{
    "Fecha": "14/07/2017 20:13:11",
    "ListaEESSPrecio": [
        {
            "C.P.": "03800",
            "Dirección": "PG COTES BAIXES,CALLE C,PARC.1, 0",
            "Horario": "L-D: 06:00-00:00",
            "Latitud": "38,714361",
            "Localidad": "ALCOY/ALCOI",
            "Longitud (WGS84)": "-0,462528",
            "Margen": "D",
            "Municipio": "Alcoy/Alcoi",
            "PrecioProducto": "1,109",
            "Provincia": "ALICANTE",
            "Remisión": "OM",
            "Rótulo": "REPSOL",
            "Tipo Venta": "P",
            "IDEESS": "3475",
            "IDMunicipio": "147",
            "IDProvincia": "03",
            "IDCCAA": "10"
        },

        {
            "C.P.": "03800",
            "Dirección": "AVENIDA TIRANT, S/N",
            "Horario": "L-D: 06:00-22:00",
            "Latitud": "38,708750",
            "Localidad": "ALCOY/ALCOI",
            "Longitud (WGS84)": "-0,472556",
            "Margen": "D",
            "Municipio": "Alcoy/Alcoi",
            "PrecioProducto": "1,109",
            "Provincia": "ALICANTE",
            "Remisión": "OM",
            "Rótulo": "REPSOL",
            "Tipo Venta": "P",
            "IDEESS": "11428",
            "IDMunicipio": "147",
            "IDProvincia": "03",
            "IDCCAA": "10"
        },
        {
            …
        },
        {
          …
        },
        {
            …
        },
        {
            …
        }
    ],
    "Nota": "Archivo de todos los productos en todas las estaciones
de servicio. La actualización de precios se realiza cada media hora,
con los precios en vigor en ese momento.",
    "ResultadoConsulta": "OK"

}
```

4.4- Problemas con los servicios web del Ministerio

A medida que avanzaba con el desarrollo de la aplicación y se realizaban las pruebas necesarias para ir dándole forma a la app, se detectó que en ocasiones los **datos obtenidos** en las respuestas **no eran los esperados**. Había momentos en que la misma petición, daba resultados diferentes. Por ejemplo, tomando como referencia la utilizada en el apartado anterior, haciendo uso del mismo filtro por ciudad y tipo de combustible, en ocasiones devolvía la respuesta con las cinco EESS, y minutos después el resultado era un objeto JSON con resultado de consulta OK y el array de EESS vacío.

Se comprobó que **cuando se realizaban las actualizaciones** de los datos por el sistema, la consulta del web service devolvía un **objeto JSON** con "ResultadoConsulta": "OK" pero cuyo contenido era un **array** que se encontraba **vacío**. En ocasiones esta situación dura apenas unos segundos, pero en otros casos llega a prolongarse durante algunos minutos (en alguna de las pruebas prácticamente 5 minutos). Hay que recordar que las actualizaciones de datos se producen por parte del Ministerio cada media hora.

La problemática fuel trasladada al equipo informático responsable del servicio web, que me hicieron llegar la siguiente respuesta:

La problemática que comenta de los servicios REST es cierta. Hay casos en los que el método de actualización de precios para los servicios REST puede provocar que no se devuelva nada durante unos segundos. Nos estamos planteando cambiar el método de actualización de precios para que no suceda esto. Le agradecemos nos haya puesto sobre aviso al respecto.

Llegado a este punto, el enfoque de la solución debía cambiar. Como se ha comentado anteriormente la aplicación depende directamente de los **datos**, y si bien es **fundamental** que sean correctos y se encuentren debidamente actualizados, también es capital que estén **disponibles de forma continua**. Para una aplicación de consulta, el no poder asegurar la disponibilidad de los datos y que cada hora puedan llegar a no estar disponibles durante 10 minutos (un 16% del tiempo), parece claramente inadmisible.

4.5- Aproximación a la solución que asegure la continuidad de los datos

Parece claro que la única **opción** posible para **asegurar** la **continuidad** en el acceso de los **datos**, era **capturarlos** en algún momento en el que estuvieran disponibles en el servicio web del Ministerio, y **posteriormente acceder a la nueva fuente**, cada vez que se quisieran consultar.

Se plantearon diferentes soluciones, llegando a realizar pruebas para valorar su viabilidad. Para disponer de todos los datos, y permitir acceder posteriormente a cualquier consulta, en primer lugar debemos realizar una llamada que nos devuelva todos los precios de los diferentes tipos de combustible de todas las EESS de España. La consulta devuelve un conjunto de datos en formato JSON de alrededor de 11Mb. Para que los datos se encuentren actualizados, sería necesario realizar llamadas periódicas a este servicio.

Si se plantea realizar la llamada desde la app móvil, y posteriormente consultar los datos en local, las consultas periódicas necesarias para mantener los datos actualizados supondrían una sobrecarga innecesaria en tiempo, y en consumo de recursos en segundo plano. Penalizarían a las aplicaciones locales sin que se estuviera haciendo uso de ellas, o bien la consulta de los datos sería muy lenta dependiendo del enfoque de la solución. Lo que parece evidente es que, el acceso de datos con este enfoque no es una buena solución, ni en términos de eficiencia ni en tiempos de respuesta.

Otra aproximación, más costosa en su implantación inicial, e incluso en su mantenimiento, pero que tanto a priori como una vez realizadas las pruebas necesarias ofrece mejores resultados, sería trasladar el **back.end a un servidor propio**. Si somos capaces desde un servidor, mediante un proceso programado de forma periódica, de **llamar al web service** en algún momento en el que a priori se encuentre disponible. **Almacenar estos datos** en una base de datos con la estructura adecuada, y posteriormente **hacer accesibles estos datos mediante un servicio web**, el problema quedaría resuelto y para el cliente desde el front-end sería transparente y sin ningún coste en consumo de recursos, tiempo, ni disponibilidad de los datos. Una vez disponemos de los datos, es **posible** utilizarlos de forma que nos permita **ampliar la funcionalidad.** Por ejemplo, las EESS disponen de latitud y longitud, lo que nos permite con la debida consulta, y si le pasamos como parámetro al servidor en la llamada nuestra posición actual o bien cualquier posición, obtener resultados en función de esta.

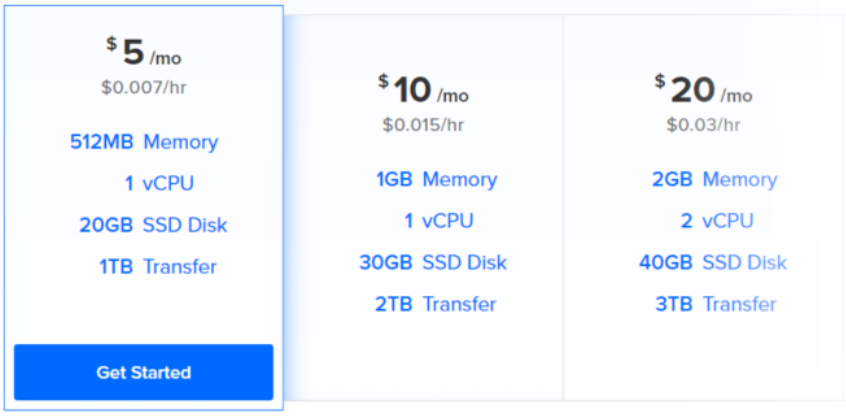
4.6- Servidor del Back-end

Para poder llevar a la práctica la aproximación descrita en el punto anterior, debemos hacerlo mediante un **servidor** que sea **accesible desde internet**, que tenga los **recursos necesarios** y sobretodo con el que **quede asegurada** la **disponibilidad del servicio**. Otro factor claramente determinante es el coste, sobre todo teniendo en cuenta que se trata de un TFG, con no mayor proyección a app gratuita del Google Play.

Teniendo en cuenta las necesidades, y estudiando diferentes opciones, se optó por un proveedor de máquinas virtuales en la nube llamado DigitalOcean (https://www.digitalocean.com/)

El proveedor dispone un variado número de planes, con precios en función de los recursos de la/s máquina/s contratadas y del ancho de banda máximo mensual. Los precios oscilan entre un máximo de 5$/mes y 640 $/mes.

Las características de la máquina virtual escogida con el plan de 5$/mes:



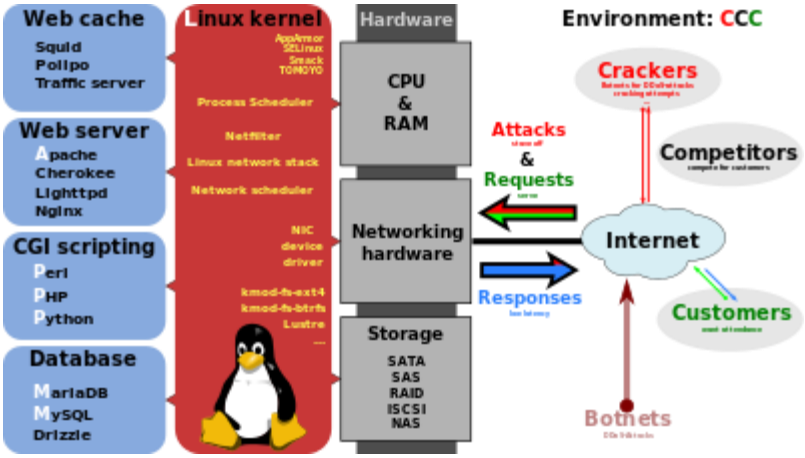
Como punto de partida del Back-end de la aplicación desarrollada, **512MB de memoria**, **1 CPU virtual** (Intel(R) Xeon(R) CPU E5-2650L v3 @ 1.80GHz), **20 GB de disco duro sólido** y **1 TB de transferencia mensual** son más que suficientes. Si llegado el momento fuera necesario aumentar los recursos, es posible hacerlo de forma sencilla.

El despliegue de la máquina virtual se hace en pocos minutos, e incluye la instalación de aplicaciones o pilas de aplicaciones como por ejemplo LAMP en un solo click. Los servidores que se ponen a disposición del usuario utilizan la solución de virtualización KVM es una solución para implementar virtualización completa con Linux.

Un punto muy importante es la disponibilidad del servicio, que en el acuerdo de nivel de servicio (**SLA** –Service Level Agreement) la compañía nos asegura un **Uptime** del **99,99%**, tanto para la máquina virtual como para el almacenamiento. Esto significa que como máximo en un año el servicio no estará disponible 53 minutos.

4.7- LAMP

Anteriormente se ha indicado que el servicio contratado permitía la instalación de la pila de aplicaciones LAMP. Se denomina "**LAMP**" a un grupo de software de código libre que se instala normalmente en conjunto para habilitar un servidor para alojar sitios y aplicaciones web dinámicas. Este término en realidad es un acrónimo que representa un **sistema operativo Linux** con un **servidor Apache**, el sitio de datos es almacenado en **base de datos MySQL** y el contenido dinámico es procesado con **PHP**.



El conjunto de software LAMP

A pesar de que en su origen estos programas de código abierto no fueron diseñados específicamente para trabajar de forma conjunta, la combinación se popularizó debido a su bajo coste de adquisición y ubicuidad de sus componentes (ya que vienen pre-instalados en la mayoría de las distribuciones linux). Cuando son combinados, representan una **solución de gran rendimiento y disponibilidad** para servidores de aplicaciones.

Como **puntos fuertes de LAMP** podemos destacar

- Es una tecnología **madura**
- Está bien **documentada**
- **Funciona bien** en apps web de tamaño medio y grande donde la gestión de los datos en tiempo real no es vital
- Tiene una **comunidad** muy grande
- PHP es el lenguaje de servidor **más utilizado del mundo**, lo usan WordPress, Drupal, Prestashop, Magento, Moodle y un largo, largo etcétera

Destacaremos también los principales puntos que se destacan como **inconvenientes de LAMP**

- No funciona bien en **aplicaciones de tiempo real** que manejan grandes cantidades datos de forma concurrente
- Emplea **varios lenguajes** de programación

Para el desarrollo del sistema planteado, las ventajas de utilizar LAMP inclinan claramente la balanza para su uso.

4.7.1. - Servidor HTTP Apache

Conocemos como servidor HTTP o **servidor web**, el **software** encargado de procesar una aplicación **en el lado del servidor**, **atendiendo** las **peticiones de los clientes**, y estableciendo en consecuencia conexiones con ellos. Estas conexiones **generan una respuesta en cualquier lenguaje**, y de acuerdo a las políticas de seguridad establecidas, que son interpretadas por el navegador web cliente. **Generalmente** se usa el **protocolo HTTP** para estas comunicaciones.

El **servidor HTTP Apache** es un servidor web HTTP de **código abierto** y uso **gratuito**, **multiplataforma** (está disponible para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras), que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. **Destaca por su seguridad y rendimiento**, siendo su punto fuerte la robustez.

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).

En **nuestro caso** el servidor Apache será herramienta indispensable para **publicar los servicios web**, a los que llamaremos desde la app móvil para obtener los datos deseados.

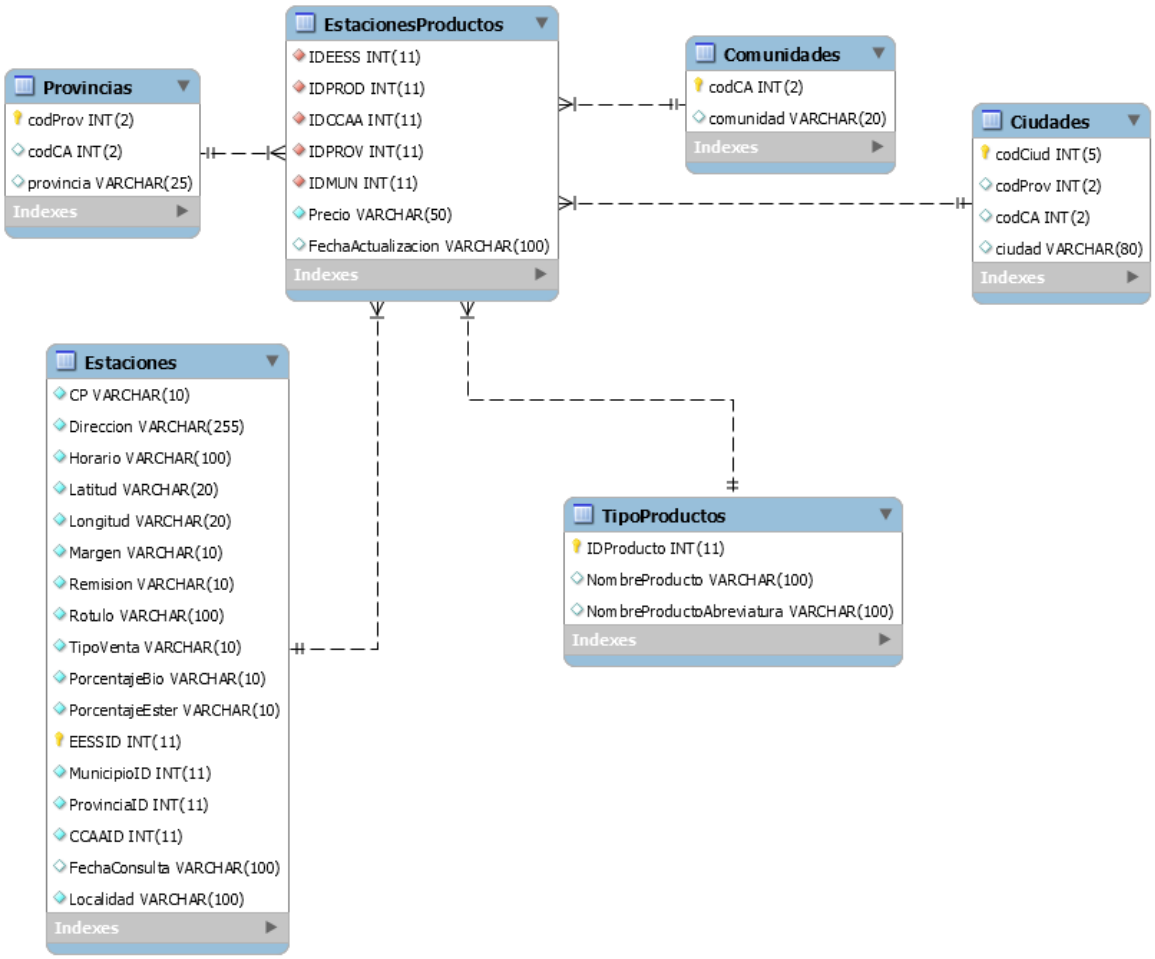
4.7.2. - Base de datos MySQL

MySQL es un **sistema de gestión de bases de datos relacional** desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como **la base datos open source más popular del mundo,** y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL A.B. fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

La **función de la BBDD** será **almacenar** los **datos** recuperados **del servicio web** del Ministerio, para ir actualizándolos de forma periódica y permitir **recuperarlos** de acuerdo a los criterios definidos en las **peticiones realizadas** mediante los servicios web publicados.

El **diagrama EER** que define la BBDD, tiene un aspecto como el siguiente:



Como se puede observar hay 6 tablas:

- **Comunidades**: contiene los códigos y nombres de las comunidades autónomas.
- **Provincia:** se encuentran definidos los códigos de provincias, comunidades autónomas a las que pertenecen estas y el nombre de las propias provincias.
- **Ciudades**: contiene los códigos de ciudad, provincia y comunidad autónoma a la que pertenecen y el nombre de ciudad.

Las tres tablas anteriores son una réplica de los datos del Ministerio que no se actualizarán ya que contiene los datos de todas las ciudades, Provincias y Comunidades Autónomas de España.

- **TiposProductos**: incluye los 8 productos petrolíferos empleados en la aplicación, definidos por el ID del producto, su nombre y su nombre abreviado.

- **Estaciones**: definen las EESS con la información de dirección, localidad, IDs de provincia y CCAA, latitud, longitud, rótulo o nombre y horario como campos principales.
- **EstacionesProductos**: contiene la información de los precios de los productos con una relación con las EESS, y códigos de: municipio, provincia, comunidad autónoma, tipo de combustible y fecha de actualización del precio del producto, que nos servirá para conocer el estado de actualización del mismo.

La **BBDD además** de la **estructura y relaciones** descritas anteriormente, incluye un **procedimiento almacenado**, para llevar a cabo la sincronización de los datos entre el web service del Ministerio y nuestra base de datos.

Un **procedimiento almacenado** (stored procedure en inglés) es un **programa almacenado físicamente** en una **base de datos**, que se compone de una o varias instrucciones SQL. Se pueden destacar las siguientes **ventajas** de utilizar procedimientos almacenados:

Mejora en la seguridad

Al incluir privilegios de base de datos con procedimientos almacenados que utilicen SQL estático, el administrador de bases de datos (DBA) puede mejorar la seguridad. El DBA o el desarrollador que crea el procedimiento almacenado debe tener los privilegios de la base de datos que necesita el procedimiento almacenado. Los usuarios de las aplicaciones cliente que llaman al procedimiento almacenado no necesitan estos privilegios. Esto puede reducir el número de usuarios que necesitan privilegios.

Reducción en el coste de desarrollo y aumento en la fiabilidad

En un entorno de aplicación de base de datos se repiten muchas tareas. Entre las tareas repetidas pueden encontrarse la devolución de un conjunto fijo de datos o la realización del mismo conjunto de peticiones a una base de datos. Al volver a utilizar un procedimiento común, un procedimiento almacenado puede proporcionar una manera enormemente eficaz de resolver estas situaciones repetitivas.

Seguridad, administración y mantenimiento centralizados para las rutinas comunes

Al gestionar la lógica compartida en un lugar del servidor, puede simplificarse la seguridad, la administración y el mantenimiento. Las aplicaciones cliente pueden llamar a procedimientos almacenados que ejecuten consultas de SQL con escaso o ningún procesamiento adicional.

A continuación se incluye el código del procedimiento comentado:

```
-- Se define el procedimiento sp_insertEESS con los nombres de los campos

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertEESS`(IN `IDEESSN` INT(11),
IN `IDMunicipio` INT(11), IN `IDProvincia` INT(11), IN `IDCCAA` INT(11), IN `cp`
VARCHAR(10), IN `direccion` VARCHAR(255), IN `horario` VARCHAR(100), IN `latitud`
VARCHAR(20), IN `longitud` VARCHAR(20), IN `margen` VARCHAR(10), IN `remision`
VARCHAR(10), IN `rotulo` VARCHAR(100), IN `tipoVenta` VARCHAR(10), IN
`porcBioEtanol` VARCHAR(10), IN `porcEstermetilico` VARCHAR(10), IN
`fechaConsulta` VARCHAR(100), IN `precioGasolina98` VARCHAR(50), IN
`precioBiodiesel` VARCHAR(50), IN `precioBioetanol` VARCHAR(50), IN
`precioGasoleoA` VARCHAR(50), IN `precioNuevoGasoleoA` VARCHAR(50), IN
`precioGasoleoB` VARCHAR(50), IN `precioGasolina95Proteccion` VARCHAR(50), IN
`localidad` VARCHAR(100))

BEGIN

/*
El primer bloque de instrucciones IF se ocupa de crear las EESS recuperadas si no
existen
*/

IF NOT EXISTS (SELECT EESSID FROM EstacionesServicio.Estaciones E WHERE
EESSID=IDEESSN) THEN

    INSERT INTO `Estaciones`(`CP`, `Direccion`, `Horario`, `Latitud`, `Longitud`,
    `Margen`, `Remision`, `Rotulo`, `TipoVenta`, `PorcentajeBio`,
    `PorcentajeEster`, `EESSID`, `MunicipioID`, `ProvinciaID`, `CCAAID`,
    `FechaConsulta`, `Localidad`)
    VALUES
    (cp,direccion,horario,latitud,longitud,margen,remision,rotulo,tipoVenta,porcB
    ioEtanol,porcEstermetilico,IDEESSN,IDMunicipio,IDProvincia,IDCCAA,fechaConsul
    ta, localidad);

END IF;

/*
El segundo bloque IF - ELSEIF, se encarga de insertar una nueva línea
correspondiente al precio del producto con ID=3 (Gasolina98) si no existía y si
ya había una línea para éste producto la actualiza con el nuevo precio si este no
es nulo, actualizando la fecha de actualización para el producto en la EESS
*/

IF NOT EXISTS (SELECT IDEESS FROM EstacionesServicio.EstacionesProductos E WHERE
IDEESS=IDEESSN AND IDPROD=3 AND IDCCAA=IDCCAA AND IDPROV=IDProvincia AND
IDMUN=IDMunicipio) THEN

    INSERT INTO `EstacionesProductos`(`IDEESS`, `IDPROD`, `IDCCAA`, `IDPROV`,
    `IDMUN`, `Precio`) VALUES
    (IDEESSN,3,IDCCAA,IDProvincia,IDMunicipio,precioGasolina98);



ELSEIF precioGasolina98<>'' THEN

     UPDATE EstacionesServicio.EstacionesProductos SET Precio=precioGasolina98,
     FechaActualizacion=fechaConsulta WHERE IDEESS=IDEESSN AND IDPROD=3 AND
     IDCCAA=IDCCAA AND IDPROV=IDProvincia AND IDMUN=IDMunicipio;

END IF;

/*
Éste último bloque IF - ELSEIF se repetiría para cada uno de los productos,
debiendo modificar el ID del producto y la comprobaación, de que el precio
recuperado del tipo de combustible en cuestión no es nulo.
A modo de ejemplo se incluye el bloque correspondiente a otro tipo de combustible
*/
```

```
IF NOT EXISTS (SELECT IDEESS FROM EstacionesServicio.EstacionesProductos E WHERE
IDEESS=IDEESSN AND IDPROD=4 AND IDCCAA=IDCCAA AND IDPROV=IDProvincia AND
IDMUN=IDMunicipio) THEN

    INSERT INTO `EstacionesProductos`(`IDEESS`, `IDPROD`, `IDCCAA`, `IDPROV`,
    `IDMUN`, `Precio`) VALUES
    (IDEESSN,4,IDCCAA,IDProvincia,IDMunicipio,precioGasolina98);

ELSEIF precioNuevoGasoleoA<>'' THEN

     UPDATE EstacionesServicio.EstacionesProductos SET Precio=precioGasolina98,
     FechaActualizacion=fechaConsulta WHERE IDEESS=IDEESSN AND IDPROD=3 AND
     IDCCAA=IDCCAA AND IDPROV=IDProvincia AND IDMUN=IDMunicipio;

END IF;

…

END
```

4.7.3. - PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un **lenguaje de código abierto**, originalmente diseñado para el **desarrollo web de contenido dinámico**. Fue uno de los primeros lenguajes de programación del lado del servidor, que se podían incorporar directamente en el documento HTML, en lugar de llamar a un archivo externo que procese los datos. El **código es interpretado por un servidor web** con un módulo de procesador de PHP que **genera la página web resultante**.

Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún coste.

Mediante **scripts PHP** se ha implementado la **sincronización de los datos** que devuelve el servicio web del Ministerio, con los que tenemos almacenados en nuestra BBDD. **También** se ha empleado PHP para **publicar los servicios web** que responderán a las peticiones de los usuarios desde la app móvil.

A continuación se incluye el código correspondiente a estas implementaciones.

Sincronización.php

```
<?php
   $param = $_REQUEST['oAuth'];

   if($param == "codigoAutorizacion"){
       // URL para la llamada al webs ervice del Ministerio
       $service_url = 'https://sedeaplicaciones.minetur.gob.es/
              ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/';
```

```
// Obtiene la respuesta de la llamada que contiene todos los precios de
// todas la EESS
$response = file_get_contents($service_url);

// Convierte el texto del JSON en un objeto JSON al que poder acceder por
// los campos
$obj = json_decode($response,true);

 $fechaSincro = $obj['Fecha'];

// Estaciones contendrá un array con todas las EESS devueltas por la
// consulta
 $Estaciones = $obj['ListaEESSPrecio'];

// Para cada elemento del array es ejecuta el bucle
foreach($Estaciones as $Estacion){

        // Se asigna a una variable, cada uno de los campos que contiene el
        // JSON, para posteriormente emplearlo en la llamada
        $IDEESS = $Estacion['IDEESS'];
        $IDMunicipio = $Estacion['IDMunicipio'];
        $IDProvincia = $Estacion['IDProvincia'];
        $IDCCAA = $Estacion['IDCCAA'];
        $rotulo = $Estacion['Rótulo'];
        $tipoVenta = $Estacion['Tipo Venta'];
        $remision = $Estacion['Remisión'];
        $cp = $Estacion['C.P.'];
        $direccion = $Estacion['Dirección'];
        $horario = $Estacion['Horario'];
        $latitud = $Estacion['Latitud'];
        $longitud = $Estacion['Longitud (WGS84)'];
        $margen = $Estacion['Margen'];
        $precioBiodiesel = $Estacion['Precio Biodiesel'];
        $precioBioetanol = $Estacion['Precio Bioetanol'];
        $precioGasNaturalComprimido = $Estacion['Precio Gas Natural
        Comprimido'];
        $precioGasNaturalLicuado = $Estacion['Precio Gas Natural Licuado'];
        $precioGaseslicuadosdelpetróleo = $Estacion['Precio Gases licuados
        del petróleo'];
        $precioGasoleoA = $Estacion['Precio Gasoleo A'];
        $precioGasoleoB = $Estacion['Precio Gasoleo B'];
        $precioGasolina95Protección = $Estacion['Precio Gasolina 95
        Protección'];
        $precioGasolina98 = $Estacion['Precio Gasolina  98'];
        $precioNuevoGasoleoA = $Estacion['Precio Nuevo Gasoleo A'];
        $porcBioEtanol = $Estacion['% BioEtanol'];
        $porcEstermetilico = $Estacion['% Éster metílico'];

        // La variable connection contiene los datos necesarios para poder
        // establecer la conexión con el servidor
        $connection = mysqli_connect("localhost", "root", "contraseñaBBDD",
        "EstacionesServicio", "3306");

        /*
        Define la llamada del procedimiento almacenado con los campos
        asignados anteriormente. Hay que destacar que se utiliza la
        función .mysqli_real_escape_string para evitar errores en las
        consultas en el procedimiento almacenado y protegernos de posibles
        ataques de inyección SQL
        */
        $storedProceure = "CALL
        sp_insertEESS(".$$IDEESS.",".$IDMunicipio.",".$IDProvincia.",
        ".$IDCCAA.", '".mysqli_real_escape_string($connection, $cp)."',
        '".mysqli_real_escape_string($connection, $direccion)."',
        '".mysqli_real_escape_string($connection,
        $horario)."','".mysqli_real_escape_string($connection, $latitud)."',
        '".mysqli_real_escape_string($connection,
```

```
            $longitud)."','".mysqli_real_escape_string($connection, $margen)."',
            '".mysqli_real_escape_string($connection,
            $remision)."','".mysqli_real_escape_string($connection, $rotulo)."',
            '".mysqli_real_escape_string($connection,
            $tipoVenta)."','".mysqli_real_escape_string($connection,
            $porcBioEtanol)."', '".mysqli_real_escape_string($connection,
            $porcEstermetilico)."')";

            // Llama al procedimiento almacenado definido anteriormente
            $result = mysqli_query($connection, $storedProceure) or die("Query
            fail: " . mysqli_error());

            }
    }

?>
```

El siguiente código descrito corresponde con la llamada al servicio web con filtro por municipio y tipo de combustible.

GetEstaciones.php

Ejemplo de llamada para obtener los precios de "Nuevo Gasóleo A" (IDProducto = 5) en las estaciones de servicio de Alcoy (IDMunicipio=147):

http:// IPServidor /GetEstaciones.php?IDMunicipio=147&IDProducto=5

```
<?php

    // Se recuperan los parámetros de la consulta IDMunicipio e IDProducto
    $IDMunicipio = $_REQUEST['IDMunicipio'];
    $IDProducto = $_REQUEST['IDProducto'];

    $servername = "localhost";
    $username = "root";
    $password = "contraseñaBBDD";
    $dbname = "EstacionesServicio";

    // Se crea la conexión con la BBDD
    $conn = new mysqli($servername, $username, $password, $dbname);

    // Se comprueba la conexión con la BBDD
    if ($conn->connect_error) {
       die("Connection failed: " . $conn->connect_error);
    }


    // Se monta la select para recuperar los datos filtrando por los
    // parámetros IDMunicipio e IDProducto

    $sql = "SELECT CP AS 'C.P.', Direccion AS 'Dirección', Horario, Latitud,
    E.Localidad AS Localidad,  Longitud AS 'Longitud (WGS84)', Margen, C.ciudad
    AS Municipio, EP.Precio AS PrecioProducto, P.provincia AS Provincia,
    Remision, Rotulo AS 'Rótulo', EP.IDEESS AS IDEESS, EP.IDMUN AS IDMunicipio,
    EP.IDPROV AS IDProvincia, EP.IDCCAA, FechaConsulta AS FechaActualizacion " .
    " FROM EstacionesServicio.EstacionesProductos AS EP " .
    " LEFT JOIN EstacionesServicio.Estaciones AS E ON EP.IDEESS=E.EESSID " .
    " LEFT JOIN EstacionesServicio.Provincias AS P ON EP.IDPROV=P.codProv " .
```

```
    " LEFT JOIN EstacionesServicio.Ciudades AS C ON EP.IDMUN=C.codCiud " .
    " WHERE EP.IDMUN=".$IDMunicipio." AND EP.IDPROD=".$IDProducto." AND
    EP.Precio<>''";

    // Los datos recuperados se montan en un array
    $result = $conn->query($sql);
    $rows = array();
    if ($result->num_rows > 0) {
       while($row = mysqli_fetch_assoc($result)) {
             $rows[] = array_map('utf8_encode', $row);     }
    }
    $conn->close();

    // Se convierte en JSON
    $json = json_encode($rows);

    // Se hace un echo del JSON si no se ha producido ningún error
    if ($json)
       echo '{"ResultadoConsulta": 0, "ListaEESSPrecio":'.json_encode($rows).'}';
    else
       echo json_last_error_msg();

?>
```

El siguiente código descrito corresponde con la llamada al servicio web con filtro por municipio y tipo de combustible.

GetEstacionesXY.php

Ejemplo de llamada para obtener los precios de "Gasóleo A" (IDProducto = 4) en un radio de 10km (Distance=10**)** alrededor de la posición definida por la latitud 38.340167 y longitud -0.540083:

http://IPServidor/GetEstacionesXY.php?IDProducto=4&Distance=10&Latitude=38.340167&Longitude=-0.540083

```
<?php

    // Se recuperan los parámetros de la consulta IDProducto, latitude, longitude
    // y Distance que corresponde con la distancia del radio máximo de búsqueda

    $IDProducto = $_REQUEST['IDProducto'];
    $latitude = $_REQUEST['Latitude'];
    $longitude = $_REQUEST['Longitude'];
    $distance = $_REQUEST['Distance'];



    $servername = "localhost";
    $username = "root";
    $password = "contraseñaBBDD";
    $dbname = "EstacionesServicio";

    // Se crea la conexión con la BBDD
    $conn = new mysqli($servername, $username, $password, $dbname);
```

```
    // Se comprueba la conexión con la BBDD
    if ($conn->connect_error) {
       die("Connection failed: " . $conn->connect_error);
    }

    /* Se monta la select para recuperar los datos filtrando por el parámetro
    IDProducto y el cálculo necesario para obtener las EESS que se encuentren
    a una distancia máxima de la latitud y longitud pasadas como argumentos
    */
    $sql = "SELECT CP AS 'C.P.', Direccion AS 'Dirección', Horario, Latitud,
    E.Localidad AS Localidad,  Longitud AS 'Longitud (WGS84)', Margen, C.ciudad
    AS Municipio, EP.Precio AS PrecioProducto, P.provincia AS Provincia,
    Remision, Rotulo AS 'Rótulo', EP.IDEESS AS IDEESS, EP.IDMUN AS IDMunicipio,
    EP.IDPROV AS IDProvincia, EP.IDCCAA, FechaConsulta AS FechaActualizacion " .
                    " ,(acos(sin(radians(".$latitude.")) *
    sin(radians(REPLACE(Latitud, ',', '.'))) + cos(radians(".$latitude.")) *
    cos(radians(REPLACE(Latitud, ',', '.'))) * cos(radians(".$longitude.") -
    radians(REPLACE(Longitud, ',', '.')))) * 6371) as Distancia" .

    " FROM EstacionesServicio.EstacionesProductos AS EP " .
    " LEFT JOIN EstacionesServicio.Estaciones AS E ON EP.IDEESS=E.EESSID " .
    " LEFT JOIN EstacionesServicio.Provincias AS P ON EP.IDPROV=P.codProv " .
    " LEFT JOIN EstacionesServicio.Ciudades AS C ON EP.IDMUN=C.codCiud " .
    " WHERE EP.IDPROD=".$IDProducto." AND EP.Precio<>'' AND " .
    "(acos(sin(radians(".$latitude.")) * sin(radians(REPLACE(Latitud, ',', '.')))
    + cos(radians(".$latitude.")) * cos(radians(REPLACE(Latitud, ',', '.'))) *
    cos(radians(".$longitude.") - radians(REPLACE(Longitud, ',', '.')))) *
    6371)<=".$distance;


    // Los datos recuperados se montan en un array
    $result = $conn->query($sql);
    $rows = array();
    if ($result->num_rows > 0) {
       while($row = mysqli_fetch_assoc($result)) {
             $rows[] = array_map('utf8_encode', $row);     }
    }
    $conn->close();

    // Se convierte en JSON
    $json = json_encode($rows);

    // Se hace un echo del JSON si no se ha producido ningún error
    if ($json)
       echo '{"ResultadoConsulta": 0, "ListaEESSPrecio":'.json_encode($rows).'}';
    else
       echo json_last_error_msg();

?>
```

5. Front – End

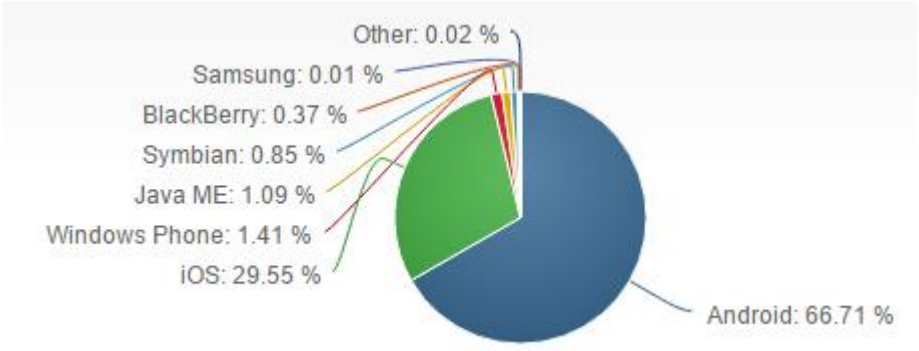
5.1. Comparativa Sistemas Operativos móviles

Un sistema operativo móvil es un **conjunto de programas de bajo nivel** que permite la abstracción de las propiedades del hardware específico del teléfono móvil y **provee servicios a las aplicaciones móviles**, que se ejecutan sobre él.
Al igual que los PCs que utilizan Windows o Linux, los dispositivos móviles tienen sus sistemas operativos como Android, iOS, Windows Phone, etc.

Los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos.

Según el servicio de estadísticas **NetMarketShare**, la cuota de mercado de sistemas operativos móviles a principios de 2017 es el siguiente:

- **Android** 66,71 % (en países como España las diferencias son más significativas, donde Android tiene más del 90% de la cuota de mercado)
- **iOS** 29,55 %
- **Windows Phone** 1,41 %
- **BlackBerry OS** 0,37 %



5.1.1. Android

Android Inc. es la empresa que creó el sistema operativo móvil. Se fundó en 2003 y fue adquirida por **Google Inc**. en el año 2005 y en 2007 fue lanzado al mercado.

Originalmente era un sistema pensado para las cámaras digitales profesionales pero fue modificado por Google para ser utilizado en dispositivos móviles como los teléfonos inteligentes y tablets.

Cuenta con el mayor número de instalaciones de smartphones en todo el mundo y está basado en el núcleo Linux. Las aplicaciones para Android se escriben y desarrollan en Java aunque con unas APIs propias.

En 2007 Google fundó la Open Handset Alliance formada por un grupo de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles. Juntos desarrollaron Android, **la primera plataforma móvil completa, abierta y libre**.

Algunos de sus miembros son Google, HTC, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia y Wind River Systems.

Aunque el sistema operativo Android **es software libre y de código abierto**, en los dispositivos vendidos, gran parte del software incluido es software propietario y de código cerrado.

5.1.2. iOS

iOS (anteriormente denominado iPhone OS) es propiedad de **Apple Inc**. Tiene la segunda mayor base de smartphones instalada en todo el mundo después de Android.

Es de **código cerrado y propietario** y construido a partir de Darwin, o lo que es lo mismo, el kernel del sistema operativo de Apple, Mac OS X.

iOS es el sistema operativo que **da vida a dispositivos como el iPhone, el iPad, el iPod Touch o el Apple TV**.

5.1.3. Windows Phone

Windows 10 Mobile (anteriormente llamado Windows Phone) es de **Microsoft**, diseñado para teléfonos inteligentes y tabletas.

Es de código cerrado y propietario y utiliza como núcleo Windows NT.

En febrero de 2010 se dio a conocer Windows Phone que integra servicios de Microsoft como OneDrive y Office, Xbox Music, Xbox Vídeo, juegos Xbox Live y Bing, pero también se integra con otros servicios que no son de su propiedad, como Facebook y cuentas de Google.

A principios de 2015, Microsoft anunció que la marca Windows Phone sería reemplazada por **Windows 10 Mobile** con el objetivo de lograr una mayor integración y unificación con su homólogo para PCs Windows 10, y proporcionar una plataforma para smartphones y tablets con tamaños de pantalla de 8 pulgadas.

El concepto de **núcleo del sistema operativo** ha aparecido a lo largo del presente punto, por ello se ha considerado interesante su descripción. El núcleo del sistema operativo,

también llamado kernel (núcleo en alemán) es aquella **parte de un sistema operativo que interactúa de forma directa con el hardware** de una máquina. Entre las funciones principales del kernel se encuentran: la gestión de memoria, la administración del sistema de archivos, la administración de servicios de entrada/salida y la asignación de recursos entre los usuarios.

En la tabla adjunta se incluyen algunas de las **características más relevantes** de los tres sistemas operativos predominantes:

Android	iOS	Windows 10 Mobile
Compañía		
Open Handset Alliance	Apple Inc.	Microsoft
Última versión		
7.1.2	10.3.2	10.0.15031.0
Licencia		
Libre y de código abierto, pero por lo general se incluye con aplicaciones y drivers propietarios.	Propietaria excepto para componentes de código abierto.	Propietaria
Programado en		
C, C ++, Java.	C , C ++ , Objective-C , Swift.	.NET C#, VB.NET, Silverlight, native C/C++, WinRTP (XMLA), DirectX
Tienda oficial de aplicaciones		
Google Play	App Store	Windows Store
Coste de desarrollo para el OS móvil		
Gratis	Gratis con Xcode 7	Gratis
Coste para publicar aplicación en la tienda oficial		
US$25 una vez por	US$99 al año	US$19, una vez por un

Android	iOS	Windows 10 Mobile
individuo		individuo; y $99 para una cuenta de la compañía
Soporte de impresoras		
4.4+ usando Google Cloud Print pero no a través de USB	Sí (AirPrint)	10+
Motor de navegador web por defecto		
Blink	WebKit	Trident (EdgeHTML después de la vesión 10)
Navegadores web disponibles		
Chrome para Android, Opera, Firefox	Safari, Chrome para iOS, Opera Mini, Firefox	Internet Explorer, Opera Mini, UC Browser, Microsoft Edge
Motor de búsqueda de los navegadores		
Muchos (entre ellos Google)	Bing, Google, Yahoo! Search, DuckDuckGo	Muchos (entre ellos Bing)
Voz sobre IP		
Protocolo SIP o software de terceros	FaceTime y software de terceros	8+ Skype
Software de pago con tecnología NFC		
Disponible en cualquier dispositivo compatible con el hardware. Android Pay para pagos NFC disponible en Play Store.	8+: iPhone 6/6 Plus vía Apple Pay	8+
USB On-The-Go		
3.1 +	No	10+

Android	iOS	Windows 10 Mobile
Reproducción de audio		
AAC LC/LTP 3GPP, HE-AACv1 (AAC+), HE-AACv2 (AAC+ mejorado) AMR-NB, AMR-WB, MP3, MIDI (tipo 0 y 1, versiones DLS 1 y 2), Ogg Vorbis, PCM/WAVE, FLAC, WAVE, Opus	AAC, AAC protegido (del iTunes Store), HE-AAC, MP3, MP3 VBR, Audible (formatos 2, 3, 4, Audible Enhanced Audio, AAX y AAX+), Apple Lossless, AIFF, WAV	MP3, WMA Std 9.2, WMA Pro, FLAC, AMR-NB, AAC-LC, AAC+, eAAC+
Reproducción de vídeo		
H.263, H.264 (hasta Baseline Profile), H.265 HEVC, MPEG-4 SP, DivX, XviD, VP8, VP9	H.264 (hasta High Profile), MPEG-4, M-JPEG	H.263, H.264, WMV, MPEG4, MPEG4 @ HD 720p 30fps, DivX, XviD
Teclado Bluetooth		
2.3+, en versiones anteriores a través de software de terceros	Sí	8.1u2+
Teclado USB		
3.1+	Con el kit de conexión de cámara	10+ algunos dispositivos
Cliente SSH		
Sí	Sí	Sí
VPN		
Sí	Sí	8.1+
Reconocimiento de voz		
Sí	5+ (Siri)	8.1+ Microsoft Cortana

5.2. Consideraciones de diseño

5.2.1 Justificación app nativa

A la hora de pensar en la creación de una aplicación móvil hay que tener en cuenta multitud de factores, entre ellos uno que tiene especial relevancia es tipo de aplicación **¿app nativa, web o híbrida?**

Entre este tipo de aplicaciones existen diferencias significativas en cuanto a tecnología de desarrollo, rendimiento, coste… En el siguiente gráfico se resumen las ventajas e inconvenientes de cada uno de ellos.

Aplicación nativa

La aplicación nativa está desarrollada y **optimizada específicamente para el sistema operativo** determinado y la plataforma de desarrollo del fabricante (Android, Blackberry, etc).

Este tipo de aplicaciones **se adapta al 100% con las funcionalidades y características del dispositivo,** obteniendo así una **mejor experiencia de uso**. Sin embargo, el desarrollo de una aplicación nativo comporta un **mayor coste**, puesto que si se desea realizar una aplicación multiplataforma se ha de realizar una nueva versión para cada sistema operativo, multiplicando así los costes de desarrollo.

Aplicación web

La aplicación web es la opción **más sencilla y económica** de crear aplicaciones, puesto que al desarrollar una única aplicación se reducen al máximo los costes de desarrollo. Asimismo, en este tipo de aplicaciones, puede utilizarse el **"responsive web design"**, creando así una única aplicación adaptada para todo tipo de dispositivos.

Por el contrario, la aplicación web ofrece una **peor experiencia de uso**, puesto que ignora las características del dispositivo y una **menor seguridad** ya que depende de la seguridad que ofrezca el propio navegador.

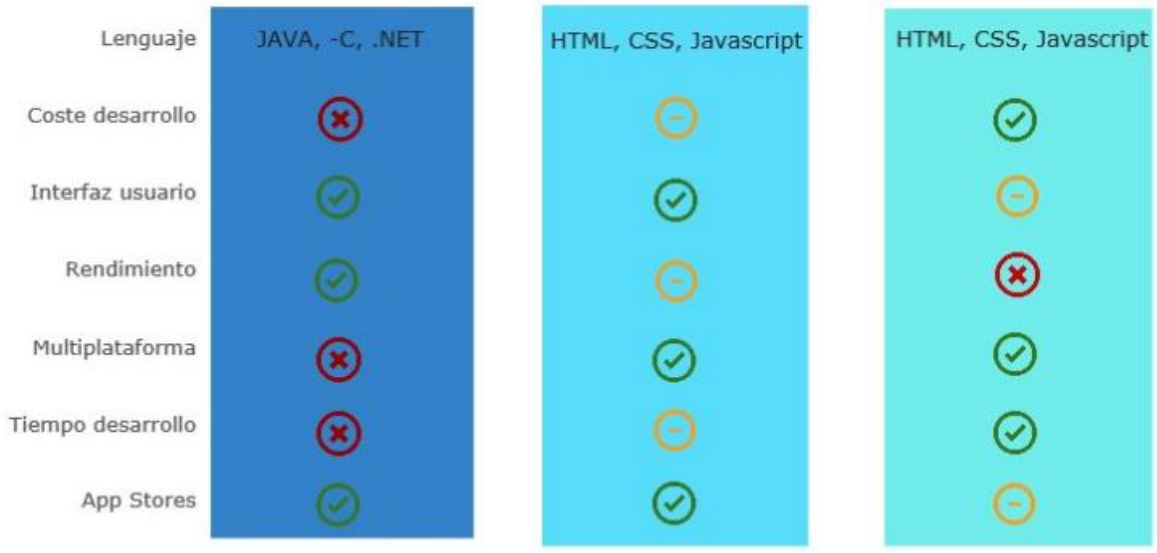
Aplicación híbrida

Este tipo de aplicación **aprovecha al máximo la versatilidad de un desarrollo web y tiene la capacidad de adaptación al dispositivo como una app nativa.**

Permite utilizar los estándares de desarrollo web (HTML5) y aprovechar las funcionalidades del dispositivo tales como la cámara, el GPS o los contactos. Además, comporta un **menor coste** que una **aplicación nativa** y una **mejor experiencia de uso** que una **aplicación web**.

Sin embargo, tiene un **rendimiento ligeramente inferior** al de una **aplicación nativa** debido a que cada página debe ser renderizada desde el servidor y supone una **mayor dificultad de desarrollo.**

A modo de resumen se aporta la siguiente comparativa gráfica:



Si bien la aplicación podría haberse desarrollado como una app híbrida intentando llegar a diferentes sistemas operativos móviles, la elección ha sido **desarrollar de forma nativa** la app para **Android**.

La elección del desarrollo nativo Android se ha realizado por diferentes **motivos**:

- Disponía de **formación previa** en **desarrollo** de aplicaciones **Android** en Java.
- Por motivos personales, el tiempo para la realización del proyecto era limitado, y adquirir los conocimientos para desarrollar la aplicación multiplataforma iría en detrimento de la calidad de la app. He preferido intentar hacer una **mejor y más completa aplicación Android**, y no quedarme en un **intento** con una con menor funcionalidad y peor calidad pero **multiplataforma**.
- Dada la **distribución de Android** en España, según datos recientes cercanos al **92%** del **mercado**, pese a la limitación de que se trate de una app mono plataforma, alcanzamos el grueso del mercado.
- Podemos **aprovechar** las **ventajas** de las **aplicaciones nativas**, como: mejor experiencia de uso; funcionalidades y características adaptadas al 100% a los dispositivos; y máxima optimización para Android y sus diferentes versiones.

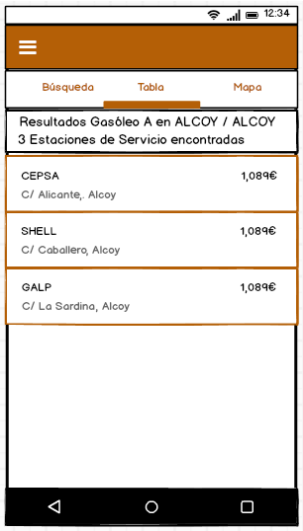
5.2.2 Diseño previo. Mockups

Como se ha indicado en diferentes puntos de la memoria, el diseño previo de la aplicación consistía en tres pantallas con las funciones claramente diferenciadas:

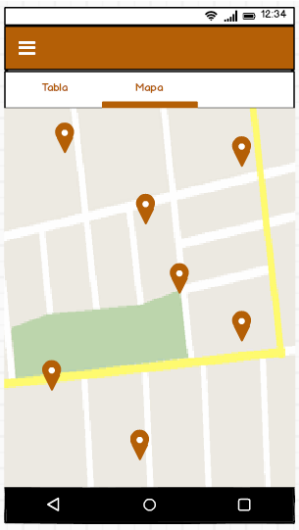
- **Búsqueda**: pantalla desde donde se seleccionarán los criterios de búsqueda: el destino de la búsqueda y el tipo de combustible.
- **Tabla de resultados**: listado con los datos obtenidos en la búsqueda para cada una de las estaciones de servicio.
- **Mapa**: representación sobre un mapa de la distribución de las EESS.

A continuación se incluyen los mockups diseñados, en la fase previa al desarrollo.

Pantalla de búsqueda



Pantalla Tabla resultados



Pantalla Mapa

5.2.3 Elección entorno de desarrollo

En otro momento del desarrollo de aplicaciones Android, se podría haber planteado la duda acerca del entorno de desarrollo a escoger. Las opciones destacadas hubieran sido Eclipse y Android.

A **finales de 2015**, **Google dejó de dar soporte para el plugin ADT para Eclipse,** y adoptó Android Studio como nueva herramienta oficial de desarrollo para Android. Por ello, actualmente ya no hay debates basados en preferencias u opiniones. Si queremos desarrollar aplicaciones nativas actualizadas **debemos utilizar Android Studio**.

En el punto **5.5 Entorno de desarrollo. Android Studio** se describen con detalle las características y funcionalidades del IDE oficial.

5.3. Arquitectura Android

El siguiente gráfico muestra la arquitectura de Android. Como se puede ver está formada por **cuatro capas**. Una de las características más importantes es que todas las capas están **basadas en software libre.**

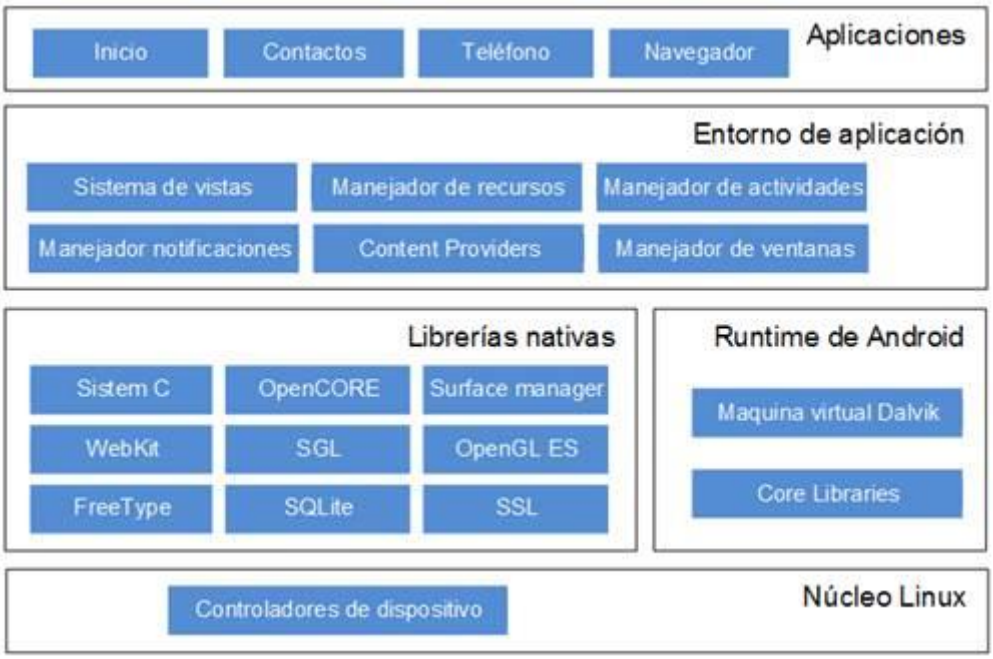


Figura 2: Arquitectura de Android.

5.3.1. El núcleo Linux

El núcleo de Android está formado por el **sistema operativo Linux versión 2.6**. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos.

Esta capa del modelo actúa como **capa de abstracción entre el hardware y el resto de la pila**. Por lo tanto, es la única que es dependiente del hardware.

5.3.2. Runtime de Android

Está **basado** en el concepto de **máquina virtual** utilizado en Java. Dado las limitaciones de los dispositivos donde ha de correr Android (poca memoria y procesador limitado) no fue posible utilizar una máquina virtual Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones. A partir de Android L se utilizará la máquina virtual ART.

Algunas características de la máquina virtual Dalvik que facilitan esta optimización de recursos son: que ejecuta ficheros Dalvik ejecutables (.dex) –formato optimizado para

ahorrar memoria. Además, está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Delega al kernel de Linux algunas funciones como threading y el manejo de la memoria a bajo nivel.

También se **incluye** en el Runtine de Android el "**core libraries**" con la mayoría de las librerías disponibles en el lenguaje Java.

5.3.3. Librerías Nativas

Incluye un **conjunto de librerías en C/C++** usadas en varios componentes de Android. Están compiladas en código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto. Algunas de estas librerías son:

- **System C library:** una derivación de la librería BSD de C estándar (libc), adaptada para dispositivos embebidos basados en Linux.
- **Media Framework:** librería basada en PacketVideo's OpenCORE; soporta codecs de reproducción y grabación de multitud de formatos de audio vídeo e imágenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.
- **Surface Manager:** maneja el acceso al subsistema de representación gráfica en 2D y 3D.
- **WebKit/Chromium:** soporta el navegador Web utilizado en Android y en la vista Webview. En la versión 4.4 WebKit ha sido reemplazado por Chromium/Blink, que es la base del navegador Chrome de Google.
- **SGL:** motor de gráficos 2D.
- **Librerías 3D:** implementación basada en OpenGL ES 1.0 API. Las librerías utilizan el acelerador hardware 3D si está disponible, o el software altamente optimizado de proyección 3D.
- **FreeType:** fuentes en bitmap y renderizado vectorial.
- **SQLite:** potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.
- **SSL:** proporciona servicios de encriptación Secure Socket Layer.

5.3.4. Entorno de Aplicación

Proporciona una **plataforma de desarrollo libre** para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones,).

Esta capa ha sido diseñada para simplificar la **reutilización de componentes**. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas (sujetas a las restricciones de seguridad). Este mismo mecanismo permite a los usuarios reemplazar componentes.

Los servicios más importantes que incluye son:

- **Views**: extenso conjunto de vistas, (parte visual de los componentes).
- **Resource Manager**: proporciona acceso a recursos que forman parte de la aplicación y que están fuera del código.
- **Activity Manager**: maneja el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.
- **Notification Manager**: permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.
- **Content Providers**: mecanismo sencillo para acceder a datos de otras aplicaciones (como los contactos).

Una de las mayores fortalezas del entorno de aplicación de Android es que se **aprovecha el lenguaje de programación Java.** El SDK de Android no acaba de ofrecer todo lo disponible para su estándar del entorno de ejecución Java (JRE), pero es compatible con una fracción muy significativa de la misma.

5.3.5. Aplicaciones

Este nivel está formado por el **conjunto de aplicaciones instaladas en una máquina Android**. Todas las aplicaciones **han de correr en la máquina virtual** Dalvik para garantizar la seguridad del sistema.

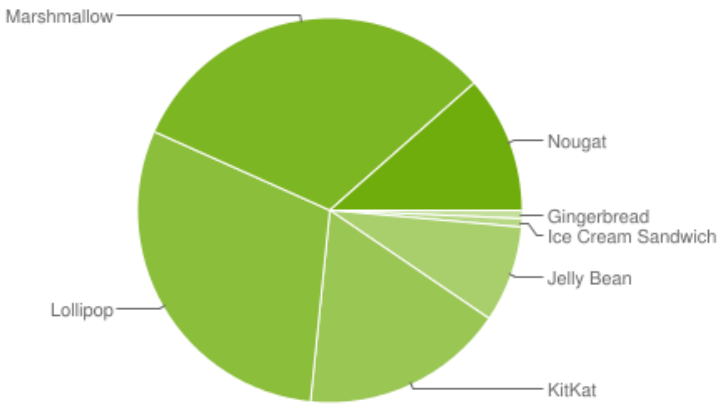
Normalmente las aplicaciones Android están escritas en Java. Para desarrollar aplicaciones en Java podemos utilizar el Android SDK. Existe otra opción consistente en desarrollar las aplicaciones utilizando C/C++. Para esta opción podemos utilizar el Android NDK (Native Development Kit).

5.4. Versiones Android

A continuación se muestra una fotografía actualizada de la **distribución de las versiones de Android.** Esta información es de vital importancia para el desarrollo de una aplicación, ya que puede marcar que los potenciales clientes aumenten o disminuyan de forma en función de las decisiones de desarrollo tomadas.

Versión	Nombre	API	Distribución	Distribución acumulada
2.3.3 - 2.3.7	Gingerbread	10	0.7%	100,00%
4.0.3 4.0.4	Ice Cream Sandwich	15	0.7%	99,30%
4.1.x	Jelly Bean	16	2.8%	98,60%
4.2.x		17	4.1%	95,80%
4.3		18	1.2%	91,70%
4.4	KitKat	19	17.1%	90,50%
5.0	Lollipop	21	7.8%	73,40%
5.1		22	22.3%	65,60%
6.0	Marshmallow	23	31.8%	43,30%
7.0	Nougat	24	10.6%	11,50%
7.1		25	0.9%	0,90%

A continuación se muestra la información representada en un gráfico circular:



Datos recopilados durante un período de 7 días hasta 6/7/2017.
No se muestran versiones con una distribución inferior al 0,1%.

5.5. Entorno de desarrollo. Android Studio

Android Studio es el **entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android** y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android, como las siguientes:

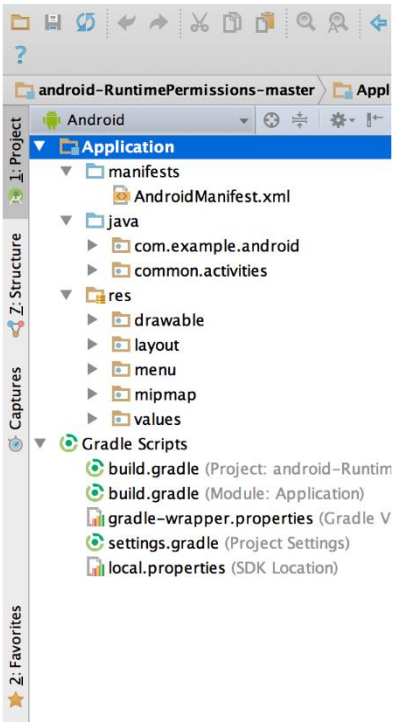
- Un sistema de compilación **basado en Gradle** flexible
- Un **emulador rápido** con varias funciones
- Un **entorno unificado** en el que puedes realizar desarrollos para todos los dispositivos Android
- **Instant Run** para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK
- **Integración** de plantillas de código y **GitHub** para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código
- **Gran cantidad de herramientas y frameworks** de prueba
- **Herramientas Lint** para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- **Compatibilidad con C++ y NDK**
- **Soporte** incorporado para **Google Cloud Platform**, lo que facilita la integración de Google Cloud Messaging y App Engine

5.5.1. Estructura del proyecto

De manera predeterminada, Android Studio muestra los archivos de tu proyecto en la vista de proyectos de Android, como se muestra en la siguiente figura. Esta vista se organiza en módulos para proporcionar un rápido acceso a los archivos de origen clave de tu proyecto.

Todos los archivos de compilación son visibles en el nivel superior de **Secuencias de comando de Gradle** y cada módulo de la aplicación contiene las siguientes carpetas:

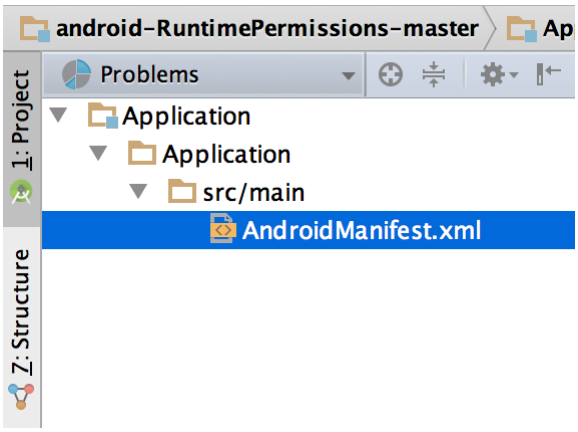
- **manifests**: contiene el archivo `AndroidManifest.xml`.
- **java**: contiene los archivos de código fuente de Java, incluido el código de prueba JUnit.
- **res**: Contiene todos los recursos, como diseños XML, cadenas de IU e imágenes de mapa de bits.



Archivos del proyecto en la vista de Android.

La estructura del proyecto para Android en el disco difiere de esta representación plana. Para ver la estructura de archivos real del proyecto, selecciona **Project** en la lista desplegable **Project** (en la figura anterior se muestra como **Android**).

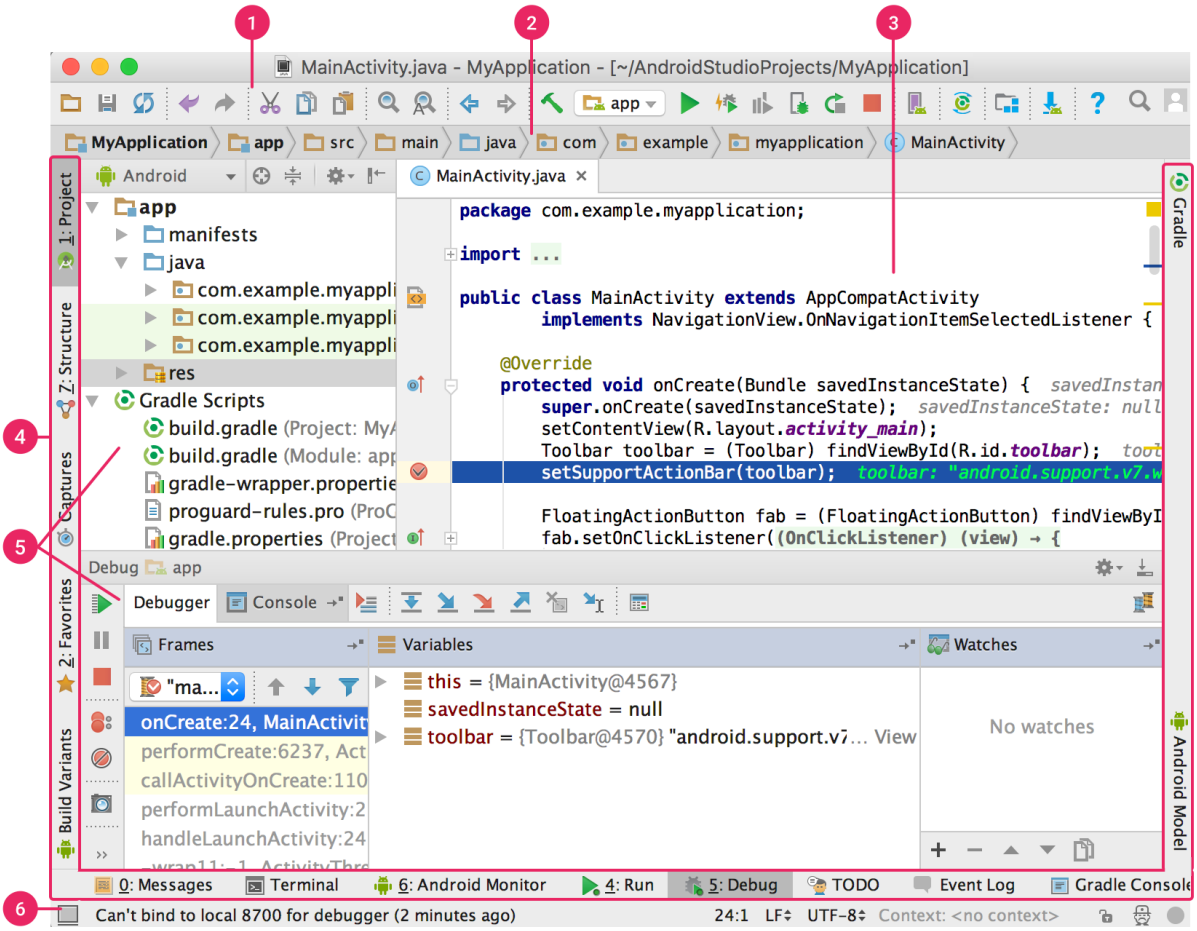
También puedes personalizar la vista de los archivos del proyecto para concentrarte en aspectos específicos del desarrollo de tu app. Por ejemplo, al seleccionar la vista **Problems** de tu proyecto, aparecerán enlaces a los archivos de origen que contengan errores conocidos de codificación y sintaxis, como una etiqueta de cierre faltante para un elemento XML en un archivo de diseño.



Archivos del proyecto en la vista Problems, en
la que se muestra un archivo de diseño con un problema.

5.5.2. Interfaz de usuario

La ventana principal de Android Studio consta de varias áreas lógicas que se identifican en la siguiente figura:



Ventana principal de Android Studio.

1. La **barra de herramientas** te permite realizar una gran variedad de acciones, como la ejecución de tu app y el inicio de herramientas de Android.

2. La **barra de navegación** te ayuda a explorar tu proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana **Project**.

3. La **ventana del editor** es el área donde puedes crear y modificar código. Según el tipo de archivo actual, el editor puede cambiar. Por ejemplo, cuando se visualiza un archivo de diseño, el editor muestra el editor de diseño.

4. La **barra de la ventana de herramientas** se extiende alrededor de la parte externa de la ventana del IDE y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.

5. Las **ventanas de herramientas** te permiten acceder a tareas específicas, como la administración de proyectos, las búsquedas, los controles de versión, etc. Puedes expandirlas y contraerlas.

6. En la **barra de estado**, se muestra el estado de tu proyecto y del IDE en sí, como también cualquier advertencia o mensaje.

Puedes organizar la ventana principal para tener más espacio en pantalla ocultando o desplazando barras y ventanas de herramientas. También puedes usar combinaciones de teclas para acceder a la mayoría de las funciones del IDE.

En cualquier momento, puedes realizar búsquedas en tu código fuente, bases de datos, acciones, elementos de la interfaz de usuario, etc., presionando dos veces la tecla Shift o haciendo clic en la lupa que se encuentra en la esquina superior derecha de la ventana de Android Studio. Esto puede ser muy útil, por ejemplo, si intentas localizar una acción específica del IDE que olvidaste cómo activar.

Autocompletado de código

Android Studio ofrece tres opciones para **completar código**, a las que puedes acceder usando combinaciones de teclas.

Combinaciones de teclas para completar código.

Tipo	Descripción	Windows y Linux	Mac
Autocompletado básico	Muestra sugerencias básicas para variables, tipos, métodos y expresiones, entre otros. Si llamas a la compleción básica dos veces seguidas, verás más resultados. Entre otros, miembros privados y miembros estáticos sin importar.	**Control + Espacio**	**Control + Espacio**
Autocompletado inteligente	Muestra opciones relevantes en función del contexto. La compleción inteligente reconoce el tipo y los flujos de datos previstos. Si llamas a la compleción inteligente dos veces seguidas, verás más resultados. Por ejemplo, cadenas.	**Control + Mayús + Espacio**	**Control + Mayús + Espacio**
Autocompletado de enunciados	Completa el enunciado actual por usted agregando elementos faltantes, como paréntesis, corchetes, llaves, formato, etc.	**Control + Mayús + Enter**	**Mayús + Comando + Enter**

También puedes realizar correcciones rápidas y mostrar acciones de intención presionando **Alt+Enter**.

Aspectos básicos del control de versión

Android Studio **admite** diferentes **sistemas de control de versión (VCS)**, incluidos Git, GitHub, CVS, Mercurial, Subversion y Google Cloud Source Repositories.

Después de importar tu app a Android Studio, usa las opciones del menú del VCS de Android Studio a fin de habilitar la compatibilidad con VCS para el sistema de control de versión deseado, crea un repositorio, importa los nuevos archivos al control de versión y realiza otras operaciones de control de versión.

En el menú del VCS ahora se muestran diversas opciones de control de versión según el sistema que hayas seleccionado.

Editor de diseño

Cuando se trabaja con archivos de **diseño XML**, Android Studio **ofrece un editor visual** con la función arrastrar y colocar, que facilita como nunca la creación de nuevos diseños. El editor de diseño se creó junto con la API ConstraintLayout, por lo que puedes compilar rápidamente un diseño que se adapte a diferentes tamaños de pantalla arrastrando vistas al lugar correcto y, luego, agregando restricciones de diseño con solo unos pocos clics.

Analizador de APK

Puedes usar el **analizador de APK** para inspeccionar fácilmente el contenido de tu APK. Revela el tamaño de cada componente para poder identificar formas de reducir el tamaño total del APK. Además, te permite obtener una vista previa de los recursos empaquetados, inspeccionar los archivos DEX para solucionar problemas de MultiDex y comparar las diferencias entre dos APK.

Estudios de recursos vectoriales y de imagen

Android Studio facilita la creación de un nuevo **recurso** de **imagen** para cada densidad. Con **Vector Asset Studio**, puedes seleccionar íconos de material design proporcionados por Google o importar un archivo SVG o PSD. Gracias a **Image Asset Studio**, también se pueden generar archivos de mapa de bits para cada densidad de pantalla a fin de admitir versiones de Android más antiguas que no admitan el formato de elementos de diseño vectoriales de Android.

Editor de traducciones

Translations Editor te proporciona una **vista única** de todos tus **recursos traducidos**, lo cual facilita la modificación o adición de traducciones, y la localización de traducciones faltantes sin abrir cada versión del archivo strings.xml. Proporciona, incluso, un vínculo para pedir servicios de traducción.

5.5.3. Sistema de compilación Gradle

Android Studio usa Gradle como la base del sistema de compilación, con más capacidades específicas de Android a través del complemento de Android para Gradle. Este sistema de compilación se ejecuta en una herramienta integrada desde el menú de Android Studio, y lo hace independientemente de la línea de comandos. Puedes usar las funciones del sistema de compilación para lo siguiente:

- personalizar, configurar y extender el proceso de compilación;
- crear múltiples APK para tu app, con diferentes funciones utilizando el mismo proyecto y los mismos módulos;
- volver a usar códigos y recursos entre conjuntos de archivos de origen.

Recurriendo a la flexibilidad de Gradle, puedes lograr todo esto sin modificar los archivos de origen de tu app. Los archivos de compilación de Android Studio se denominan build.gradle. Son archivos de texto sin formato que usan la sintaxis Groovy para configurar la compilación con elementos proporcionados por el complemento de Android para Gradle. Cada proyecto tiene un archivo de compilación de nivel superior para todo el proyecto y archivos de compilación de nivel de módulo independientes para cada módulo. Cuando importas un proyecto existente, Android Studio genera automáticamente los archivos de compilación necesarios.

Variantes de compilación

El sistema de compilación puede ayudarte a crear **diferentes versiones de la misma aplicación** a partir de un solo proyecto. Esto resulta útil cuando tienes una versión gratuita o una versión pago de tu app, o si quieres distribuir múltiples APK para diferentes configuraciones de dispositivos en Google Play.

Divisiones de APK

La **división de APK** te permite crear de forma eficiente **varios APK en función** de la **densidad de la pantalla o ABI**. Por ejemplo, la división de APK te permite crear versiones hdpi y mdpi independientes para una app sin dejar de considerarlas como una variante individual y permitiéndoles compartir la configuración de una app de prueba, javac, dx y ProGuard.

Reducción de recursos

La **reducción de recursos** en Android Studio **elimina automáticamente los recursos sin usar** del paquete de tu app y de las dependencias de bibliotecas. Por ejemplo, si en tu aplicación se usan servicios de Google Play para acceder a la funcionalidad de Google

Drive y actualmente no usas Google Sign-In, la reducción de recursos puede eliminar los diferentes recursos de elemento de diseño de los botones SignInButton.

Nota: La reducción de recursos funciona con herramientas de reducción de código, como ProGuard.

Administración de dependencias

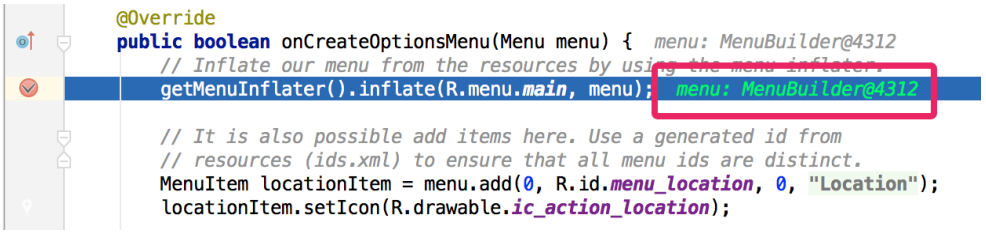
Las **dependencias** para tu proyecto **se especifican** por nombre en el archivo **build.gradle**. Gradle se ocupa de buscar tus dependencias y hacer que estén disponibles en tu compilación. Puedes declarar dependencias de módulos, dependencias binarias remotas y dependencias binarias locales en tu archivo build.gradle. Android Studio configura los proyectos para que usen el repositorio central de Maven de forma predeterminada. (Esta configuración está incluida en el archivo de compilación de nivel superior del proyecto).

5.5.4. Herramientas de depuración

Android Studio te ayuda a **depurar y mejorar el rendimiento** de tu **código**. Esto incluye **herramientas integradas de depuración y análisis de rendimiento**.

Depuración integrada

Usa la depuración integrada para **mejorar las revisiones de código** en la vista del depurador con verificación integrada de referencias, expresiones y valores de variables.



Valor de una variable integrada.

Monitores de rendimiento

Android Studio proporciona **monitores de rendimiento** para que puedas realizar de manera más sencilla un **seguimiento del uso de CPU y memoria** de tu **app**, buscar objetos sin asignar, **localizar pérdidas de memoria**, optimizar el rendimiento de los gráficos y analizar solicitudes de la red. Con tu app ejecutándose en un dispositivo o emulador, abre la ventana de herramientas **Android Monitor** y haz clic en la pestaña **Monitors**.

Seguimiento de asignación de memoria

Android Studio te **permite realizar** un **seguimiento** de la **asignación de memoria** mientras controla el uso de esta. El seguimiento de la asignación de memoria te permite controlar

dónde se asignan los objetos cuando realizas ciertas acciones. Conocer estas asignaciones te permite optimizar el rendimiento de tu app y el uso de la memoria ajustando las llamadas del método relacionadas con las acciones en cuestión.

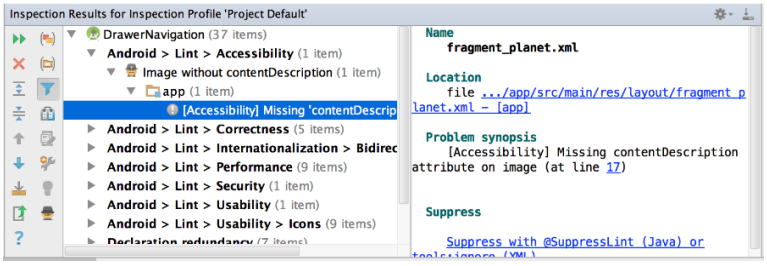
Monitores de rendimiento

Las **herramientas del Android SDK**, como **Systrace, logcat y Traceview** generan **datos de rendimiento y depuración para un análisis detallado de la app.**

Inspecciones de código

Cuando compilas tu programa, **Android Studio ejecuta automáticamente inspecciones de Lint** y otras inspecciones de IDE configuradas para ayudarte a **identificar** y corregir problemas fácilmente con respecto a la **calidad** estructural de tu **código**.

La herramienta Lint verifica los archivos de origen de tu proyecto Android para detectar posibles errores y mejoras de optimización en relación con la corrección, la seguridad, el rendimiento, el uso, la accesibilidad y la internacionalización.



Resultados de una inspección de Lint en Android Studio.

Además de las verificaciones de Lint, Android Studio también realiza inspecciones de código de IntelliJ y valida anotaciones para simplificar tu flujo de trabajo de codificación.

Mensajes de log

Cuando compilas y ejecutas tu app con Android Studio, puedes ver mensajes adb de salida y mensajes de registro del dispositivo (logcat) haciendo clic en **Android Monitor** en la parte inferior de la ventana.

Si quieres depurar tu app con el Monitor de dispositivos Android, puedes iniciar el Monitor de dispositivos haciendo clic en **Tools** > **Android** > **Android Device Monitor**. En el Monitor de dispositivos encontrarás el conjunto completo de herramientas DDMS para perfilar tu app, controlar comportamientos del dispositivo y más. En este también se incluye la herramienta del Visor de jerarquía para ayudarte a optimizar tus diseños.

5.6. Principales clases y librerías empleadas

Una vez definidas las clases de la aplicación, se considera necesario **describir** algunas de las **principales clases y librerías empleadas**, a las que en algunos casos se ha hecho referencia anteriormente, o bien son conceptos fundamentales en Android para cualquier aplicación.

Vista (View)

Las vistas son los **elementos** que **componen** la **interfaz de usuario** de una aplicación. Son por ejemplo un botón, una entrada de texto, etc. Todas las vistas van a ser objetos descendientes de la clase View, y por tanto, pueden ser definidos utilizando Java. Sin embargo, lo **habitual** va a ser **definir** las **vistas** utilizando un **fichero XML** y dejar que el sistema cree los objetos por nosotros a partir de este fichero. Esta forma de trabajar es muy similar a la definición de una página web utilizando código HTML.

Layout

Un **layout** es un **conjunto** de **vistas agrupadas** de una determinada forma. Se dispone de diferentes tipos de layouts para organizar las vistas de forma lineal, en cuadrícula o bien indicando la posición actual en cada vista. Los layouts también son objetos descendientes de la clase View. Igual que las vistas los layouts pueden ser **definidos** en código, aunque lo habitual es hacer en **código XML**.

Activity

Una aplicación en Android va a estar formada por un conjunto de elementos básicos de visualización, coloquialmente conocidos como **pantallas de la aplicación**. En Android cada uno de estos elementos, o pantallas se conoce como **actividad**. Su función principal es la **creación del interfaz de usuario**. Una actividad suele necesitar de varias actividades para crear el interfaz de usuario. Las diferentes actividades creadas serán independientes entre sí, aunque todas trabajarán con un objetivo común. Toda actividad ha de pertenecer a una clase descendiente de activity.

Intención (Intent)

Una intención representa la **voluntad de realizar alguna acción**: como realizar una llamada de teléfono o abrir Google Maps por ejemplo. Será **necesario** utilizarlos cada ver que **queramos lanzar una actividad**. Los componentes lanzados pueden ser internos o externos a nuestra aplicación. **También utilizaremos** las **intenciones** para el **intercambio de información** entre estos componentes.

Fragment

La llegada de las tablets trajo el problema de que las aplicaciones de Android ahora deben soportar pantallas más grandes. Si hacemos un diseño de aplicación pensada para un dispositivo móvil y luego se ejecuta en una Tablet, el resultado no suele ser satisfactorio.

Para ayudar al diseñador a resolver este problema, en la versión 3.0 de Android aparecen los fragments. Un **fragment** está formado por la **unión de varias vistas** para **crear** un **bloque funcional del interfaz de usuario**. Una vez creados los fragments dentro de una actividad, según el tamaño de pantalla disponible.

ListView

Una vista ListView, **visualiza** una **lista deslizable de varios elementos** con la misma estructura, donde **cada** elemento **puede definirse como un layout**.

Definir un ListView conlleva los siguientes cuatro pasos:

- Diseñar un Layout que contenga al Listview
- Diseñar un layout individual que se repetirá en la lista
- Implementar una actividad que visualice el layout con el ListView
- Personalizar cada uno de los layout individuales según nuestros datos.

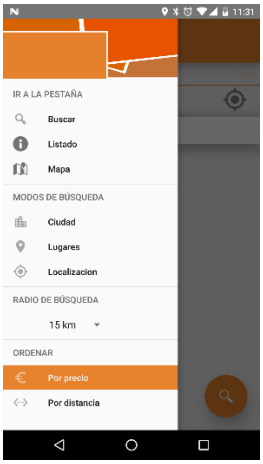
SwipeRefreshLayout

El patrón de diseño Swipe To Refresh, más conocido como Pull To Refresh, puede ser considerado como el mecanismo estándar en aplicaciones móviles para actualizar el contenido de un listado o grid de datos. La actualización de los datos se produciría al deslizar hacía abajo, una vez alcanzado el límite superior del listado.

Google incluye este patrón en Material Design y proporciona, a través de la librería de compatibilidad, un ViewGroup denominado **SwipeRefreshLayout** para implementar esta funcionalidad. SwipeRefreshLayout será el padre de un View deslizable, como por ejemplo ListView, ScrollView o RecyclerView, y mediante un listener nos informará del gesto Swipe To Refresh para que actuemos en consecuencia. También proporciona un indicador de la actualización.

NavigationDrawer

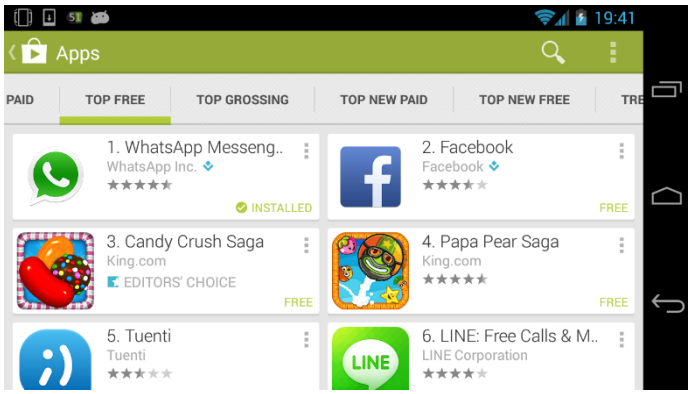
Navigation Drawer o **panel lateral de navegación**, es un panel en el que **se muestran las principales opciones de navegación de la app** en el borde izquierdo de la pantalla. La mayor parte del tiempo está oculto, pero aparece cuando el usuario desliza un dedo desde el borde izquierdo de la pantalla o, cuando el usuario toca el ícono de la app en la barra de acciones.



ViewPager

Un **ViewPager** es un **ViewGroup** que **permite desplazarnos** por **distintos layouts** o "páginas" dentro de una misma Activity simplemente **deslizando las páginas** a la derecha o izquierda. Este widget no está presente en la API de Android y se encuentra disponible en la librería de compatibilidad lo que permite su utilización en cualquier versión de Android.

Un ejemplo clásico de ViewPager, se podía encontrar en versiones anteriores de Google Play, para navegar entre los rankings de las aplicaciones:



ViewPageIndicator

Es una **librería** "open source" con licencia Apache 2.0, desarrollada por el programador de Android Jake Wharton. **Permite conocer en que página del ViewPager nos encontramos** y/o si hay páginas adyacentes a las que se pueda navegar.

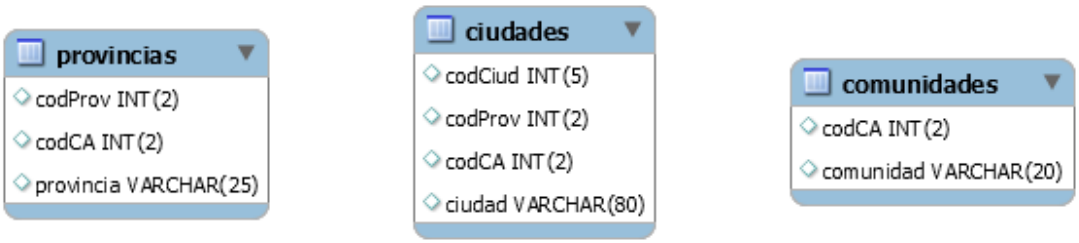
SQLite

Las bases de datos son una herramienta de gran potencia en la creación de herramientas informáticas. Hasta hace muy poco resultaba costoso y complejo utilizar bases de datos en nuestras aplicaciones. No obstante, **Android incorpora** la librería **SQLite**, que nos permitirá utilizar bases de datos mediante el lenguaje SQL, de una forma sencilla y

utilizando muy pocos recursos del sistema. SQL es el lenguaje de programación más utilizado para bases de datos.

Para manipular una base de datos en Android usaremos la clase abstracta SQLiteOpenHelper que nos facilita tanto la creación automática de la base de datos, como el trabajar con futuras versiones de la base de datos.

En el caso que nos ocupa la base de datos se emplea para almacenar las ciudades, provincias y CCAA y poder realizar las consultas con las relaciones entre ellas. El esquema que resume el diseño de la BBDD sería el siguiente:



Volley

Es una **librería desarrollada por Google** para **optimizar** el envío de **peticiones HTTP** desde las aplicaciones Android hacia servidores externos. Permite facilitar el manejo de estas peticiones y lo más importante **hacerlas más rápidas**.

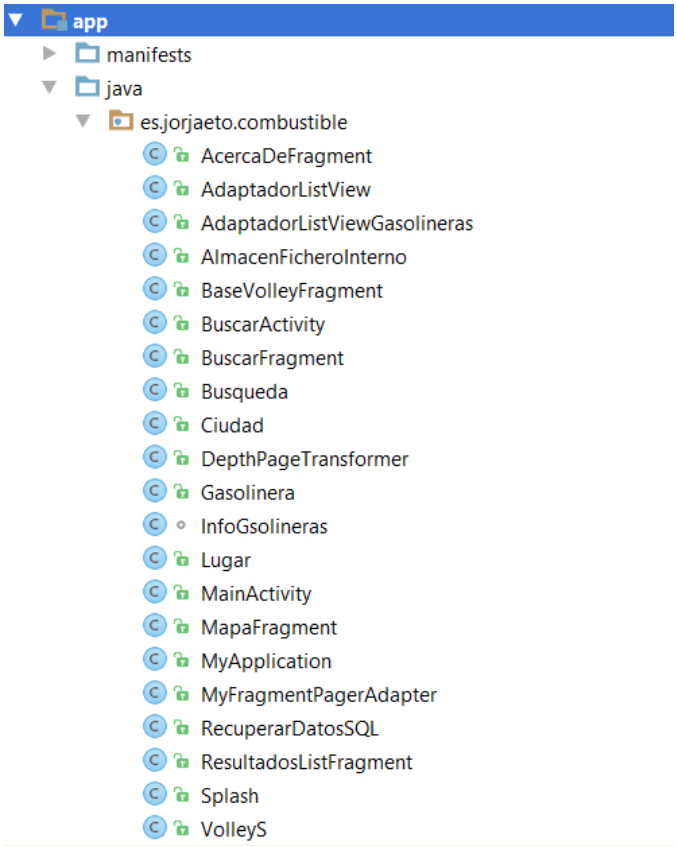
No pertenece al API de Android, pero ha sido desarrollada por Google, por lo que es posible que sea incluida en un futuro.

Presenta las siguientes ventajas:

- **Gestión automática de hilos**: No es necesario crear nuevos hilos o AsyncTasck de forma manual. Solo se deberá programar el escuchador adecuado cuando se produzca la descarga.
- **Caché transparente**: Las descargas son guardadas de forma automática en disco o memoria. Si se solicita un contenido ya descargado, la respuesta será inmediata. La caché es manejada gracias a las cabeceras del protocolo HTTP (Last-Modified, If-Modified-Since, …).
- **Manejo automático de colas de petición con prioridades**: Volley ha sido diseñada para realizar múltiples descargar simultáneas, pero no se recomienda su uso para la descarga de grandes volúmenes de datos. En este caso es más interesante usar la clase DownloadManager.

5.7. Clases desarrolladas para la aplicación

En la siguiente ventana de Android Studio, en la vista de proyectos de Android, pueden observarse las clases Java que forman el código de la aplicación.



A modo de resumen se describirá la funcionalidad principal de cada una de las clases:

AcercaDeFragment: clase que extiende o es hija de Fragment, Se encargará de mostrar el layout correspondiente.

AdaptadorListView: corresponde al adaptador personalizado (por ello hereda de BaseAdapter) del Listview que mostrará las ciudades y provincias en BuscarActivity.

AdaptadorListViewGasolineras: adaptador personalizado (hereda de BaseAdapter) del Listview que mostrará los resultados en ResultadosListFragment.

AlmacenFicheroInterno: clase encargada de guardar y recuperar los datos correspondientes a la última búsqueda realizada, en el almacenamiento interno.

BaseVolley: clase hija de Fragment, que servirá como base para la llamada al servicio web, utiliza la clase VolleyS. Se describirá con más detalle posteriormente.

BuscarActivity: activity que extiende de ListActivity, y que emplea la clase RecuperarDatosSQL para mostrar los datos del listado de provincias o el de ciudades correspondientes a la provincia previamente seleccionada. También nos permite aplicar un filtro que se va estableciendo a medida que tecleamos el nombre de la ciudad buscada. Si escribimos sin seleccionar ninguna provincia, nos filtrará las ciudades de todas las provincias que contengan el texto escrito y si hemos seleccionado previamente una provincia solo las que correspondan a esta.

BuscarFragment: clase que deriva de BaseVolleyFragment y que es la encargada de proporcionar la funcionalidad de búsqueda de la aplicación. Gestiona los diferentes modos de búsqueda, se encarga de guardar y recuperar la última búsqueda mediante AlmacenFicheroInterno, realiza la llamada del web service y comunica al resto de fragments la actualización de datos.

Busqueda: implementa el objeto que representa una búsqueda, contendrá otros objetos definidos en la propia aplicación como ciudad y lugar así como otras variables que definirán la búsqueda como por ejemplo el tipo de combustible.

Ciudad: representa la entidad ciudad y que contiene un objeto provincia definido en el mismo archivo .java.

DepthPageTransformer: clase que implementa ViewPager.PageTransformer para crea un efecto visual al movernos entre las pestañas de la aplicación.

Gasolinera: permitirá crear los objetos gasolinera, con todas las variables y métodos necesarios para su definición y manipulación. Implementa Comparable para poder ordenar el vector <Gasolineras> por precio y por distancia.

InfoGasolineras: personaliza el cuadro de información de los marcadores que representarán la ubicación de las gasolineras en MapaFragment. Permite que aparezca toda la información deseada, y con el formato propio definido.

Lugar: representa la entidad Lugar, definida por un nombre y una posición latitud, longitud para el correcto funcionamiento en el modo Lugares.

MainActivity: contiene el código que define la activity principal. Hereda de FragmentActivity e implementa NavigationView.OnNavigationItemSelectedListener ya que incluye Navigation Drawer y debemos detectar los cambios en este. También implementa ActualizacionDeDatos y CargaBusquedaCompleta que son callbacks definidos en BuscarFragment para notificar al MainActivity que se han producido cambios y que realice las operaciones que correspondan. Contiene el ViewPager que contendrá al resto de Fragments.

MapaFragment: extiende de la clase Fragment e implementa OnMapReadyCallback necesario para realizar las operaciones sobre el mapa, cuando este se encuentre preparado. También implementa GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener, LocationListener que nos permitirá dar la funcionalidad de ubicación en la aplicación (previa solicitud de permisos en tiempo de ejecución).
Como aspectos a destacar:

- Define el **vector de gasolineras** que contiene la información obtenida con los criterios de la última búsqueda, así como los métodos necesarios para su manipulación.
- **Marca las gasolineras en el mapa.**
- **Calcula la distancia entre** un **punto** definido por latitud y longitud **y** la **posición actual.**

MyApplication: hereda de la clase Application y se crea para poder hacer uso de la librería ACRA (Application Crash Reports for Android), lo que nos permitirá analizar posibles errores en la aplicación tras su distribución en Google Play.

MyFragmentPageAdapter: personaliza el FragmentPageAdapter y por ello extiende de esta clase. Se emplea para modificar el título de las diferentes páginas del PageViewer.

RecuperarDatosSQL: clase hija de SQLiteOpenHelper, que generará la base de datos de ciudades, provincias y comunidades autónomas (si esta no existe), a partir de un fichero import presente en recursos. Y será la encargada de recuperar los datos para mostrarlos en la activity Buscar, de acuerdo a las solicitudes del usuario.

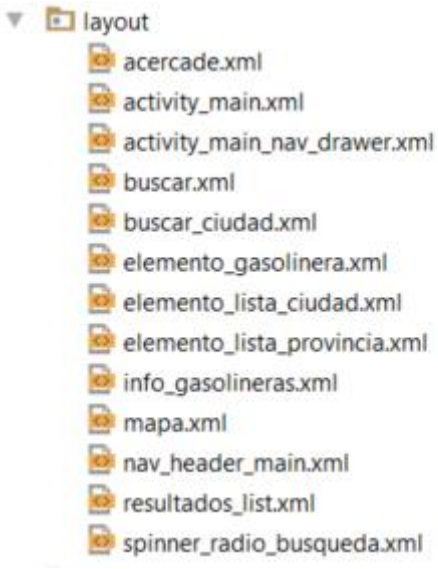
ResultadosListFragment: Muestra los resultados de la búsqueda en un Listview, por ello hereda de ListFragment. Contiene un objeto SwipeRefreshLayout, para permitir actualizar los datos de acuerdo a los criterios de búsqueda actuales, simplemente deslizando hacia abajo. Al pulsar sobre uno de los resultados de búsqueda, se lanzará Google Maps con la posición de la estación de servicio.

Splash: Activity de tipo splash que muestra durante un segundo el logo de la aplicación y a continuación lanza el MainActivity.

VolleyS: Clase que utiliza la librería Volley de Android, para crear una única instancia de sí misma y por tanto una única de cola de peticiones que define, como aconseja el propio Android en una aplicación de este tipo. Esta clase es empleada por BaseVolleyFragment.

5.8. Layouts diseñados

En la siguiente ventana de Android Studio, en la vista de proyectos de Android, pueden observarse en la carpeta **layout**, todos los layouts diseñados para la aplicación.



acercade: layout formado únicamente por un RelativeLayout y un TextView que mostrará el logo de la aplicación. Se empleará tanto en la activity Splash que se mostrará 1 segundo al abrir la aplicación, como en el AcercaDeFragment que se añadirá en la página 0 del ViewPager.

activity_main: corresponde con el layout principal sin el navigation drawer. Es el diseño inicial de la pantalla principal. Un linear layout contiene el Toolbar, un TitlePageIndicator que nos servirá para indicar el título de las páginas y por último el ViewPager.

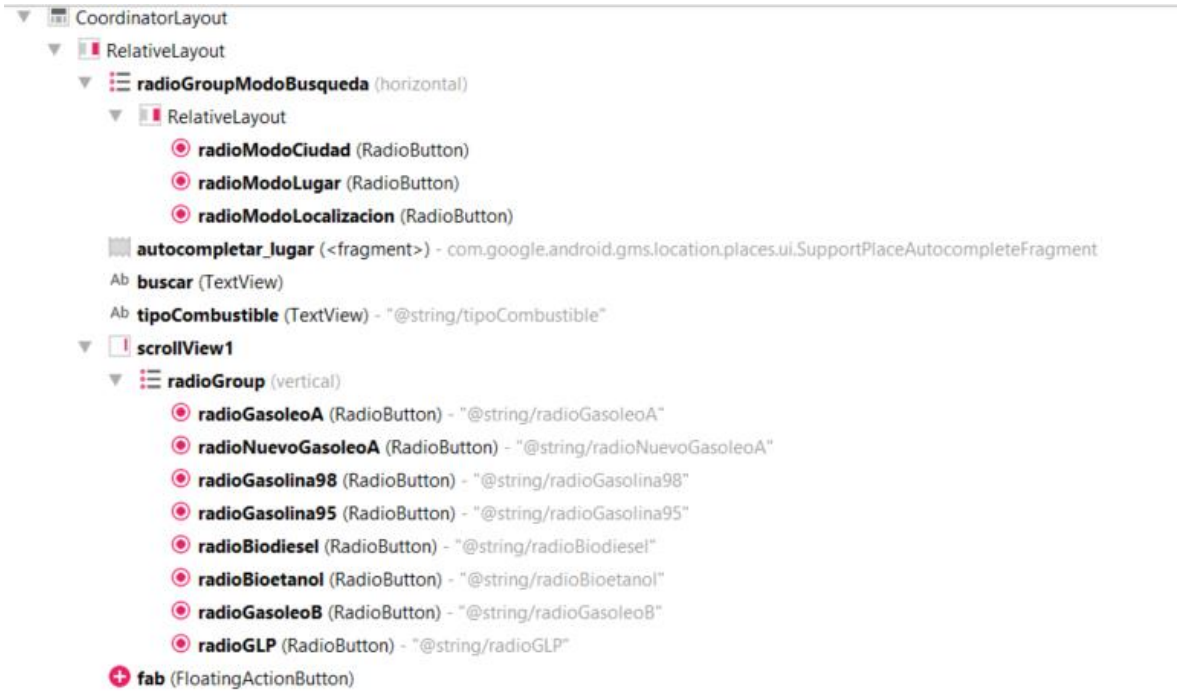
activity_main_nav_drawer: drawer layout que contiene el layout anterior, **activity_main** y el widget NavigationView donde se indica el layout que corresponde con la cabecera del Navigation Drawer y el menú con los ítems que formarán el contenido del panel lateral.

buscar: es uno de los más complejos en diseño de la aplicación, por el número de elementos y para conseguir buenos resultados con un único diseño para varios tamaños de pantallas. El contenedor principal en este caso es un coordinator layout que contiene como elementos visuales:

- **RadioGroupModoBusqueda** radioGroup con los radioButtons que formará el menú que gestionará los modos de búsqueda.
- Fragment **autocompletar_kugar** para implementar la posibilidad de buscar cualquier calle, punto de interés o ciudad de España.

- TextView **buscar** donde se reflejará el destino de búsqueda seleccionado en cada momento.
- **RadioGroup** que contendrá los radioButtons para escoger el tipo de combustible deseado. Se incrusta en un ScrollView para que sean visibles en todos los tamaños de pantalla.
- **FloatingActionButton** en la parte inferior derecha de la pantalla, para implementar la funcionalidad de buscar.

Se aporta el esquema jerárquico del diseño para facilitar la compresión del diseño



buscar_ciudad: diseño que se mostrará en la activity **BuscarActivity** para seleccionar la ciudad como objetivo de búsqueda en el modo ciudad. Está compuesto por un EditText donde podremos escribir y que nos servirá de filtro, y un ListView que mostrará:

- Las provincias si no hemos escrito ni seleccionado ninguna provincia.
- Las ciudades de la provincia seleccionada, en caso de haber pinchado en una.
- Todas las ciudades que contengan una cadena independientemente de la provincia a la que pertenezcan, si nos ponemos a escribir sin haber seleccionado ninguna provincia.
- Las ciudades que pertenezcan a la provincia seleccionada previamente y que contengan la cadena filtro que estamos escribiendo.

elemento_gasolinera: detalle que se mostrará en el ListView del fragment ResultadosListFragment, por cada una de las estaciones de servicio. Formado por TextViews y los LinearLayouts necesarios para los efectos visuales y composición deseada.

Aunque aparentemente parece simple, para llegar al resultado final y que las diferentes casuísticas de tamaños de las cadenas se muestren correctamente ha supuesto un esfuerzo notable. Mostrará los detalles de nombre o rótulo de la estación de servicio, dirección, fecha de actualización del precio, precio, distancia y horario de apertura.

elemento_lista_ciudad: elemento del ListView que mostrará el detalle de cada una de las ciudades mostradas en el activity BuscarActivity. Contiene 2 LinearLayouts anidados y dos TextViews que mostrarán el nombre de la ciudad y la provincia.

elemento_lista_provincia: elemento del ListView que mostrará el detalle de cada una de las provincias mostradas en el activity BuscarActivity. Contiene 1 LinearLayout y un TextView para mostrar el nombre de la provincia.

info_gasolinera: layout que nos permitirá personalizar el cuadro de información que se mostrará en cada una de las gasolineras, al pulsar cada uno de los marcadores que representan las EESS en la página mapa (MapaFragment). Se emplea en la clase **InfoGasolineras.**

mapa: diseño para mostrar el mapa que nos permitirá representar las EESS. Contiene un RelativeLayout como contenedor del **com.google.android.gms.maps.MapView.**

nav_header_main: cabecera del panel lateral deslizante (NavigationDrawer)**.**

resultados_list: layout que mostrará los resultados, empleado en la clase ResultadosListFragment. El contenedor principal es un LinearLayout, que incluirá un TextView donde se reflejará el resumen de la búsqueda realizada o bien las instrucciones si todavía no se ha realizado ninguna búsqueda. Un **android.support.v4.widget.SwipeRefreshLayout** contendrá el ListView con los resultados para permitir la actualización deslizando hacia abajo. Y por último un ImageView con la imagen que se mostrará si no hay resultados.

spinner_radio_busqueda diseño necesario para el spinner que permitirá seleccionar el radio máximo de búsqueda en el menú del NavigationDrawer.

5.9. Recursos creados

Para que el resultado final sea el conseguido, ha sido necesario crear un número de recursos importante, para permitir la personalización de la aplicación.

Se han diseñado los **iconos** con la **herramienta Asset Studio** (Configure Image Asset) de Android Studio.

Mediante el archivo **colors.xml** se han personalizado los colores que se emplean en la aplicación.

Haciendo uso del archivo **dimen.xml** se ha podido ajustar la separación entre los elementos del menú, en el Navigation Drawer.

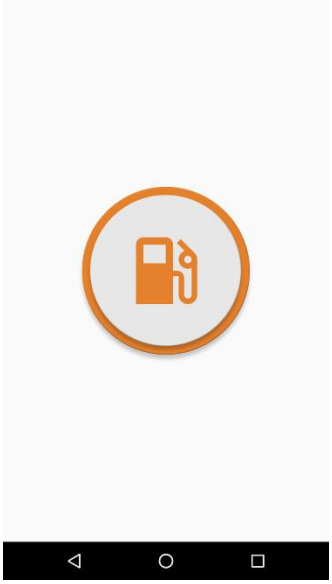
El archivo **strings.xml** centraliza todas las cadenas de texto constantes, facilitando su mantenimiento y traducción a varios idiomas.

El archivo **styles.xml** ha permitido aplicar los estilos haciendo uso de los colores, modificando dimensiones y tamaños de letra.

También ha sido necesario generar un **menú** en la carpeta con ese nombre, que contiene las opciones de acceso rápido y configuración de búsqueda visible desde el Navigation Drawer.

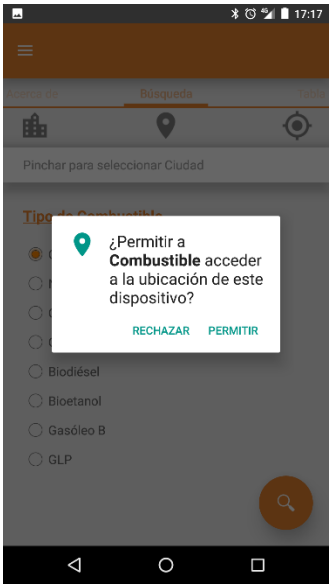
5.10. Manual de la aplicación

Cuando abrimos la aplicación, se muestra el logo de la misma durante 1 segundo y posteriormente se lanza la aplicación principal.



Ventana splash con logo

La primera vez que se abra la aplicación, nos solicitará **permiso** para que esta pueda acceder a la **ubicación del dispositivo** y así ofrecer la funcionalidad completa. Con el permiso facilitado, la ubicación es más precisa y permitirá tanto la búsqueda desde nuestra posición actual, como calcular las distancias entre la posición actual y las estaciones de servicio encontradas.

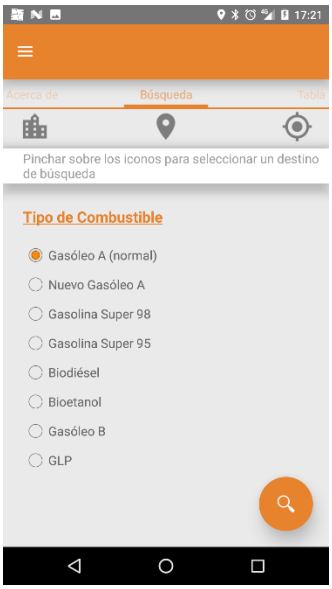


Solicitud de permiso de ubicación

La aplicación se encuentra distribuida en páginas: **Búsqueda**, **Resultados** y **Mapa**, a las que podemos acceder deslizando lateralmente desde las páginas contiguas, pinchando desde los títulos de las mismas, o desde los accesos rápidos del panel lateral de navegación.

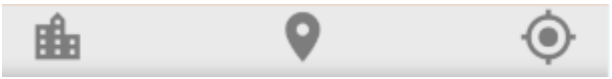
Página Búsqueda

Si todavía no se ha realizado ninguna búsqueda, la pantalla mostrada tras la solicitud del permiso será la página de búsqueda, indicándonos que deberemos pinchar sobre alguno de los tres iconos del menú tipo de búsqueda para seleccionar un destino.



Página de búsqueda sin búsqueda previa

Como se puede apreciar, los tres iconos se encuentran con tono gris y por tanto no está activado ningún modo de búsqueda. Se deberá seleccionar alguno de los tres y escoger un destino.



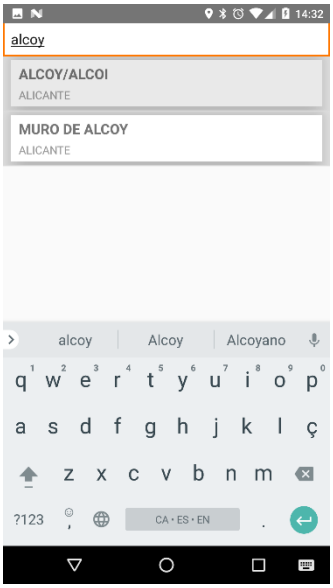
Los iconos representan los siguientes tipos de búsqueda:

- **Modo ciudad**: nos permitirá seleccionar cualquier ciudad de España, y los resultados nos mostrarán las EESS correspondientes al filtro realizado y que se encuentren en ese término municipal. Para ello deberemos pulsar sobre el primer icono con la representación de edificios.
- **Modo lugar**: pinchando sobre el segundo icono, con la marca de posición de un mapa nos permitirá seleccionar cualquier lugar de España mediante el widget autocompletado de sitios del API de Google Places. Nos permite seleccionar cualquier lugar característico, monumentos, calles, ciudades, establecimientos etc
- **Modo ubicación**: seleccionando el icono situado a la derecha seleccionaremos como destino de búsqueda la posición actual.

Modo Ciudad

Cuando pinchemos sobre el icono del **modo ciudad**, nos aparecerá la ventana que nos permitirá **buscar la ciudad** en la que queremos realizar la búsqueda. En primer lugar se mostrará el listado de provincias. Si escribimos en la caja de texto de la parte superior sin haber seleccionado ninguna provincia, nos **filtrará a medida que escribamos** todas las ciudades de España que contengan la cadena que estamos escribiendo. Si se selecciona una provincia, las ciudades mostradas serán las de esa provincia con el filtro que introduzcamos en la caja de texto.

Listado de provincias



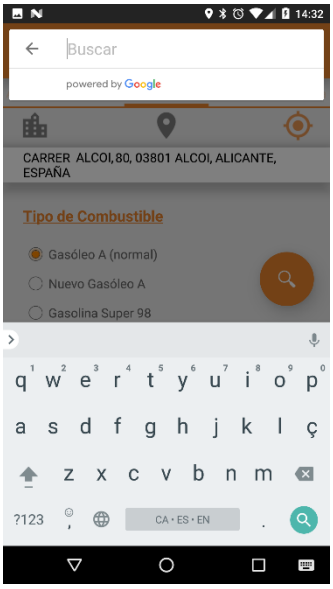
Ejemplo filtro de ciudad

Una vez localizada la ciudad deseada, al pulsar sobre ella, volveremos a la pantalla de búsqueda donde se mostrará el nombre de la misma, y quedará seleccioonado el modo ciudad. 

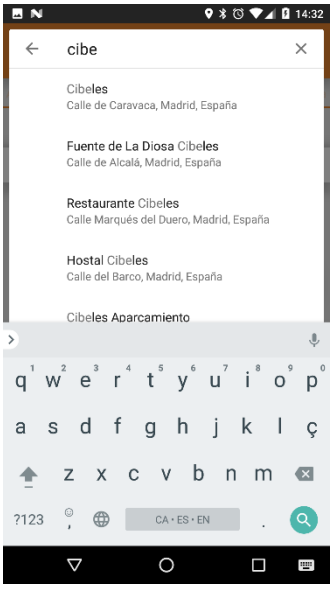
Modo Lugares

Al presionar sobre el menú modo lugares, nos mostrará el widget de google que a medida que escribamos nos **autocompletará** los **sitios** en función del **filtro**.

Hay que tener en cuenta que **en el modo lugares**, los **resultados obtenidos** serán los de las EESS que se encuentren **dentro del radio configurado** (ver apartado Accesos rápidos y configuraciones) **alrededor de la posición del lugar seleccionado**.

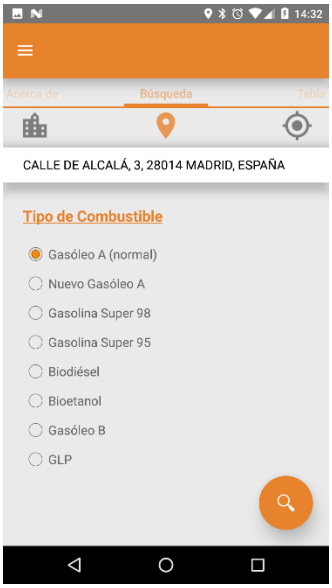


Widget google



Ejemplo filtro sitios

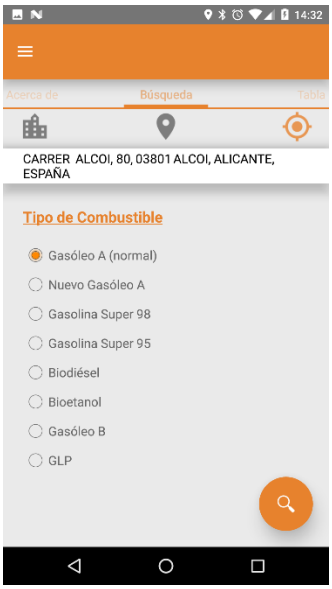
Cuando seleccionemos la opción deseada, volveremos a la pantalla de búsqueda donde aparecerá el nombre del destino seleccionado y quedará activado el modo lugares.



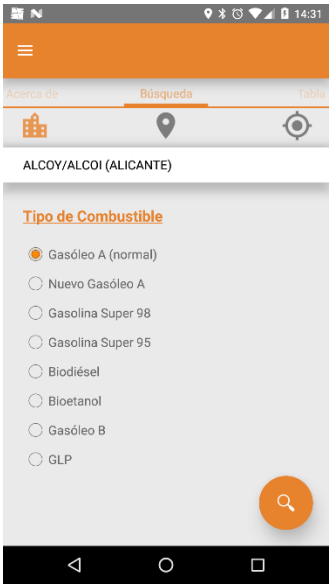
Modo Ubicación

Si el dispositivo dispone del **permiso de ubicación** para la aplicación, **y tiene actualizada la posición actual**, al pinchar sobre el icono del modo ubicación nos indicará en el cuadro de texto de la página buscar la descripción de la posición actual, y quedará seleccionado el modo ubicación.

Hay que tener en cuenta que **en el modo ubicación**, los **resultados obtenidos** serán los de las EESS que se encuentren **dentro del radio configurado** (ver apartado Accesos rápidos y configuraciones) **alrededor de la posición en el momento de la búsqueda**.



Una vez **establecido el destino de búsqueda** desde cualquiera de los tres modos, **seleccionaremos el tipo de combustible**. Con los parámetros de busqueda configurados presionaremos sobre el **botón buscar** en la parte inferior de la ventana.



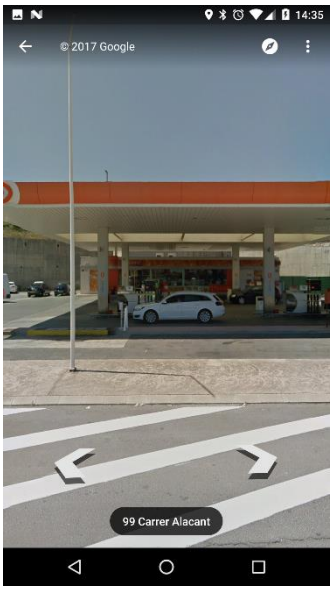
Página Resultados

Si **disponemos de conexión a internet**, se llamará al servicio que **obtendrá los datos y nos llevará a la ventana resultados**, donde podremos ver:

- En la **parte superior** un **resumen de la búsqueda**, donde se indica el tipo de combustible, el destino de la búsqueda y el número de EESS encontradas.
- **A continuación** en forma de **lista**, el detalle de cada una de las **EESS que cumplen con los criterios de búsqueda.** Para cada estación de servicio se mostrará el nombre o **rótulo, dirección, fecha de actualización** de los datos obtenidos, **precio** del producto, **horario** y **distancia** desde la ubación donde se ha realizado la búsqueda a la estación.

Si **pulsamos** sobre el **resumen** de alguna de las **EESS** nos **abrirá Google Maps** con la posición de la estación, y la posibilidad de acceder con un click a la vista en Street View.

Google Maps desde la app



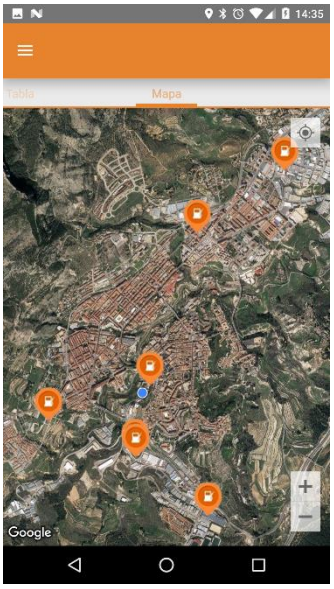
Street View

Si queremos **actualizar** los datos **desde** la página **Resultados**, bastará con **deslizar hacia abajo.**

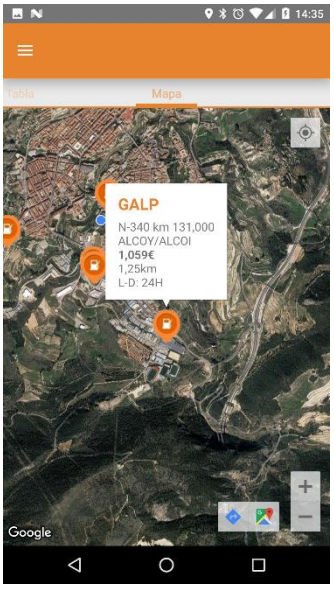
Vista de la actualización al deslizar

Página Mapa

La página Mapa, muestra la **distribución de las EESS sobre un mapa.** Si están cercanas a nuestra ubicación, nos permitirá observar también la posición respecto a esta. Si **pulsamos** sobre alguna de las **marcas** que representan las **estaciones**, aparecerá un **cuadro** donde se **resumen** los **datos** correspondientes a la propia **estación** y al **precio** del producto buscado.



Distribución EESS en mapa

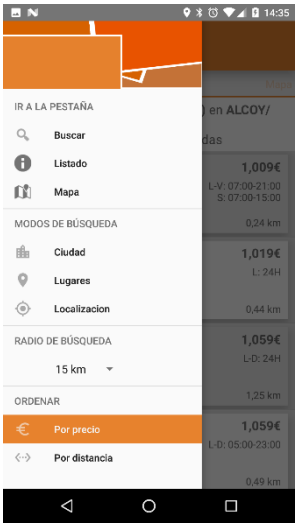


Detalle al pulsar marca

Como se puede observar en la parte inferior de la segunda captura, al seleccionar una estación, aparecen las opciones desde esta página de calcular una ruta a la estación o bien abrir Google Maps con la posición de la gasolinera.

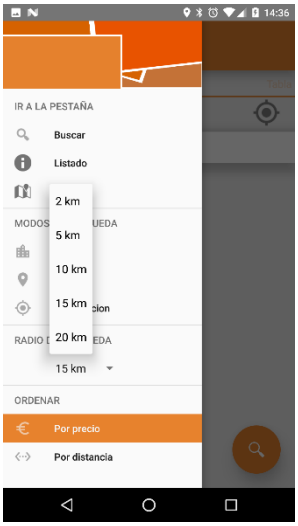
Configuraciones y accesos rápidos

Desde cualquier página de la aplicación podemos **abrir** el **panel lateral**, pulsando sobre las tres rayas blancas de la parte superior izquierda de la aplicación en la barra de herramientas, o bien deslizando lateralmente desde el extremo izquierdo de la ventana hacia el centro.

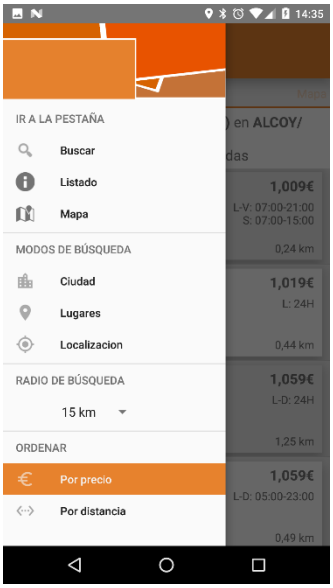


El menú lateral nos permite:

- **Acceder a cualquiera de las páginas**: podremos acceder a cualquiera de las páginas sin que sea la contigua, pulsando sobre el nombre de esta en el menú **IR A LA PESTAÑA**.
- **Lanzar cualquier modo de búsqueda**: pinchando sobre el nombre del modo de búsqueda deseado del menú **MODOS DE BÚSQUEDA,** se lanzará el modo correspondiente sin necesidad de estar en la página Buscar.
- **Seleccionar el radio de búsqueda**: desplegando el control del menú **RADIO DE BÚSQUEDA**, podremos seleccionar la **distancia máxima de búsqueda desde un punto establecido desde el modo de búsqueda lugares o modo ubicación.**
- **Ordenar resultados por precio o por distancia:** permite que el listado en la página Resultados se muestre ordenado por precio o por distancia y se refleje como campo principal el criterio de ordenación

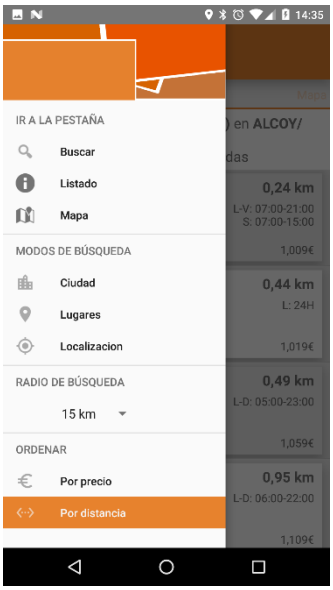


Selección del radio de búsqueda



Selección ordenar precio

Detalle con orden precio



Selección ordenar distancia

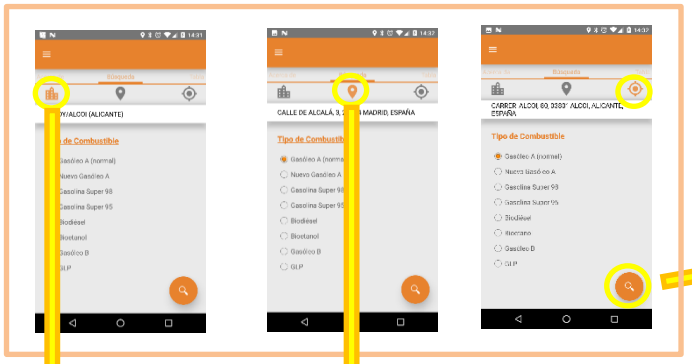
Detalle con orden distancia

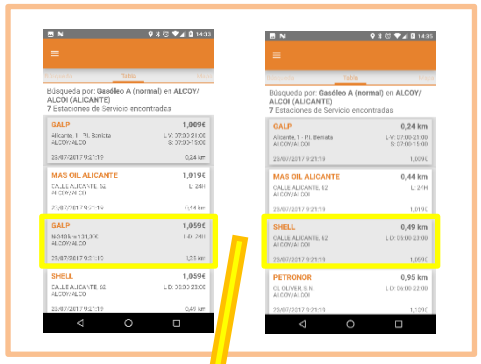
RESUMEN FLUJO DE LA APLICACIÓN

PÁGINA BUSCAR CON SELECCIÓN DE LOS MODOS DE BÚSQUEDA

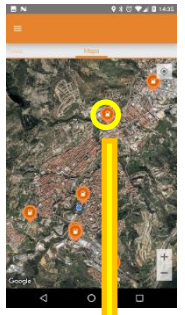
PÁGINA RESULTADOS

PÁGINA MAPA

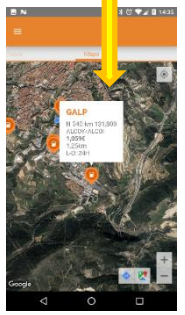


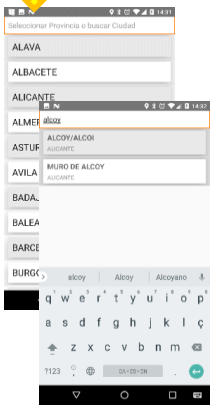


DESLIZANDO NOS MOVEMOS ENTRE LAS PESTAÑAS

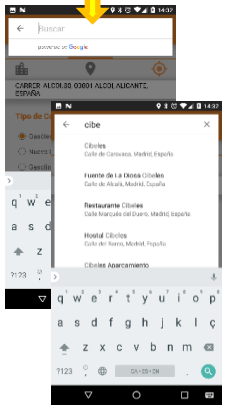


AL SELECCIONAR UN MARCADOR MUESTRA EL DETALLE DE LA ESTACIÓN DE SERVICIO



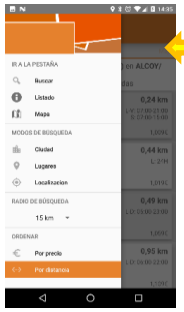


LLAMADA A ACTIVIDAD PARA BÚSCAR CIUDAD



LLAMADA PARA SELLECCIONAR EL LUGAR

PINCHANDO SOBRE EL DETALLE DE CUALQUIER ESTACIÓN DE SERVIIO ABRE GOOGLE MAPS



MOSTRAR NAVIGATION DRAWER DESDE CUALQUIER PÁGINA

5.11 - Características a destacar y puntos a mejorar

A modo de resumen se enumeran las características a destacar de la aplicación:

- **Búsqueda fácil y rápida**. Guarda los datos y configuraciones de la última búsqueda válida para buscar con un solo click.
- **Respuesta rápida,** se consultan los datos de forma que los archivos de respuesta son pequeños y solo nos devolverá los datos con los resultados deseados y no todo el listado. En los modos Lugares y Ubicación quedan limitados al radio de búsqueda establecido.
- La aplicación dispone una **interfaz totalmente personalizada y muy intuitiva**. Ha sido diseñada siguiendo las guías de **Material Design**:
    - Elementos con elevación
    - Floating Button
    - Swipe Refresh
    - Navigation Drawer
- **Solicitud de permisos en tiempo de ejecución**.
- El diseño de la aplicación **cubre el 99,3% de las versiones Android** desde la API 15 (4.0.3) a la 25 (7.1).

A continuación se destacan algunos puntos a mejorar que por falta de tiempo, o recursos no se han podido implementar:

- **Horarios estáticos**: los horarios mostrados en los resultados son texto y no se puede filtrar por las EESS abiertas, ni se indica directamente si la estación está abierta a cerrada.
- **No se muestran estadísticas de precios:** podría ser interesante reflejar un histórico de los precios en las EESS.
- **No se calculan búsquedas por rutas**.
- Se intentaron realizar **búsquedas por voz con el asistente de Google desde OK Google,** pero para ello la aplicación debe estar publicada en el Google Play y todavía no lo está.

6- Conclusiones

Todas las conclusiones que puedo obtener sobre el TFG son positivas.

En la parte técnica, la **aplicación cumple con las especificaciones marcadas** al iniciar el proyecto:

- Se considera que se han conseguido las premisas de realizar una aplicación en la que la **búsqueda** sea **rápida y fácil,** para que cualquier usuario de Android pueda hacer uso de ella.
- Dispone de **toda la funcionalidad pretendida**, incluso durante el desarrollo de la aplicación, se ha añadido la búsqueda por lugares que puede ser muy útil.
- Se **ajusta** en **diseño** a lo **esbozado en los trabajos previos.**
- La aplicación dispone una **interfaz actual y fluida** en su uso.
- El diseño de la aplicación **cubre el 99,3% de las versiones Android** desde la API 15 (4.0.3) a la 25 (7.1).
- Los **datos se obtienen de forma rápida**.

Por todo ello la **valoración** en la **parcela de realización personal es muy satisfactoria**.

Hay que reconocer que no todo ha sido un camino de rosas y **ha habido momentos difíciles**, en los que la solución a los problemas surgidos se demoraba más de lo esperado. Un ejemplo claro fue la necesidad de replantear toda la solución al detectar los problemas del servicio web del Ministerio. Si bien con **esfuerzo y constancia se han podido superar.**

Entendiendo el TFG como la culminación del Grado, puedo concluir que estos años de estudio, me han permitido:

- **Adquirir multitud de conocimientos técnicos**, muchos de los cuales he podido incorporar a mi puesto de trabajo.
- Marcarme nuevas metas profesionales, y confirmar o más bien reafirmar que si mi situación personal me lo permite, **mi futuro profesional seguirá ligado a la informática**.

Resumiendo, el Grado en Ingeniería Informática me ha ayudado a **seguir madurando profesional y personalmente**.

7- Referencias

- **El gran libro de Android**

  4ª Edición

  Editorial Marcombo

  ISBN: 978-84-267-2167-9

- **El gran libro de Android Avanzado.**

  3ª Edición.

  Editorial Marcombo

  ISBN: 978-84-267-2257-7

- **Servicio Web, LAMP, Servidor HTTP Apache, MySQL, PHP etc**

  https://es.wikipedia.org

- **Servicios Web del Ministerio**

  https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/help

  http://geoportalgasolineras.es/

- **Proveedor de máquinas virtuales en la nube**

  https://cloud.digitalocean.com/login

- **Ventajas de utilizar procedimientos almacenados**

  https://www.ibm.com/support/knowledgecenter/es/SSEPGG_8.2.0/com.ibm.db2.udb.dc.doc/dc/c_spbenefits.htm

- **PHP**

  http://www.php.net

- **Características y tabla comparativa de los sistemas operativos móviles más usados**

  http://jmacuna73.blogspot.com.es/2017/03/sistemas-operativos-moviles.html

- **Comparativa apps nativas, web o híbridas**
  http://www.raona.com/es/Solutions/Template/163/App-nativa-web-o-h%C3%ADbrida-

- **Cuota de mercado de Android, febrero 2017. Kantar Worldpanel**
  http://es.kantar.com/tech/m%C3%B3vil/2017/abril-2017-cuota-de-mercado-de-smartphones-en-espa%C3%B1a/

- **Arquitectura de Android. Android curso**
  http://www.androidcurso.com/index.php/99

- **Android developer dashboards julio 2017**
  https://developer.android.com/about/dashboards/index.html#Platform

- **Entorno de desarrollo Android Studio**
  https://developer.android.com/studio/intro/index.html

- **Developers google Autocompletado de sitios API de Google Places, Intenciones Google Maps etc**
  https://developers.google.com

- **Stack overflow. Solución de multitud de problemas**
  https://stackoverflow.com/