



Informe final de prácticas

Desarrollo de Sistemas de Software basados en Componentes y
Servicios

Juan Carlos Serrano Pérez

jcsp0003@correo.ugr.es

Pedro Manuel Gómez-Portillo López

gomezportillo@correo.ugr.es

11 de enero de 2019

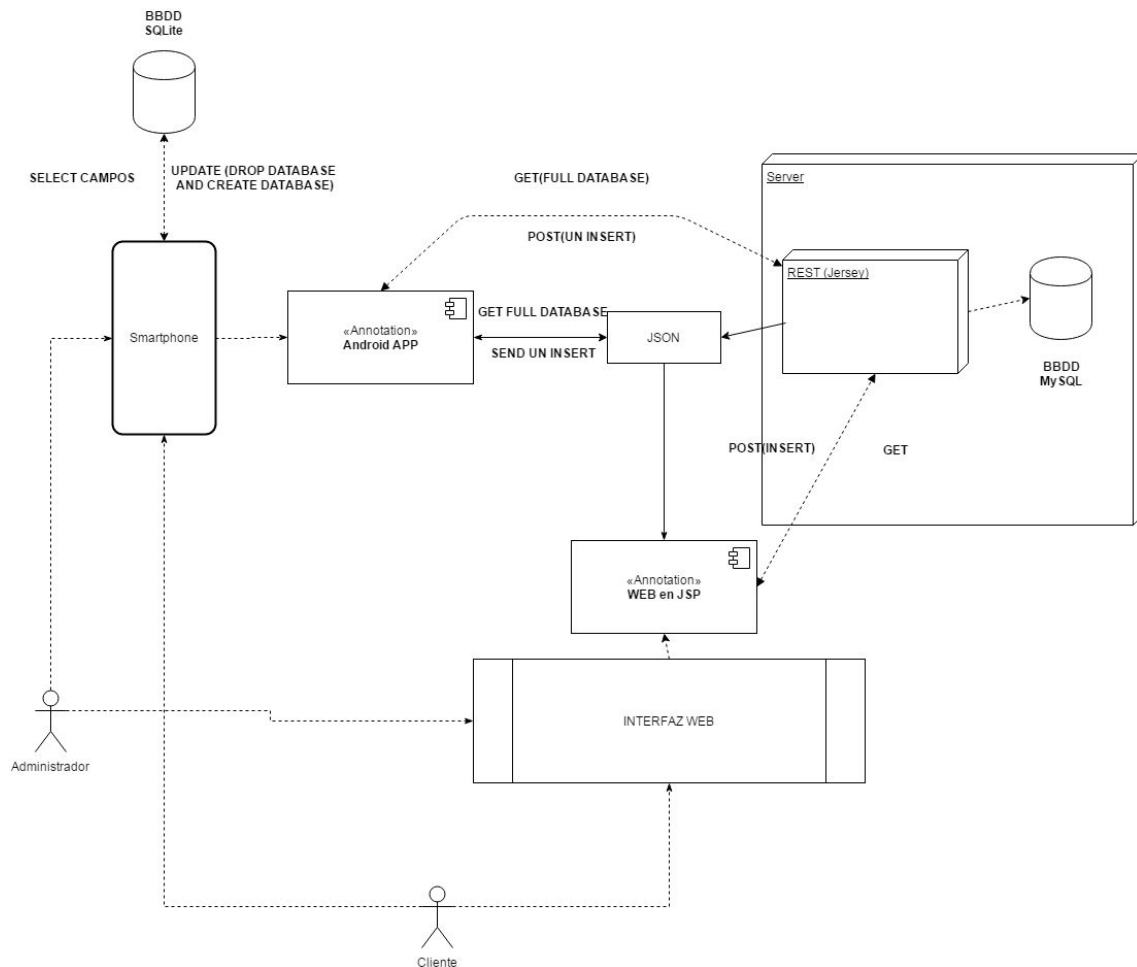
Índice

1. Introducción	3
2. Aplicación web	4
2.1. PaaS utilizados	4
2.1.1. Heroku	5
2.1.2. ClearDB	5
2.2. Diagrama de clases	6
2.3. Servidor	7
2.3.1. Uso de URIs	8
2.3.2. XML y JSON	9
2.3.3. Diagrama de Entidad-Relación de la base de datos	10
2.4. Sitio web	11
2.4.1. Página principal	11
2.4.2. Farmacias	12
2.4.3. Productos	13
2.4.4. Usuarios	14
2.4.5. Carrito	15
2.4.6. Pedidos	16
2.4.7. Librerías y frameworks utilizados	16
3. Aplicación móvil	17
3.1. Diagrama de clases	17
3.3. Diagrama de Entidad-Relación de la base de datos	18
3.2. Comunicación REST	19
3.3. Funcionamiento general	19
3.4. Uso de recursos	24
4. Bibliografía	25

1. Introducción

Este documento es la memoria de la práctica final del laboratorio de la asignatura Desarrollo de Sistemas de Software basados en Componentes y Servicios. A lo largo del mismo explicaremos las decisiones tomadas y los resultados obtenidos al terminar la práctica.

Para el desarrollo de este proyecto nos hemos basado en el diagrama de contexto proporcionado, que indica cómo deben comunicarse el servidor, la web y el móvil a través de una interfaz REST.



2. Aplicación web

En este capítulo se presenta la parte de la aplicación web. El repositorio en el que se ha trabajado para desarrollar el servidor puede verse en el siguiente enlace,

<https://github.com/gomezportillo/dss-pharmacy>.

Del mismo modo, el video demostración está subido en YouTube en la siguiente dirección,

<https://www.youtube.com/watch?v=34ATytwrbNA>

El lenguaje de programación utilizado ha sido **Python** con la librería **Flask** ya que es la más parecida a la librería Jersey de Java según StackOverflow¹.

Flask proporciona herramientas para trabajar a más alto nivel que Jersey, por lo que se ha evitado a toda costa usarlas. De hecho, el resultado final ha sido un proyecto de más bajo nivel que de haber utilizado Jersey, ya que tanto la serialización de los objetos a XML o JSON, la gestión total de la base de datos o el manejo de las rutas, entre otros, se hace de manera totalmente manual.

2.1. PaaS utilizados

Como el proyecto se ha realizado en Navidad, y los dos integrantes del equipo no podíamos quedar para trabajar juntos (yo vivo en Ciudad Real y mi compañero en Jaén), y con el fin de acercar esta práctica al desarrollo de un proyecto real, hemos optado por utilizar servicios PaaS² para desplegar el servidor.

Esto ha permitido que el servidor pueda ser fácilmente desplegado en la nube, por lo que la aplicación Android puede conectarse a él de manera cómoda sin necesidad de tenerlo ejecutado localmente para poder hacer pruebas.

Además, a la hora de corregir esta práctica creemos que es más cómodo hacerlo en un servidor desplegado y no tener que configurar un entorno local en el que hacerlo.

¹ <https://stackoverflow.com/questions/13483793/restful-python-for-java-jersey-developer>

² Platform as a Service

2.1.1. Heroku³

Heroku es un sitio web que ofrece un servicio PaaS tanto de manera gratuita como pagando en el que desplegar servidores en la nube para acceder a ellos cómodamente. En este caso, la ruta en la que se puede ver la parte del servidor y la página web es la siguiente,

<https://dss-pharmacy.herokuapp.com>.

Esta página web es perfectamente funcional, y se puede acceder a ella para ver el resultado final de la parte web.

El sitio web es discutido en la sección [1.4. Sitio web](#).

La carga de los datos desde la base de datos, de la que se hablará a continuación, es un poco lenta (del orden de 4 ó 5 segundos), por lo que es necesario tener un poco de paciencia.

En caso de que Heroku esté caído o simplemente se quiera ejecutar localmente, basta con descargar la práctica del repositorio, instalar las dependencias, ejecutar el servidor como se indica en el readme y acceder a la dirección *localhost:80*.

2.1.2. ClearDB⁴

Como optamos por trabajar con el servidor en la nube no podíamos desplegar la base de datos localmente, por lo que tuvimos que utilizar un sitio de *Database-as-a-Service* al que se conectara nuestro servidor desde Heroku y que soportara MySQL, y ClearDB es un sitio web que permite justamente esto.

El servicio de base de datos se encuentra alojado en la siguiente dirección,

us-cdbr-gcp-east-01.cleardb.net.

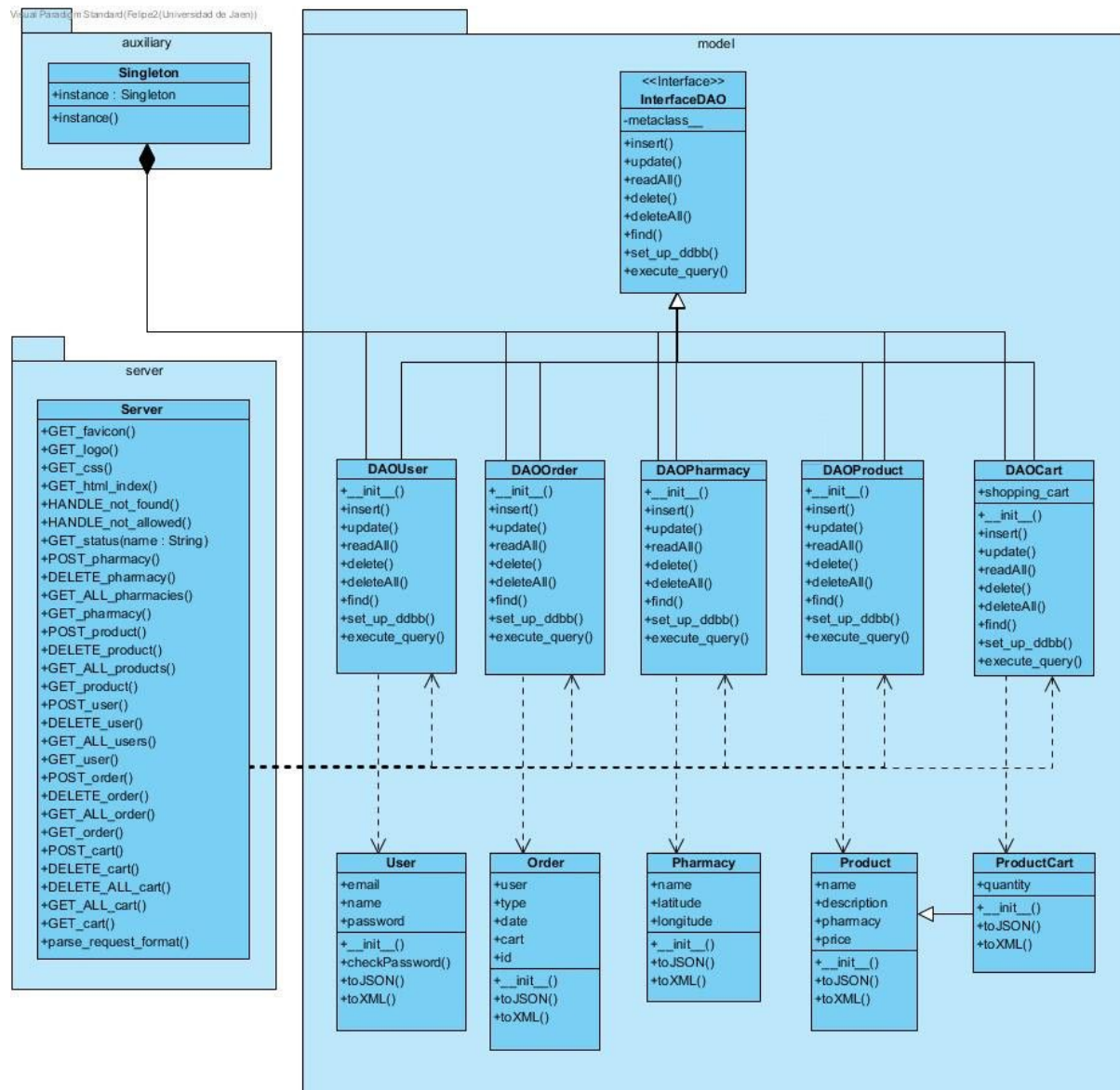
Este enlace se indica de manera prácticamente anecdótica, ya que es imposible trabajar con él sin el usuario y contraseña.

³ <https://heroku.com/>

⁴ <http://cleardb.com/>

2.2. Diagrama de clases

A continuación se presenta el diagrama de clases del servidor.



Para generar este diagrama se ha utilizado la herramienta de generación de diagrama clases desde código fuente de *Visual Paradigm*⁵.

El mismo diagrama con mayor resolución puede encontrarse [en el siguiente enlace](#).

⁵ <https://www.visual-paradigm.com/>

2.3. Servidor

Para diseñar el sistema de rutas se ha utilizado el mismo que en la práctica de servicios web.

\$BASE_URL/<servicio> devuelve la página web del servicio especificado, entre los que están

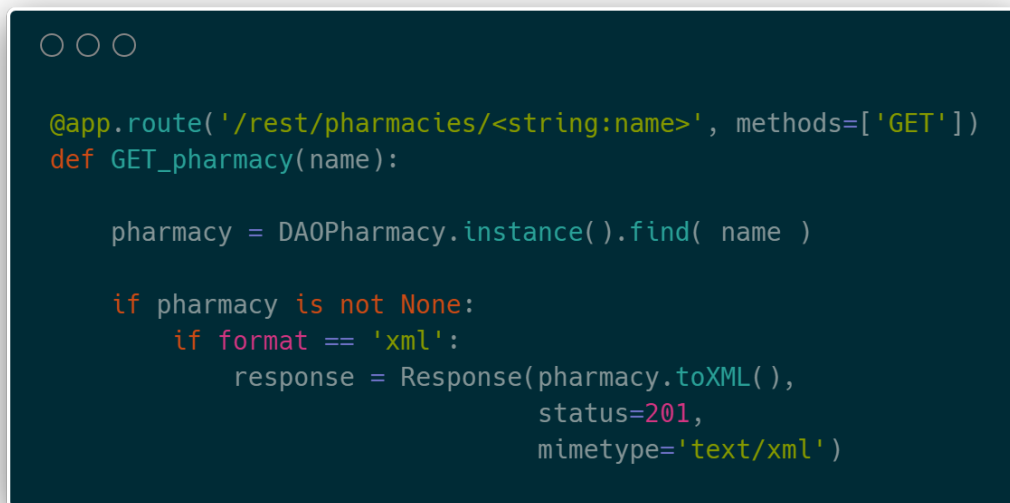
- *BASE_URL/pharmacies*
- *BASE_URL/products*
- *BASE_URL/users*
- *BASE_URL/orders*
- *BASE_URL/cart*

\$BASE_URL/rest/<servicio> es la interfaz a la API REST del servidor y acepta y responde peticiones HTTP usando URIs, lo que será explicado más adelante. Cada servicio tiene su ruta, y puede ser accedido y modificado a través de peticiones POST, PUT, GET y DELETE.

2.3.1. Uso de URIs

Para identificar los recursos del servidor se ha implementado un sistema de URIs⁶. A diferencia del uso de URLs⁷, donde los datos deben ser incluidos en la petición HTTP, el uso de URIs permite que cada elemento del servidor pueda ser accedido directamente desde un navegador web.

En la imagen de abajo⁸ se muestra un fragmento simplificado de la función que se encarga de manejar las peticiones HTTP GET sobre la ruta `/rest/pharmacies/*`.

A screenshot of a code editor with a dark blue background and light green text. The code defines a function `GET_pharmacy` that takes a `name` parameter. It uses `DAOPharmacy.instance().find(name)` to retrieve a pharmacy object. If the object is not `None` and the `format` is `'xml'`, it returns a `Response` object with `status=201` and `mimetype='text/xml'`.

```

@app.route('/rest/pharmacies/<string:name>', methods=['GET'])
def GET_pharmacy(name):

    pharmacy = DAOPharmacy.instance().find( name )

    if pharmacy is not None:
        if format == 'xml':
            response = Response(pharmacy.toXML(),
                                status=201,
                                mimetype='text/xml')
```

Como vemos, cuando llegar una petición GET a la ruta `/rest/pharmacies/<nombre_farmacia>`, se obtiene la instancia del singleton DAO⁹ de la clase `Farmacia`, al que se le pide que encuentre la farmacia, y si el formato en el que se ha pedido es XML, se serializa y se manda configurando a mano la cabecera de la respuesta.

Por ejemplo, para acceder a todos los usuarios basta acceder a la URI <https://dss-pharmacy.herokuapp.com/rest/users>, y si solo queremos al administrador lo podemos encontrar en <https://dss-pharmacy.herokuapp.com/rest/users/admin>.

⁶ *Uniform Resource Identifier*

⁷ *Uniform Resource Locator*

⁸ Imagen generada con la herramienta web <https://carbon.now.sh>

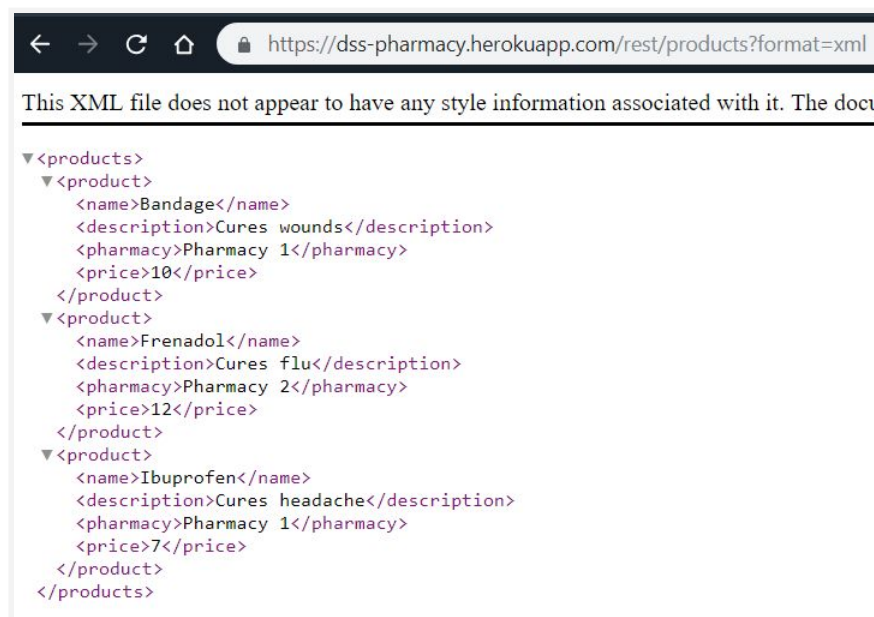
⁹ Data Access Object

2.3.2. XML y JSON

Todos los datos del servidor pueden ser obtenidos en formato JSON o XML. Para elegir el formato basta con incluir el parámetro `?format=xml` o `?format=json` al final de la URL.

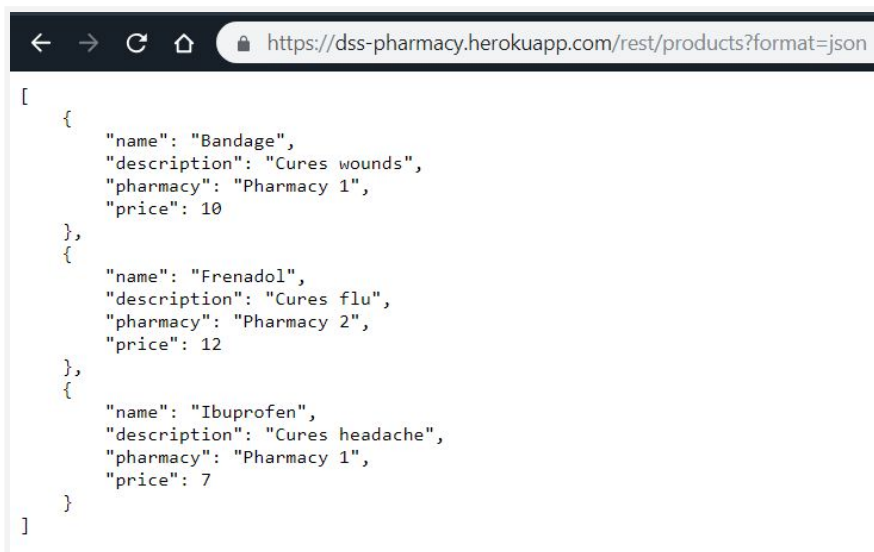
Por defecto se ha usado JSON, ya que tanto las librerías utilizadas tanto para las páginas web como las de Android trabajan con este formato, por lo que de no especificar ninguno los mensajes serán devueltos en JSON.

Las imágenes inferiores muestran ésto, primero en XML y luego en JSON.



The screenshot shows a web browser with the address bar displaying `https://dss-pharmacy.herokuapp.com/rest/products?format=xml`. Below the address bar, a message states: "This XML file does not appear to have any style information associated with it. The document". The main content area displays the following XML structure:

```
<products>
  <product>
    <name>Bandage</name>
    <description>Cures wounds</description>
    <pharmacy>Pharmacy 1</pharmacy>
    <price>10</price>
  </product>
  <product>
    <name>Frenadol</name>
    <description>Cures flu</description>
    <pharmacy>Pharmacy 2</pharmacy>
    <price>12</price>
  </product>
  <product>
    <name>Ibuprofen</name>
    <description>Cures headache</description>
    <pharmacy>Pharmacy 1</pharmacy>
    <price>7</price>
  </product>
</products>
```



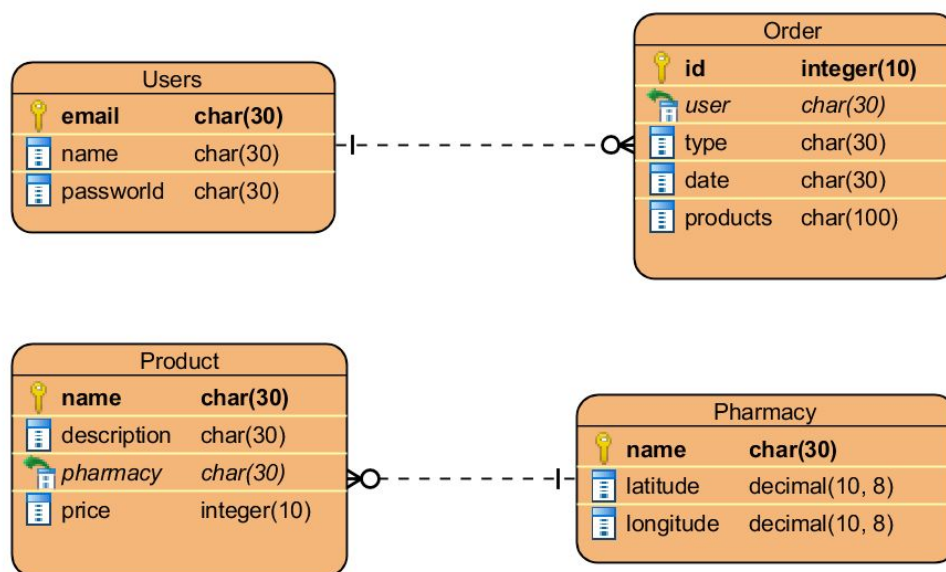
The screenshot shows a web browser with the address bar displaying `https://dss-pharmacy.herokuapp.com/rest/products?format=json`. The main content area displays the following JSON structure:

```
[
  {
    "name": "Bandage",
    "description": "Cures wounds",
    "pharmacy": "Pharmacy 1",
    "price": 10
  },
  {
    "name": "Frenadol",
    "description": "Cures flu",
    "pharmacy": "Pharmacy 2",
    "price": 12
  },
  {
    "name": "Ibuprofen",
    "description": "Cures headache",
    "pharmacy": "Pharmacy 1",
    "price": 7
  }
]
```

El parseado de la URL y el serializado de los objetos en estos dos formatos no se delega en ninguna librería ni función de Flask, sino que es hecho de manera manual para evitar utilizar herramientas de alto nivel.

2.3.3. Diagrama de Entidad-Relación de la base de datos

La persistencia del servidor se basa en una base de datos MySQL almacenada en ClearDB. A continuación se presenta el diagrama Entidad-Relación con las tablas creadas para este proyecto generado con la herramienta *Visual Paradigm*.



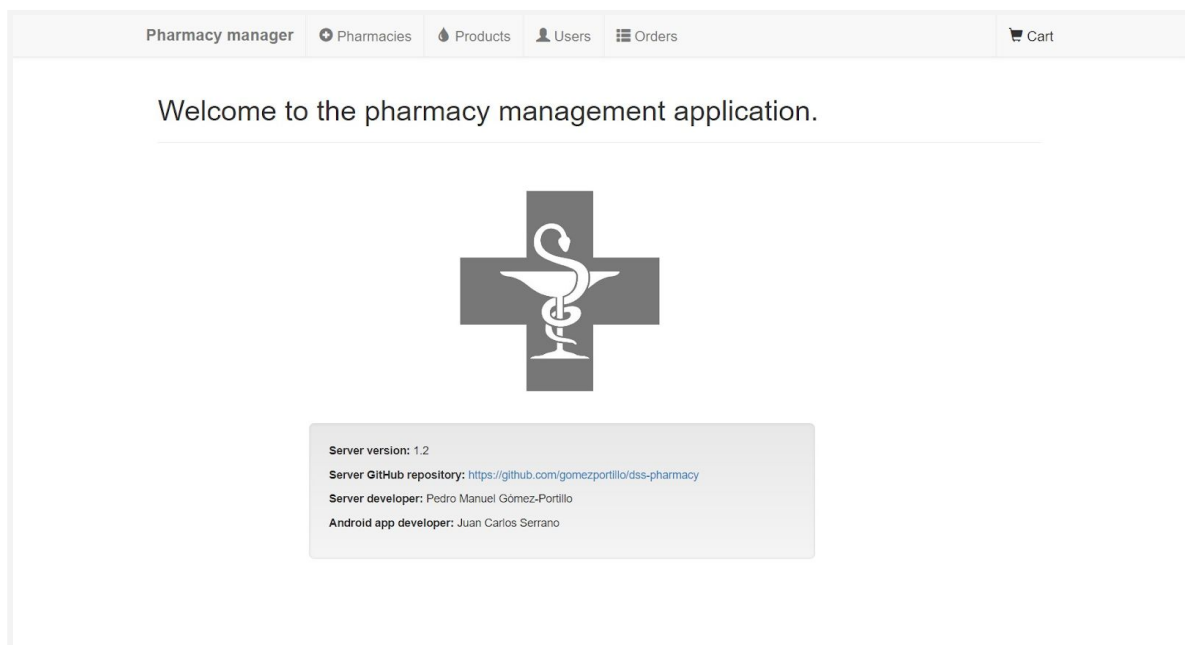
2.4. Sitio web

En esta sección se presenta el sitio web y sus diferentes secciones. Antes de empezar, cabe decir que se ha supuesto que el único usuario de la web es el administrador, y los únicos usuarios de la aplicación móvil son los clientes, por lo que el administrador tiene poder pleno en la web para añadir, editar y borrar los elementos que crea conveniente.

En cada página que permite la creación de elementos por parte del administrador se han escrito valores de prueba con valores coherentes, lo que permite probarlos rápidamente. Por ejemplo, en la página de creación de farmacias se han llenado los campos con una latitud y una longitud que permite crear una farmacia a la vista del mapa, para comprobar que toda la funcionalidad es ejecutada correctamente.

2.4.1. Página principal

Es la página que se muestra al acceder al sitio web. En ella se puede consultar la versión del servidor, así como su repositorio y sus desarrolladores. Esta información es obtenida a través de una petición GET al servidor.

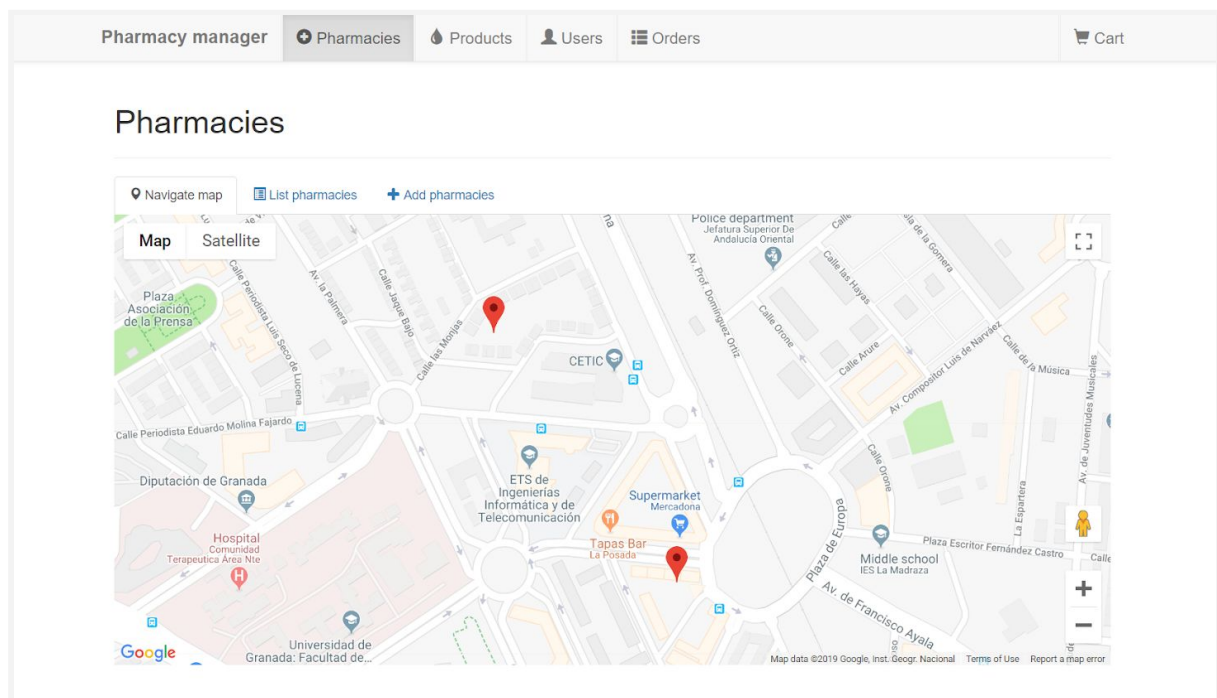


2.4.2. Farmacias

En esta página se muestra la información de las farmacias almacenadas en el servidor. En la pestaña *Naviga map* se ha incluido usando la API de Google Maps un mapa con el nombre y la localización exacta de las farmacias obtenidas a través de una petición GET en el servidor.

En la pestaña *List pharmacies* puede verse la misma información en forma de lista, así como borrar farmacias a través de una petición DELETE al servidor.

Por último, en la pestaña *Add pharmacies* se pueden añadir o editar farmacias incluyendo su nombre, latitud y longitud, que son enviadas al servidor a través de un peticiones PUT y POST.



2.4.3. Productos

En la página *Products* pueden verse y editarse los productos del servidor. En la primera pestaña, *See products*, aparecen listados todos los productos del sistema, que el administrador puede añadir al carro, lo que genera una petición PUT en el carro, o borrarlo, lo que genera una petición DELETE.

En la segunda pestaña, *Add products*, el administrador puede crear nuevos productos o editar los que ya existen, generando una petición POST en el servidor.

Pharmacy manager	Pharmacies	Products	Users	Orders	Cart
Products					
<div>See products</div> <div>+ Add products</div>					
Name	Description	Price (€)	Pharmacy	Add to cart	Delete
Bandage	Cures wounds	10	Pharmacy 1	<div>+</div>	<div></div>
Frenadol	Cures flu	12	Pharmacy 2	<div>+</div>	<div></div>
Ibuprofen	Cures headache	7	Pharmacy 1	<div>+</div>	<div></div>

Para evitar que el administrador seleccione una farmacia inválida al crear un producto necesita seleccionarla de una lista desplegable, que es llenada con la lista de farmacias obtenidas del servidor a través de una petición GET.

Pharmacy where is sold






Pharmacy 1

Pharmacy 1


Pharmacy 2




2.4.4. Usuarios

En la página *Users* el administrador puede ver los usuarios que existen y borrarlos, y en la pestaña *Add users* puede crearlos o editarlos, todo a través de peticiones HTTP.

Pharmacy manager  Pharmacies  Products ** Users**  Orders  Cart

Users

 See users [+ Add user](#)

Email	Name	Password	Delete
admin	Administrator	admin	
gomezportillo@dss.com	Pedro Manuel Gomez-Portillo	1234	
juan.carlos.wow.95@gmail.com	Juan Carlos Serrano	secretpassworkd	

2.4.5. Carrito

Cada vez que un producto es añadido al carro aparece listado aquí, estos aparecerán en la página *Cart*, y si es añadido cuando ya existe se incrementará en 1 su cantidad. Inmediatamente debajo de los productos aparece su precio total.

Tras esto se encuentran las opciones de *Vaciar el carrito*, *Reservar los productos* o *Comprarlos*. Al pulsar en vaciar el carrito se enviará una petición DELETE al servidor que lo vaciará. Por otro lado, si pulsamos en Reservar o Comprar se comprobará en el servidor que el email introducido sea correcto y que el carrito no esté vacío, tras lo cual se generará la orden de reserva o compra.

Para esta página se ha supuesto que el administrador puede hacer pedidos en nombre de cualquier otro usuario introduciendo su email.

Pharmacy manager

Pharmacies

Products

Users

Orders

Cart

Cart

Name	Description	Price (€)	Pharmacy	Quantity
Frenadol	Cures flu	12	Pharmacy 2	1
Bandage	Cures wounds	10	Pharmacy 1	1

Total price: 22€

Email:

Enter a registered email for reserving or purchasing your cart

Empty cart

Reserve products

Buy products

2.4.6. Pedidos

Por último, en la página *Orders* pueden verse los pedidos que han sido efectuados. Entre los datos de los pedidos se incluye el ID, que es la clave primaria y generado automáticamente por la base de datos, el cliente, el tipo de pedido, la fecha en la que ha sido generado y la lista de productos que incluye.

La lista de pedidos se obtiene a través de una petición GET al servidor, y como se supone que son pedidos firmes no se da la opción de editarlos ni borrarlos.

Pharmacy manager					
Pharmacies		Products	Users	Orders	Cart
Orders					
ID	Client	Type	Date	List of products	
1	juan.carlos.wow@gmail.com	Reserve	1-1-2019 11:24:44	• Ibuprofen. Cures headache. Pharmacy 1. 7EUR x 2u	
11	gomezportillo@dss.com	Purchase	2019-01-06 20:22:07	• Frenadol. Cures flu. Pharmacy 2. 12EUR x 1u • Bandage. Cures wounds. Pharmacy 1. 10EUR x 1u	

2.4.7. Librerías y frameworks utilizados

Para el desarrollo de las páginas web se ha utilizado el framework *Bootstrap*¹⁰ para la parte de diseño y la librería JavaScript *jQuery*¹¹ para la parte funcional. Toda la información que se intercambia entre el servidor y las páginas web se hace en formato JSON y a través de su API REST.

¹⁰ <https://getbootstrap.com/>

¹¹ <https://jquery.com/>

3. Aplicación móvil

En este capítulo se presenta la parte de la aplicación Android. El repositorio en el que se ha trabajado para desarrollar la aplicación puede verse en el siguiente enlace,

https://github.com/xenahort/Aplicacion_Android_maps_receptiva_y_adaptable

y se puede encontrar un vídeo en el que se hace uso de la aplicación en el siguiente enlace,

<https://www.youtube.com/watch?v=yHRL6B2Xp1Q&feature=youtu.be>

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma.

El sistema permite programar aplicaciones en una variación de Java. El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

3.1. Diagrama de clases

A continuación se presenta el diagrama de clases del cliente Android.

3.2. Comunicación REST

Para la comunicación REST con el servidor se ha utilizado la librería Retrofit. Se trata de un cliente HTTP de tipo seguro para Android y Java, que hace sencillo conectar a un servicio web REST traduciendo la API a interfaces Java.

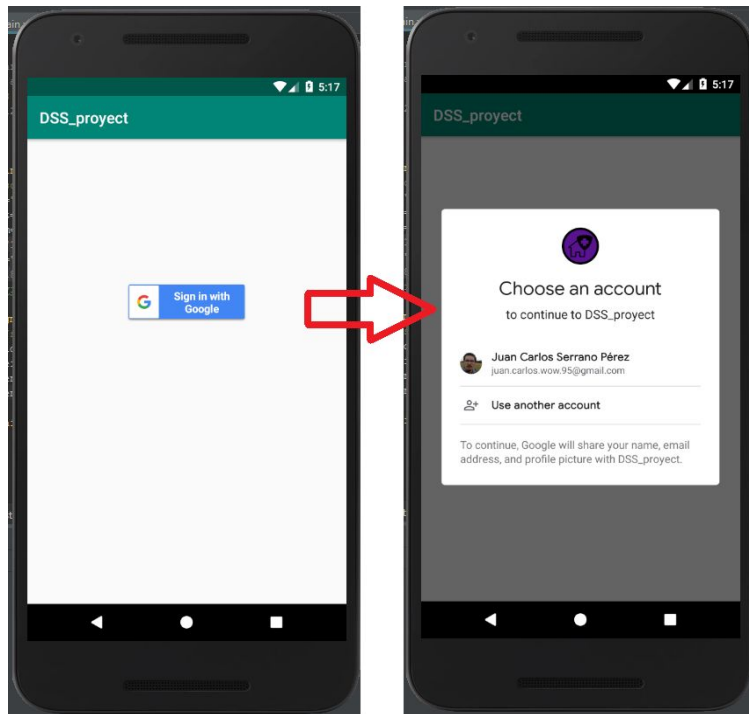
Con Retrofit podemos consumir datos JSON o XML, que después son transformados en Objetos Java. Todas las peticiones GET, POST, PUT, PATCH, y DELETE pueden ser ejecutadas.

Se ha implementado el paquete Comunicacion dentro del cual se encuentra la interfaz GetPostService que está compuesta por las 3 funciones de comunicación REST de las que hace uso la aplicación móvil:

- **Call<List<Farmacia>> getAllPharm()** que hará una petición GET al servidor a la ruta /rest/pharmacies y la aplicación recibirá una lista de las farmacias más cercanas.
- **Call<List<Producto>> getAllProduct()** que hará una petición GET al servidor a la ruta /rest/products y la aplicación recibirá una lista de los distintos productos disponibles en cada farmacia así como sus características.
- **Call<Respuesta> crearPedido(@Field("email"), @Field("type"), @Field("date"), @Field("cart"))** que hará una petición POST al servidor a la ruta /rest/orders para realizar una reserva. La aplicación recibirá un objeto de tipo Respuesta con el resultado de la operación.

3.3. Funcionamiento general

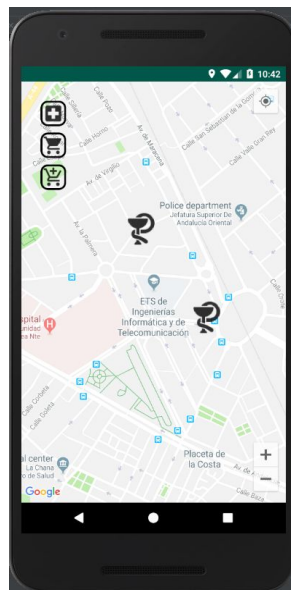
Inicialmente la app mostrará una actividad para iniciar sesión a través de una cuenta de Google. Una vez seleccionada la cuenta que se desee se pasará a la actividad central de la aplicación.



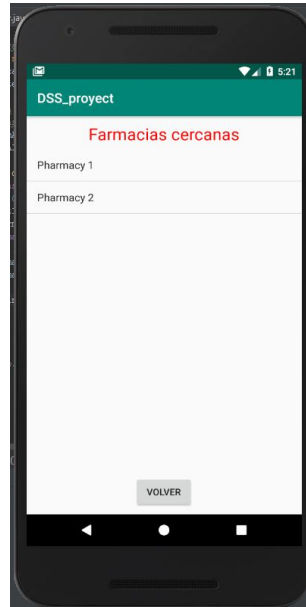
En esa actividad se nos marcará en el mapa las distintas farmacias y podremos ver información específica como su dirección, teléfono, página web u horario de apertura, también podemos solicitar que nos guíe hasta allí.

En esta actividad, en la zona superior izquierda encontramos tres botones:

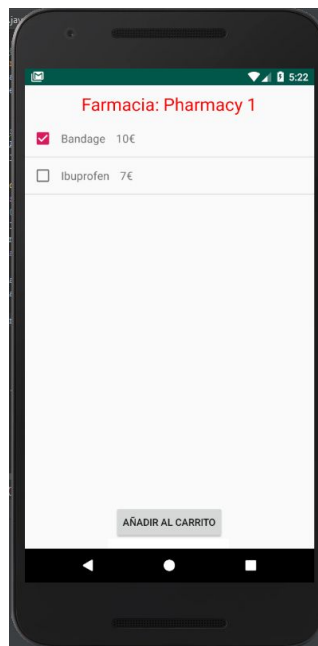
- Superior: nos llevará a la lista de farmacias cercanas.
- Central: nos llevará al carrito donde poder ver los productos ya seleccionados.
- Inferior: nos llevará al historial de reservas realizadas.



Como se ha mencionado si seleccionamos el botón superior iremos a una actividad similar a la siguiente en la que aparecerán las distintas farmacias por orden de cercanía y podremos elegir la que se desee.

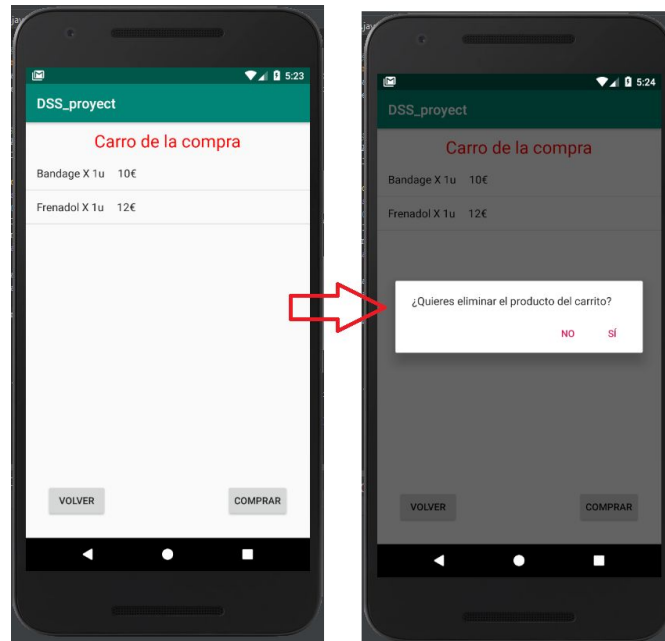


Una vez seleccionada la farmacia pasaremos a una lista de multiselección del inventario disponible en la farmacia seleccionada. Una vez seleccionados los productos los añadiremos al carrito y volveremos a la actividad central del mapa.



En la actividad central del mapa, si pulsamos el botón central iremos a la actividad encargada de mostrarnos el contenido del carrito. En ella podemos ver una lista de los elementos que hemos añadido, las unidades y el precio total.

En caso de no desear un producto lo podemos seleccionar y se nos mostrará un diálogo preguntándonos si queremos eliminarlo del carrito.

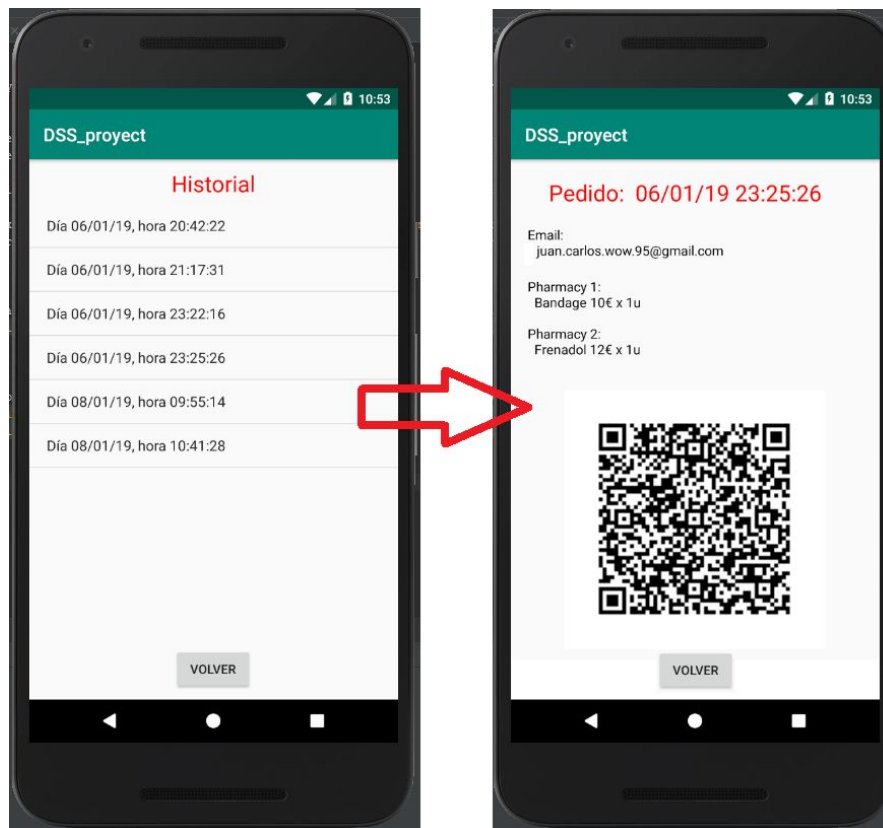


Una vez pulsamos el botón comprar se procederá a hacer la reserva de los productos e iremos a una nueva actividad en la que se nos mostrará un código Qr como el siguiente a modo de recibo que contiene elementos como nuestro email, tipo de transacción, fecha en la que se ha realizado y el contenido del carrito.



```
"email": "juan.carlos.wow.95@gmail.com",  
"type": "Purchase",  
"date": "Sun Jan 06 17:26:20 2019",  
"products": "Bandage. Cures wounds. Pharmacy 1. 10EUR x 1u; Frenadol.  
Cures flu. Pharmacy 2. 12EUR x 1u;"
```

Nuevamente en la actividad central de la aplicación, si pulsamos en este caso el botón inferior iremos a al historial de las reservas realizadas y podremos ver los detalles de la reserva.



3.4. Uso de recursos

Haciendo uso del emulador de dispositivos virtuales que trae incorporado Android Studio se ha comparado el gasto medio de recursos que hace la aplicación en cada una de las actividades en un dispositivo Android Nexus 5X.

	CPU	Memoria	Red	Energía
MainActivity	1%	40 MB	0.3 KB/s	Light
MapsActivity	30%	70 MB	5 KB/s	Medium
ListaFarmaciasActivity	2%	50 MB	0 KB/s	Ligh
ListaProductosActivity	3%	61 MB	0.4 KB/s	Ligh
ListaProductosCarritoActivity	3%	55 MB	0 KB/s	Ligh
HacerReservaActivity	6%	72 MB	0.6 KB/s	Medium
HistorialActivity	3%	55 MB	0 KB/s	Ligh
DetallesReservaActivity	7%	69 MB	0 KB/s	Medium

A raíz de los resultados vemos cómo el gasto de recursos de la aplicación es muy bajo y únicamente destacan tres actividades:

- **MapsActivity**: Se trata de la actividad que consume más recursos debido a que debe no solo realizar una petición GET al servidor para solicitar las farmacias, sino que principalmente debe hacer uso de la API de Google para cargar el mapa de la zona.
- **HacerReservaActivity y DetallesReservaActivity**: cabe destacar que estas actividades hacen un gasto de recursos ligeramente por encima de la media ya que nuevamente hace no solo una petición POST al servidor (solo la primera) sino que genera una imagen con el código Qr que servirá al usuario como recibo de la reserva.

Finalmente una característica importante es el tamaño final de la aplicación, ya que los dispositivos móviles disponen de un tamaño de almacenamiento muy limitado. La apk generada, ocupa solamente **3.2MB** lo que consideramos que es un tamaño más que aceptable ya que hemos minimizado la resolución de las imágenes así como otros elementos para que usuario pueda disfrutarlas independientemente de dispositivo.

4. Bibliografía

[1] "Enviando Datos Con el Cliente HTTP Retrofit 2 para Android", Code Envato Tuts+, 2019. [Online]. Available: <https://code.tutsplus.com/es/tutorials/sending-data-with-retrofit-2-http-client-for-android--cms-27845>. [Accessed: 06- Jan- 2019].

[2] J. Pastor, "Nexus 5X, análisis: el precio condena al digno heredero del Nexus 5", Xataka.com, 2019. [Online]. Available: <https://www.xataka.com/analisis/nexus-5x-analisis-el-precio-condena-al-digno-heredero-del-nexus-5>. [Accessed: 06- Jan- 2019].

[3] A. Gonzalez, "¿Qué es Android?", Xatakandroid.com, 2019. [Online]. Available: <https://www.xatakandroid.com/sistema-operativo/que-es-android>. [Accessed: 07- Jan- 2019].

[4] "Flask (A Python Microframework)", Flask.pocoo.org, 2019. [Online]. Available: <http://flask.pocoo.org/>. [Accessed: 08- Jan- 2019]

[5] M. Pilgrim, Dive into Python 3. Apress, 2009.