

2020

Aplicación WEB con Flask

AUTOR: FRANCISCO JAVIER HERNÁNDEZ

CORREO DE CONTACTO: fj.herez@gmail.com

Índice

Descripción del proyecto	2
Tecnología empleada	2
Guía de instalación	3
Instalar mariadb en windows.....	3
Instalar mariadb en Linux.....	5
Instalar Python en windows	5
Instalar Python en Linux (Debian 9).....	7
Descargar y configurar los archivos del repositorio en Windows.....	7
Descargar y configurar los archivos del repositorio en Linux.....	11
Guía de Usuario.....	14
Diagrama Entidad-Relación y Modelo Relacional	20
Conclusión	21

Descripción del proyecto

Este proyecto tiene el objetivo de desarrollar una aplicación web que se parezca en funcionamiento a una tienda online. El funcionamiento interno de la tienda debe tener un sistema CRUD para consultar, actualizar, eliminar y añadir datos, tanto de productos como de clientes y mostrarlos en el navegador a través de vistas. Se deberá emplear scripts de JQuery para el control del funcionamiento del front-end como por ejemplo, para la validación de un formulario. A través de Ajax se deberá enviar inicialmente, los datos a la base de datos mediante un fichero JSON. La página deberá ser Responsive, es decir, deberá tener diseño adaptativo.

Tecnología empleada

Para la resolución de los objetivos planteados se utilizará las siguientes tecnologías.

En primer lugar se ha seleccionado como lenguaje servidor, python. Se ha decidido usar este lenguaje debido a que tiene algunas ventajas sobre PHP como por ejemplo, el código queda mucho más limpio y legible debido a que este lenguaje al carecer de “{” para definir bucles, IF o funciones, utiliza el sistema de indentación. Python necesita menos requerimientos para poner un servidor WEB en funcionamiento, ya que solo necesitas tener python instalado como tal, también tiene su propio gestor de paquetes internos “pip”, no tiene tanta documentación como PHP pero si tiene una comunidad muy amplia capaz de resolver multitud de problemas que pueden surgir durante el desarrollo de la APP. Uno de los puntos más importantes para utilizar python sobre PHP, es debido a que es un lenguaje multiparadigma y le permite ser compatibles con otras áreas, como la de inteligencia artificial. El framework de python que se utilizará será Flask ya que utiliza el motor de plantillas jinja2, es bastante sencillo desarrollar apps con él, flask tiene una documentación muy amplia y tiene un mejor rendimiento que Django debido a que flask es más simple.

La base de datos que se ha empleado es MariaDB debido principalmente a que es un proyecto distribuido con licencia GPL, lo que quiere decir que es gratuito y de código abierto. Otra de las causas es el cambio de motor que usa mariaDB, mysql utiliza MyISAM y mariaDB utiliza aria. La ventaja que tiene aria frente a MyISAM es que aria guarda su caché en RAM lo que mejora sutilmente su rendimiento y no en el disco como MyISAM. MariaDB permite de una manera más sencilla la optimización de una base de datos, ya que almacena estadísticas del servicio. En general debido a que mariaDB es una bifurcación de mysql no presenta casi ninguna desventaja frente a mysql.

Se ha utilizado para el desarrollo front-end, JQuery, Ajax, Jinja2, Bootstrap, CSS y HTML. La razón para utilizar JQuery en este caso es por la sencillez y las líneas de códigos que te vas a ahorrar al interactuar con la página WEB, la principal desventaja de JQuery que es el rendimiento al manipular el DOM no es un problema en esta situación ya que la APP no se espera que tenga que realizar muchas operaciones por segundo. Se ha utilizado Ajax debido a que es una de las técnicas más extendidas para la creación de web interactivas, se comunica con el servidor de forma asíncrona, de esta forma es posible realizar cambios en la página sin necesidad de recargarla. Jinja2 es el motor de plantillas de flask y facilita el control del DOM de la página y la vista de los

datos que el servidor obtiene de consultas o procesamiento de datos en el navegador. Se ha utilizado Bootstrap para cumplir con los objetivos de una página responsive y también se ha empleado CSS puro para adaptar el bootstrap a las necesidades que se tenía en cada caso. Por último se ha utilizado HTML5 para el desarrollo de la estructura de las páginas.

Guía de instalación

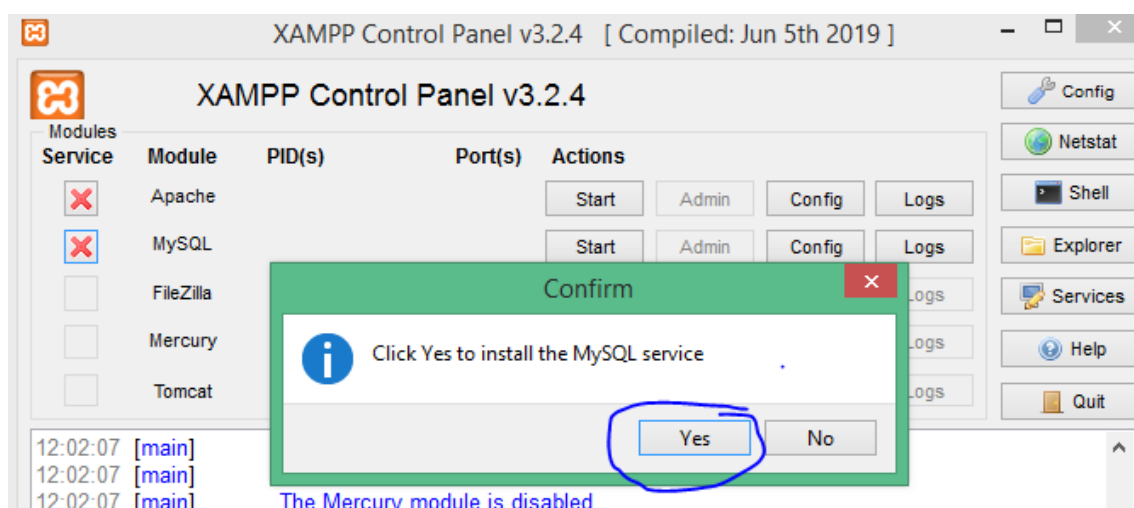
Primero se debe preparar el entorno, dependerá de si estas utilizando sistemas operativos Windows o Linux. Se comienza con la instalación con el sistema de gestión de la base datos que se empleará, mariaDB.

Instalar mariadb en windows

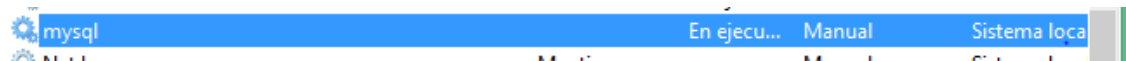
En Windows se procede a instalar XAMPP, una vez instalado si se prefiere controlar como un servicio debemos ejecutar el panel de control de XAMPP como administrador y hay que fijarse en el lateral



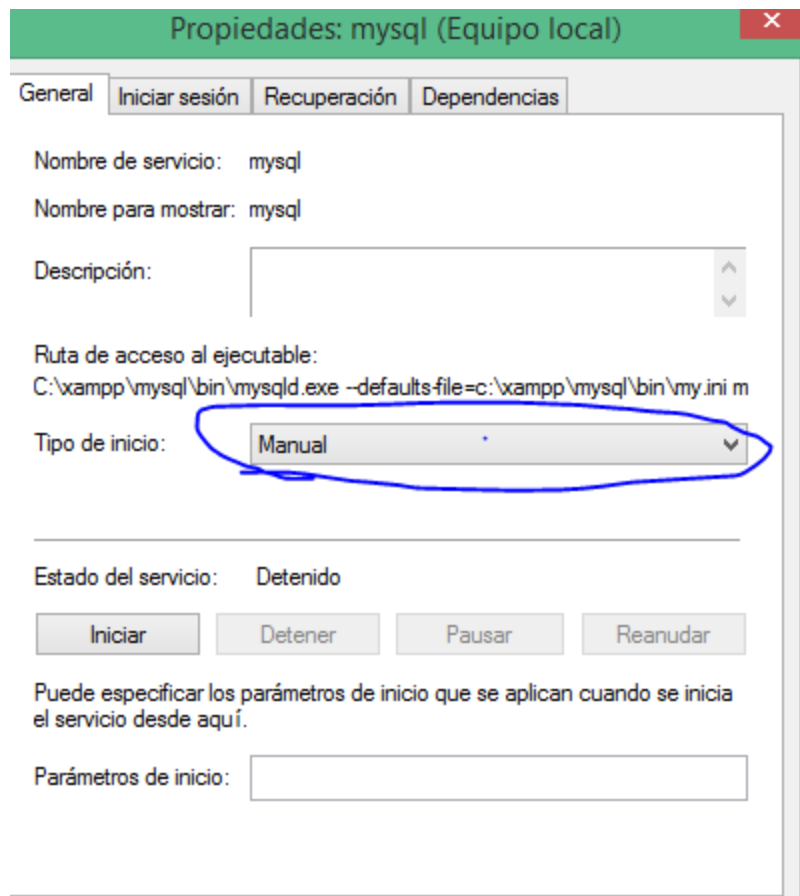
Debemos hacer click en el servicio que queremos instalar, en este caso mysql



A continuación comprobamos que se ha instalado el servicio en servicios locales



Hacemos click izquierdo sobre el servicio para evitar que se ejecute con el inicio del sistema



Una vez realizado estos pasos ejecutamos una consola CMD como administrador y comprobamos que podemos ejecutar mysql.exe

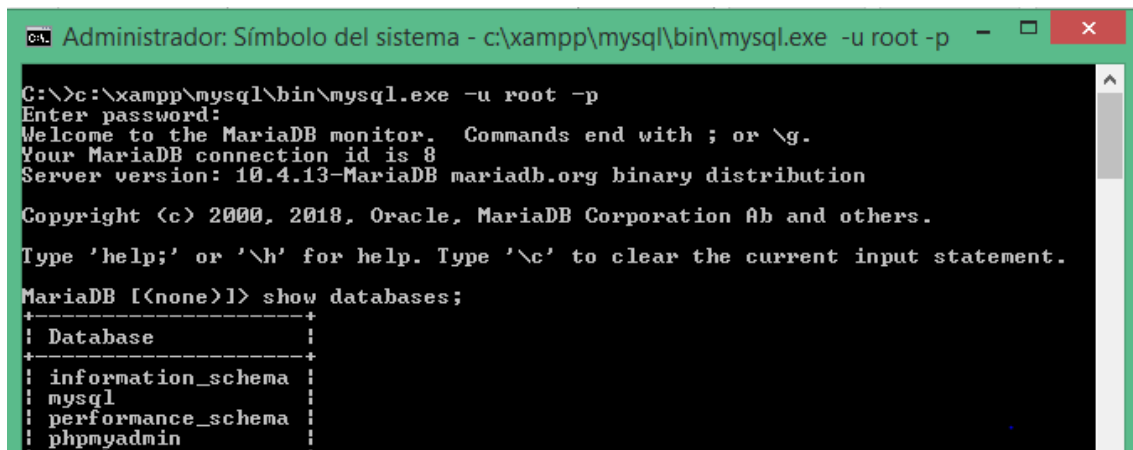
```
Administrator: Símbolo del sistema
C:\>sc query mysql
NOMBRE_SERVICIO: mysql
TIPO           : 10  WIN32_OWN_PROCESS
ESTADO         : 1  STOPPED
CÓDIGO_DE_SALIDA_DE_WIN32 : 1077  <0x435>
CÓDIGO_DE_SALIDA_DEL_SERVICIO: 0  <0x0>
PUNTO_DE_CONTROL : 0x0
ESPERA         : 0x0

C:\>sc start mysql
NOMBRE_SERVICIO: mysql
TIPO           : 10  WIN32_OWN_PROCESS
ESTADO         : 2  START_PENDING
                : (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
CÓDIGO_DE_SALIDA_DE_WIN32 : 0  <0x0>
CÓDIGO_DE_SALIDA_DEL_SERVICIO: 0  <0x0>
PUNTO_DE_CONTROL : 0x3
INDICACIÓN_INICIO : 0x3a98
PID            : 4788
MARCAS         :

C:\>sc query mysql
NOMBRE_SERVICIO: mysql
TIPO           : 10  WIN32_OWN_PROCESS
ESTADO         : 4  RUNNING
                : (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
CÓDIGO_DE_SALIDA_DE_WIN32 : 0  <0x0>
CÓDIGO_DE_SALIDA_DEL_SERVICIO: 0  <0x0>
PUNTO_DE_CONTROL : 0x0
INDICACIÓN_INICIO : 0x0

C:\>
```

Para ejecutar mariaDB desde CMD se debe introducir el siguiente comando



```

C:\>c:\xampp\mysql\bin\mysql.exe -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.13-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
+-----+

```

Instalar mariadb en Linux

En Linux para instalar mariadb se debe introducir el comando:

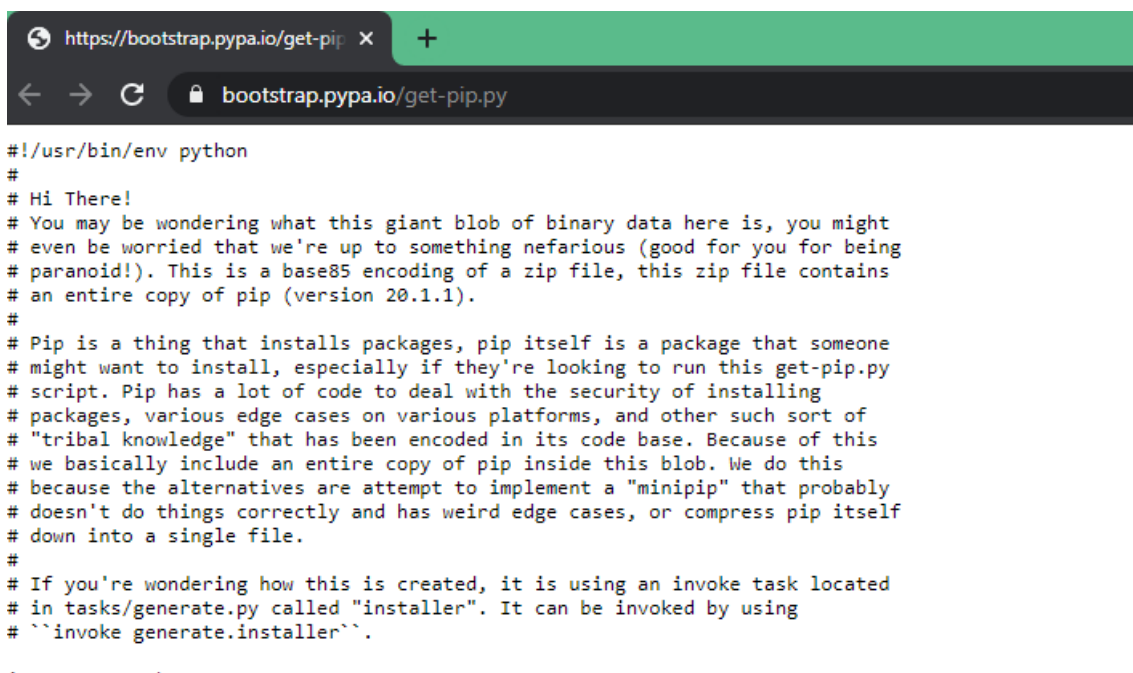
- apt install mariadb-server

Instalar Python en windows

Una vez que ya se ha instalado y configurado el servicio de mysql, se procede con la instalación de python desde la URL: "<https://www.python.org/downloads/>". La versión utilizada en el proyecto es la versión 3.7.8.

Una vez instalado con las preferencias que desee cada usuario. Se debe comprobar que pip está instalado, en caso de que el gestor de paquetes no se instale se debe hacer lo siguiente:

Primero visitar esta URL: <https://bootstrap.pypa.io/get-pip.py>

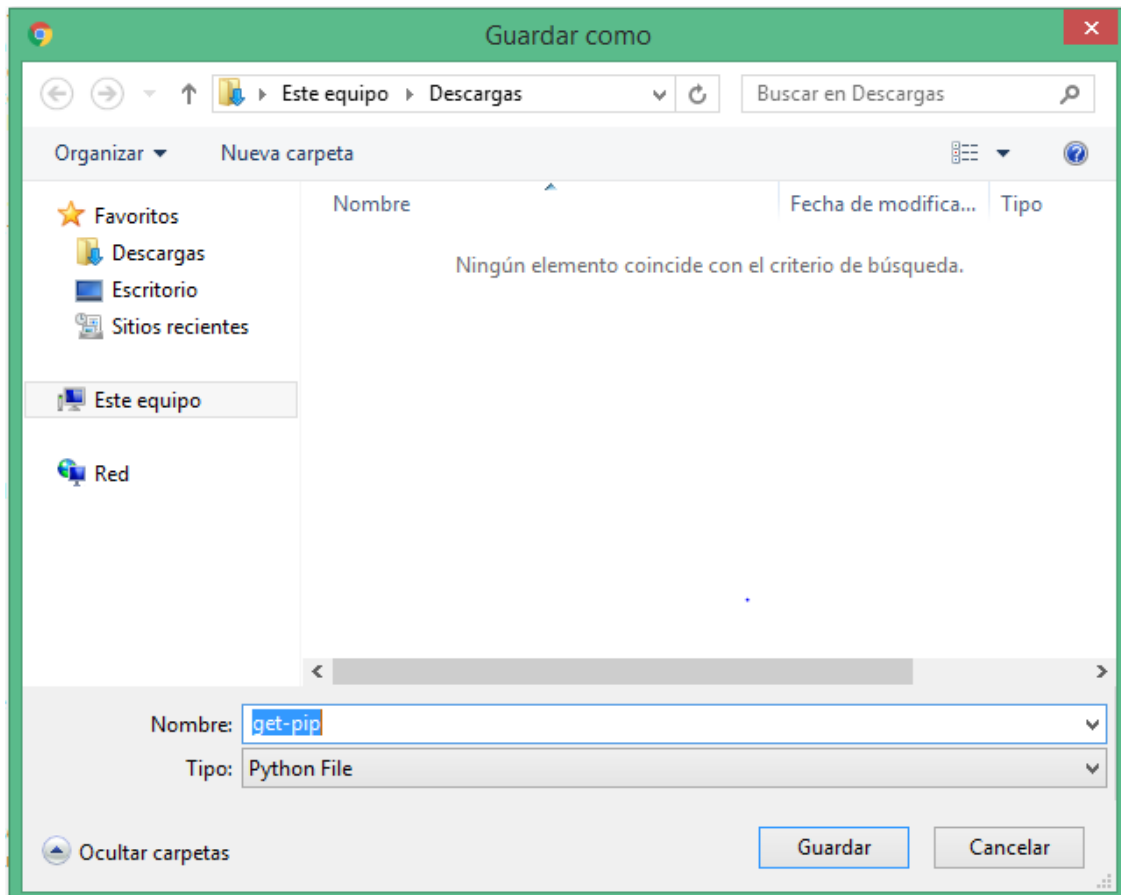


```

#!/usr/bin/env python
#
# Hi There!
#
# You may be wondering what this giant blob of binary data here is, you might
# even be worried that we're up to something nefarious (good for you for being
# paranoid!). This is a base85 encoding of a zip file, this zip file contains
# an entire copy of pip (version 20.1.1).
#
# Pip is a thing that installs packages, pip itself is a package that someone
# might want to install, especially if they're looking to run this get-pip.py
# script. Pip has a lot of code to deal with the security of installing
# packages, various edge cases on various platforms, and other such sort of
# "tribal knowledge" that has been encoded in its code base. Because of this
# we basically include an entire copy of pip inside this blob. We do this
# because the alternatives are attempt to implement a "minipip" that probably
# doesn't do things correctly and has weird edge cases, or compress pip itself
# down into a single file.
#
# If you're wondering how this is created, it is using an invoke task located
# in tasks/generate.py called "installer". It can be invoked by using
# ``invoke generate.installer``.

```

A continuación se debe guardar como un archivo con extensión .py



Después de guardar el archivo lo ejecutamos con python

```
Símbolo del sistema
c:\>python d:\get-pip.py
C:\Users\erorer\AppData\Local\Programs\Python\Python37\lib\site-packages\setuptools\distutils_patch.py:26: UserWarning: Distutils was imported before Setuptools. This usage is discouraged and may exhibit undesirable behaviors or errors. Please use Setuptools' objects directly or at least import Setuptools first.
  "Distutils was imported before Setuptools. This usage is discouraged "
Collecting pip
  Using cached pip-20.1.1-py2.py3-none-any.whl (1.5 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.1.1
    Uninstalling pip-20.1.1:
      Successfully uninstalled pip-20.1.1
  Successfully installed pip-20.1.1
c:\>pip --version
pip 20.1.1 from c:\users\erorer\AppData\Local\Programs\Python\Python37\lib\site-packages\pip (python 3.7)
c:\>
```

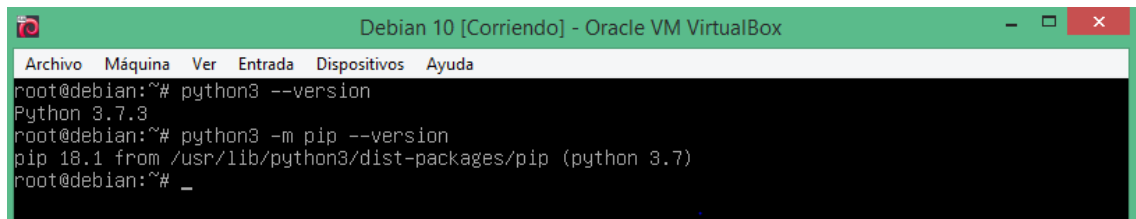
Una vez instalado pip, se procede a instalar las librerías de python necesarias con pip:

- pip install flask
- pip install flask-mysqldb

Instalar Python en Linux (Debian 9)

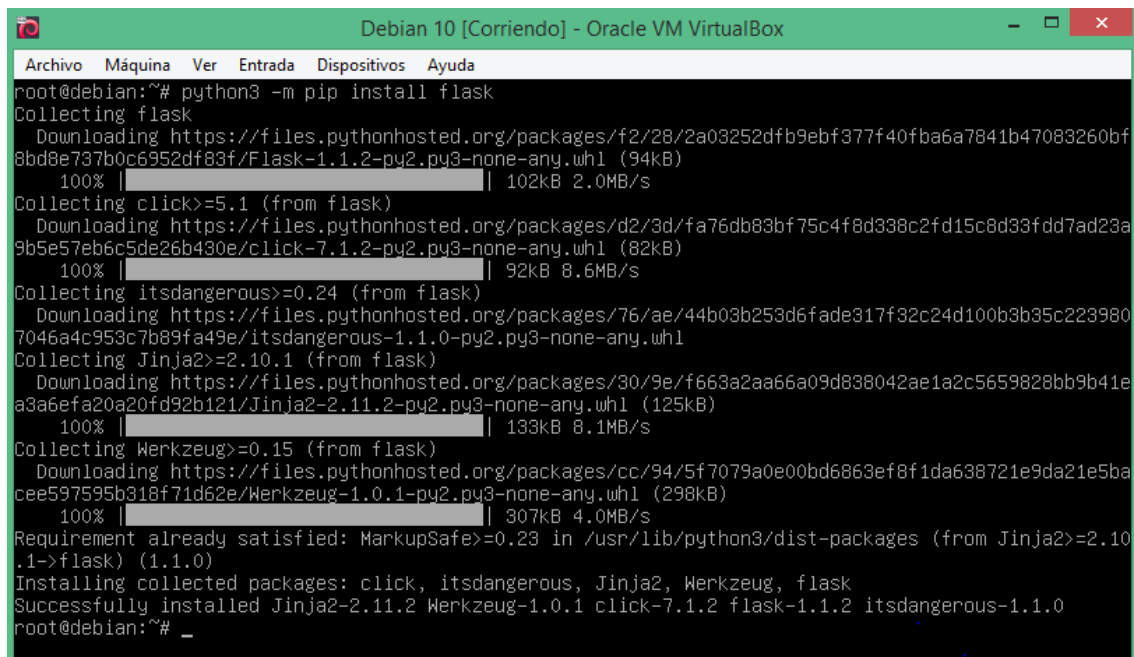
En Linux para instalar python se debe introducir el comando:

- apt install python3.7
- apt install python3-pip
- apt install default-libmysqlclient-dev
- python3 -m pip install flask
- python3 -m pip install flask-mysqldb



```
Debian 10 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@debian:~# python3 --version
Python 3.7.3
root@debian:~# python3 -m pip --version
pip 18.1 from /usr/lib/python3/dist-packages/pip (python 3.7)
root@debian:~# _
```

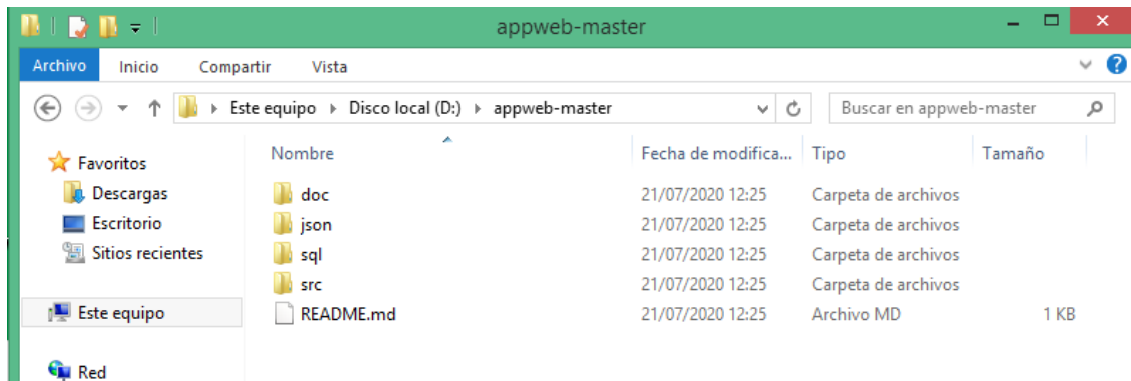
Se instala a continuación las librerías necesarias



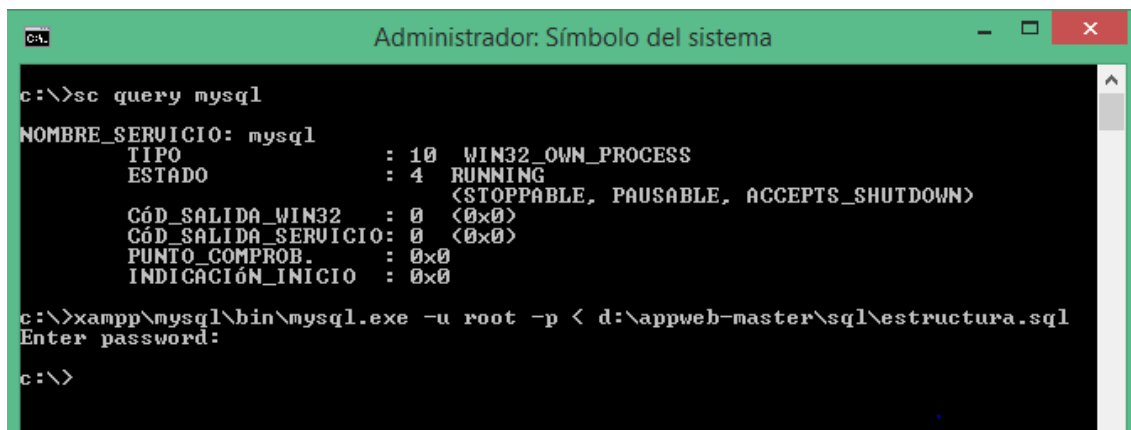
```
Debian 10 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@debian:~# python3 -m pip install flask
Collecting flask
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0c6952df83f/Flask-1.1.2-py2.py3-none-any.whl (94kB)
    100% |#####| 102kB 2.0MB/s
Collecting click>=5.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/d2/3d/fa76db83bf75c4f8d338c2fd15c8d33fdd7ad23a9b5e57eb6c5de26b430e/click-7.1.2-py2.py3-none-any.whl (82kB)
    100% |#####| 92kB 8.6MB/s
Collecting itsdangerous>=0.24 (from flask)
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
    100% |#####| 133kB 8.1MB/s
Collecting Jinja2>=2.10.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/30/9e/f663a2aa66a09d838042ae1a2c5659828bb9b41ea3a6efa20a20fd92b121/Jinja2-2.11.2-py2.py3-none-any.whl (125kB)
    100% |#####| 133kB 8.1MB/s
Collecting Werkzeug>=0.15 (from flask)
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5baee597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl (298kB)
    100% |#####| 307kB 4.0MB/s
Requirement already satisfied: MarkupSafe>=0.23 in /usr/lib/python3/dist-packages (from Jinja2>=2.10.1->flask) (1.1.0)
Installing collected packages: click, itsdangerous, Jinja2, Werkzeug, flask
Successfully installed Jinja2-2.11.2 Werkzeug-1.0.1 click-7.1.2 flask-1.1.2 itsdangerous-1.1.0
root@debian:~# _
```

Descargar y configurar los archivos del repositorio en Windows

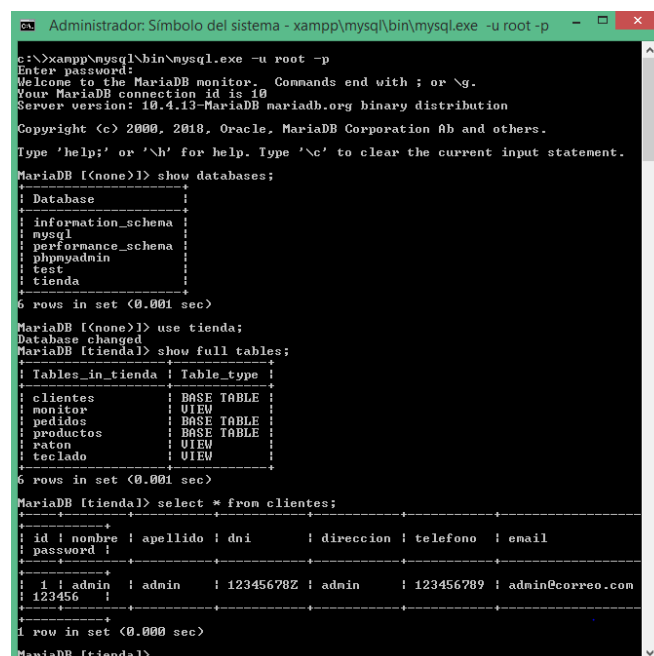
A continuación se descargan los archivos del repositorio “<https://github.com/javihernanp/appweb>”, una vez que tenemos los archivos descargados en el equipo.



Se debe ejecutar el archivo estructura.sql dentro de la carpeta sql de la siguiente forma.



Comprobamos que se ha creado correctamente la base de datos y que en clientes a introducido al usuario “admin” (encargado de la futura administración de la página).



A continuación en la carpeta src se encuentra el código que se ejecutará para poner en funcionamiento la APP. Hay que modificar dentro del archivo “app.py” la ruta del archivo JSON desde el cual se insertarán los datos de los productos en la base de datos.

```

return render_template('carrito.html', pedidos = datos)
'''actualizar tabla productos admin'''

@app.route('/productupdate')
def productupdate():
    file = "D:\\appweb-master\\json\\productos.json" #cambiar ruta
    json_data=open(file).read()
    json_obj = json.loads(json_data)
    cursor = mysql.connection.cursor()
    def validate_string(val):
        if val != None:
            if type(val) is int:
                return str(val).encode('utf-8')

```

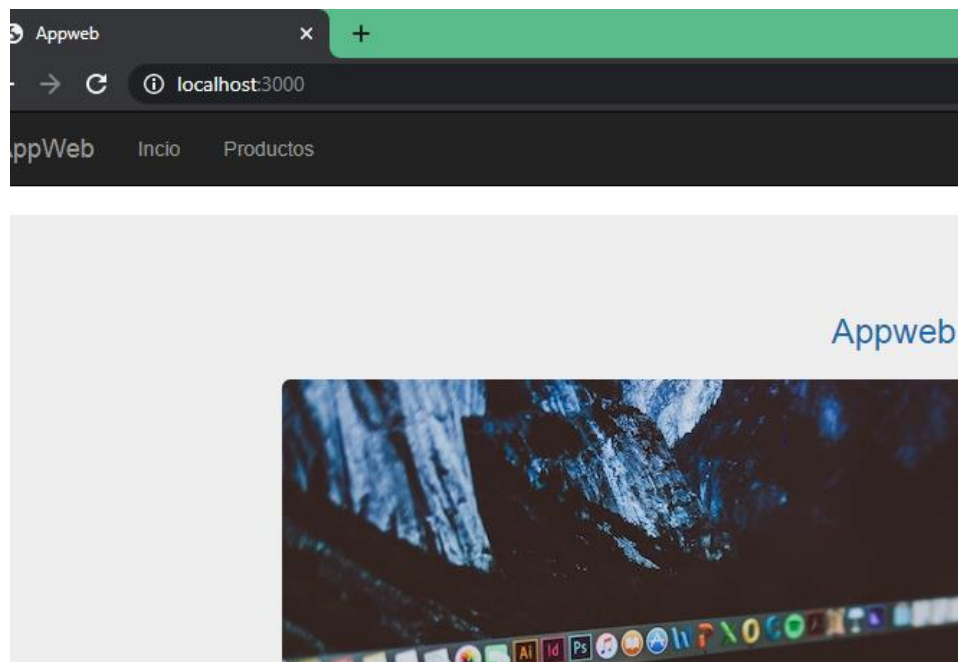
Una vez modificado ya podemos probar a ejecutar el archivo “app.py”

```

C:\> d:\appweb-master\src\app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 300-367-268
* Running on http://127.0.0.1:3000/ (Press CTRL+C to quit)

```

Comprobamos que se ve en el navegador



Iniciamos sesión con el usuario admin (Email: admin@correo.com , Password: 123456). Cuando hemos iniciado sesión pulsamos en el enlace de actualizar tabla productos, para cargar los datos de los productos en la base de datos

Appweb

localhost:3000/profile/1

AppWeb Incio Productos

Carrito

Nombre	Apellido	DNI	Dirección	Teléfono	Email	Opciones	Opciones Admin
Admin	admin	12345678Z	admin	123456789	admin@correo.com	edit delete	actualizar tabla productos carrito

localhost:3000 dice

¿Estas seguro de que deseas restaurar la tabla productos?, Se vaciara los datos de todos los carritos

[Aceptar](#) [Cancelar](#)

```
MariaDB [tienda]> select * from productos;
```

id	nombre	precio	tipo	descripcion
1	monitor a	23	monitor	aqui se pone la descripcion del producto
2	monitor b	23	monitor	aqui esta la descripcion
3	monitor c	23	monitor	aqui esta la descripcion
7	raton a	23	raton	aqui esta la descripcion
8	raton b	23	raton	aqui esta la descripcion
5	teclado b	23	teclado	aqui esta la descripcion
6	teclado c	23	teclado	aqui esta la descripcion
4	teclado z	23	teclado	aqui esta la descripcion

```
8 rows in set (0.000 sec)
```

```
MariaDB [tienda]>
```

Ver todos los productos

Monitores

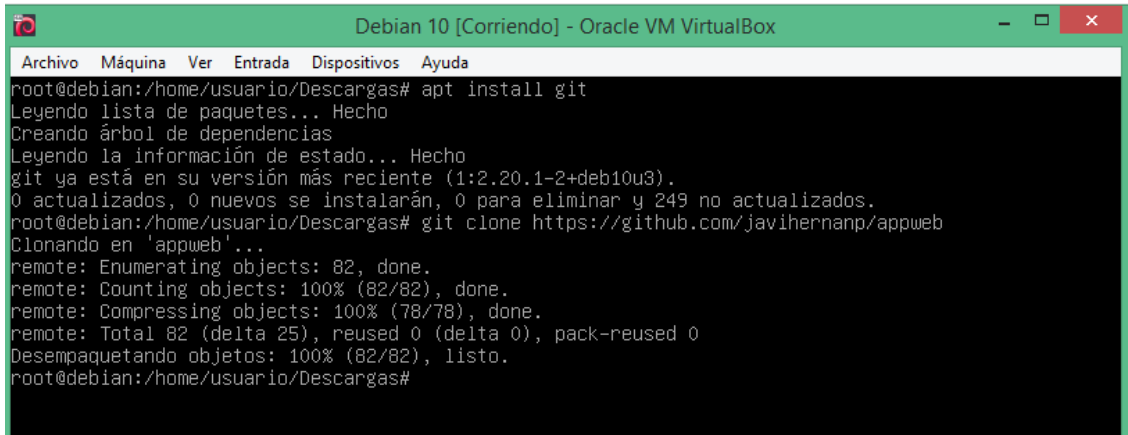
Teclados

Ratones

Nombre	Precio	tipo	cantidad	Opciones	Opadmin
monitor a	23	monitor	11	Detalles Comprar	Editar Borrar
monitor b	23	monitor	11	Detalles Comprar	Editar Borrar
monitor c	23	monitor	11	Detalles Comprar	Editar Borrar
raton a	23	raton	11	Detalles Comprar	Editar Borrar
raton b	23	raton	11	Detalles Comprar	Editar Borrar
teclado b	23	teclado	11	Detalles Comprar	Editar Borrar
teclado c	23	teclado	11	Detalles Comprar	Editar Borrar
teclado z	23	teclado	11	Detalles Comprar	Editar Borrar

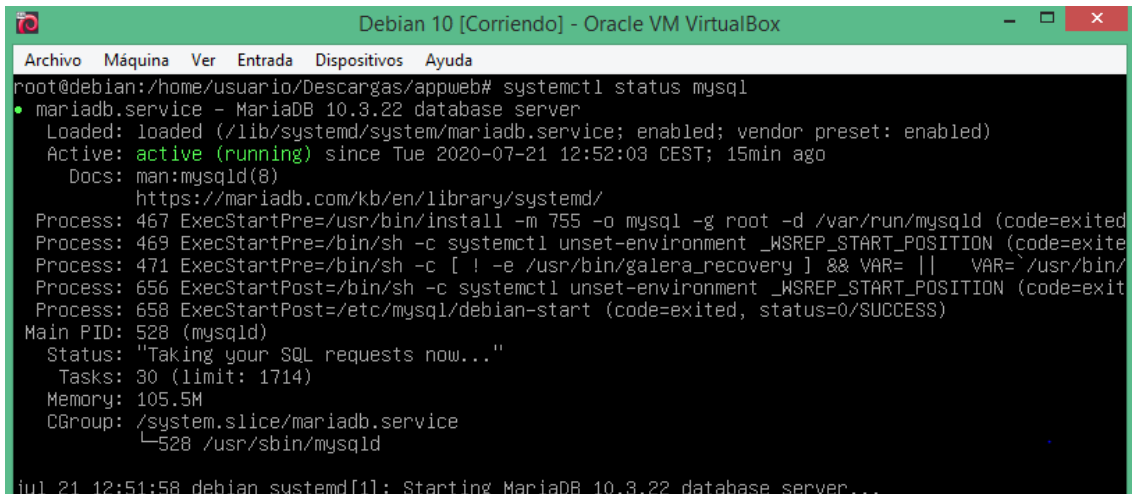
Descargar y configurar los archivos del repositorio en Linux

Se descarga con git el contenido del repositorio



```
Debian 10 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@debian:/home/usuario/Descargas# apt install git
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
git ya está en su versión más reciente (1:2.20.1-2+deb10u3).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 249 no actualizados.
root@debian:/home/usuario/Descargas# git clone https://github.com/javihernanp/appweb
Clonando en 'appweb'...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (78/78), done.
remote: Total 82 (delta 25), reused 0 (delta 0), pack-reused 0
Desempaquetando objetos: 100% (82/82), listo.
root@debian:/home/usuario/Descargas#
```

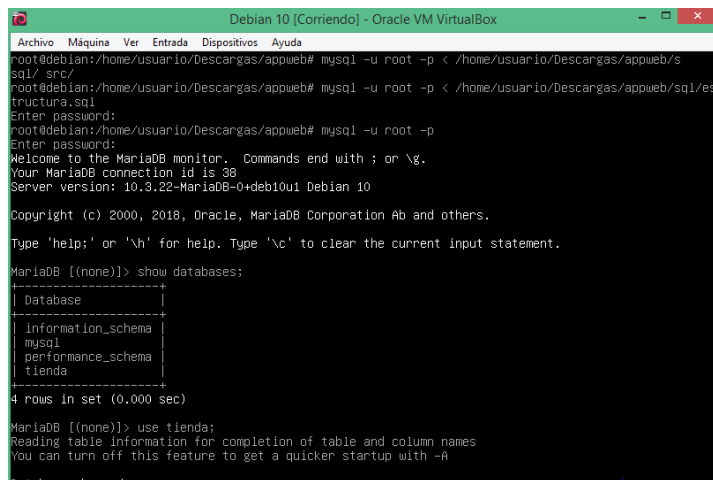
A continuación se comprueba el estado del servicio mysql



```
Debian 10 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@debian:/home/usuario/Descargas/appweb# systemctl status mysql
• mariadb.service - MariaDB 10.3.22 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-07-21 12:52:03 CEST; 15min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 467 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysql (code=exited)
   Process: 469 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited)
   Process: 471 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR=/usr/bin/
   Process: 656 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exit)
   Process: 658 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
  Main PID: 528 (mysqld)
    Status: "Taking your SQL requests now..."
     Tasks: 30 (limit: 1714)
    Memory: 105.5M
    CGroup: /system.slice/mariadb.service
           └─528 /usr/sbin/mysqld

jul 21 12:51:58 debian systemd[1]: Starting MariaDB 10.3.22 database server...
```

Se crea la base de datos necesaria para el funcionamiento de la APP y se comprueba los datos que debe tener por defecto.



```
Debian 10 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@debian:/home/usuario/Descargas/appweb# mysql -u root -p < /home/usuario/Descargas/appweb/s
sql/ src/
root@debian:/home/usuario/Descargas/appweb# mysql -u root -p < /home/usuario/Descargas/appweb/sql/es
tructura.sql
Enter password:
root@debian:/home/usuario/Descargas/appweb# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 10.3.22-MariaDB-0+deb10u1 Debian 10

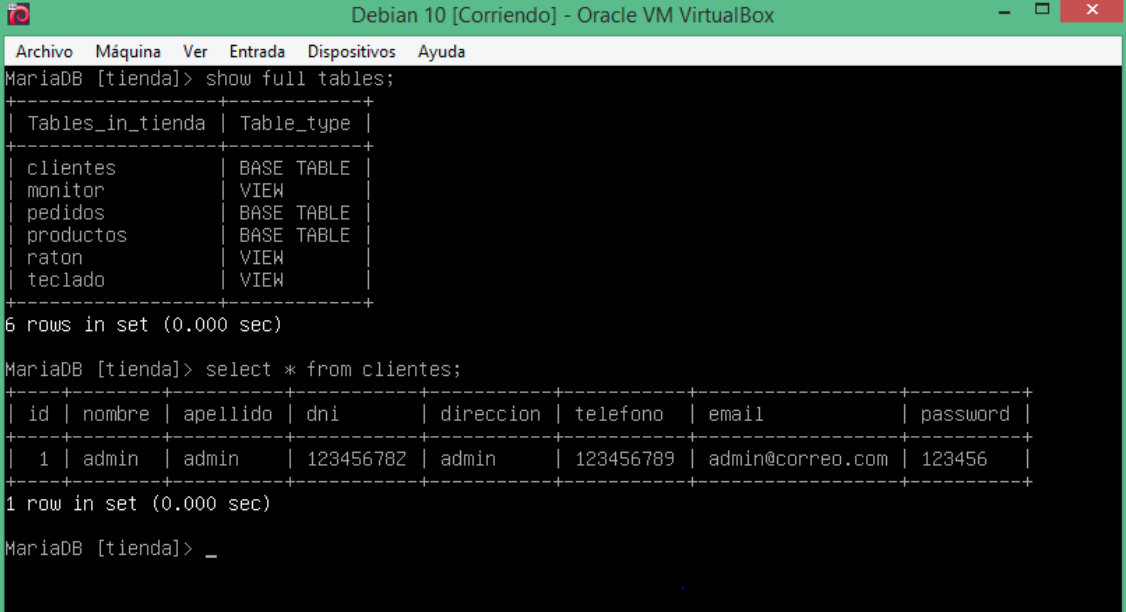
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| tienda |
+-----+
4 rows in set (0.000 sec)

MariaDB [(none)]> use tienda;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

database changed
```



```

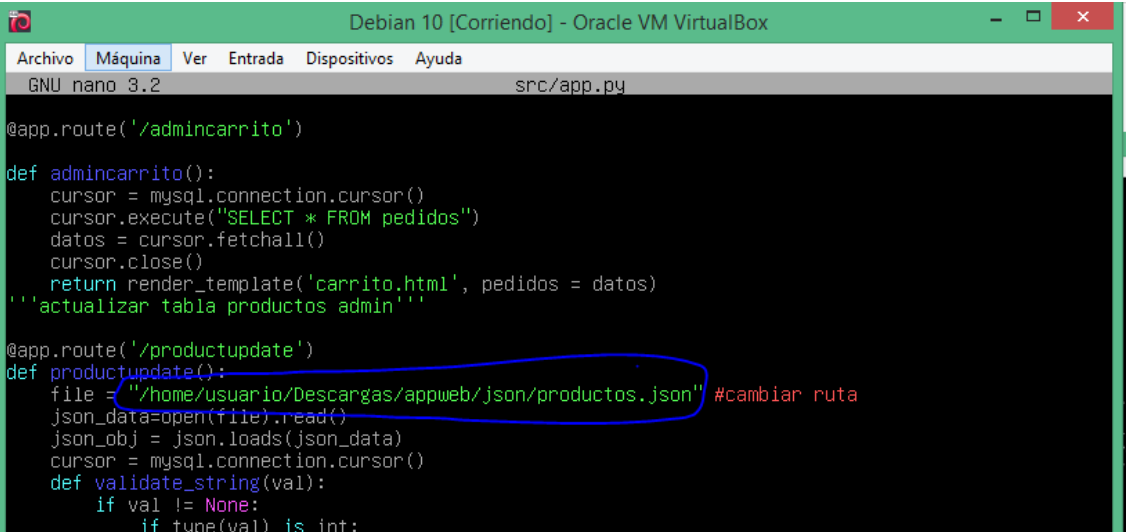
Debian 10 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
MariaDB [tienda]> show full tables;
+-----+-----+
| Tables_in_tienda | Table_type |
+-----+-----+
| clientes          | BASE TABLE |
| monitor           | VIEW        |
| pedidos           | BASE TABLE |
| productos         | BASE TABLE |
| raton             | VIEW        |
| teclado           | VIEW        |
+-----+-----+
6 rows in set (0.000 sec)

MariaDB [tienda]> select * from clientes;
+----+-----+-----+-----+-----+-----+-----+
| id | nombre | apellido | dni      | direccion | telefono | email          | password |
+----+-----+-----+-----+-----+-----+-----+
| 1  | admin  | admin    | 123456782 | admin     | 123456789 | admin@correo.com | 123456   |
+----+-----+-----+-----+-----+-----+-----+
1 row in set (0.000 sec)

MariaDB [tienda]> _

```

A continuación se modifica el archivo app.py que está en la carpeta SRC con la ruta donde este el archivo JSON que contienen los datos sobre los productos



```

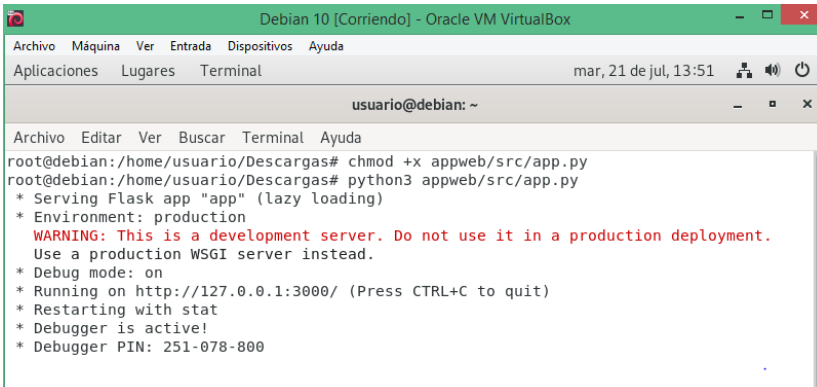
Debian 10 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
GNU nano 3.2 src/app.py

@app.route('/admincarrito')
def admincarrito():
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM pedidos")
    datos = cursor.fetchall()
    cursor.close()
    return render_template('carrito.html', pedidos = datos)
'''actualizar tabla productos admin'''

@app.route('/productupdate')
def productupdate():
    file = "/home/usuario/Descargas/appweb/json/productos.json" #cambiar ruta
    json_data=open(file).read()
    json_obj = json.loads(json_data)
    cursor = mysql.connection.cursor()
    def validate_string(val):
        if val != None:
            if type(val) is int:

```

Una vez modificado el archivo se pone en marcha el servidor



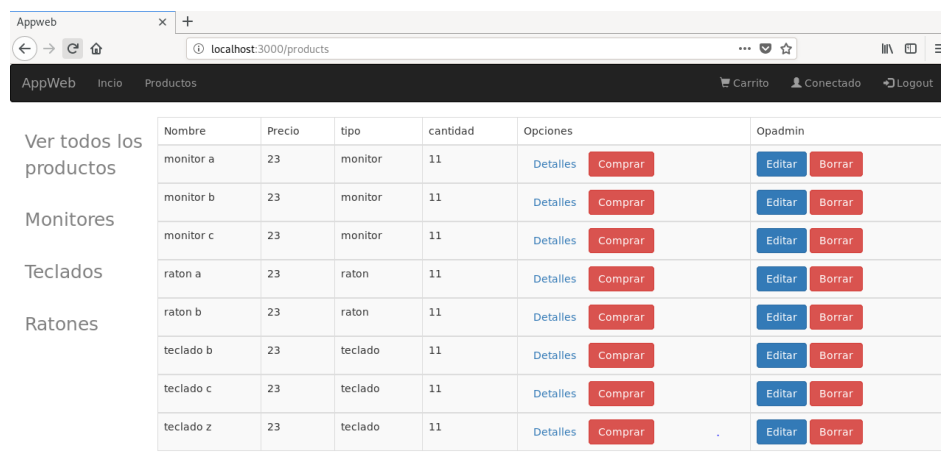
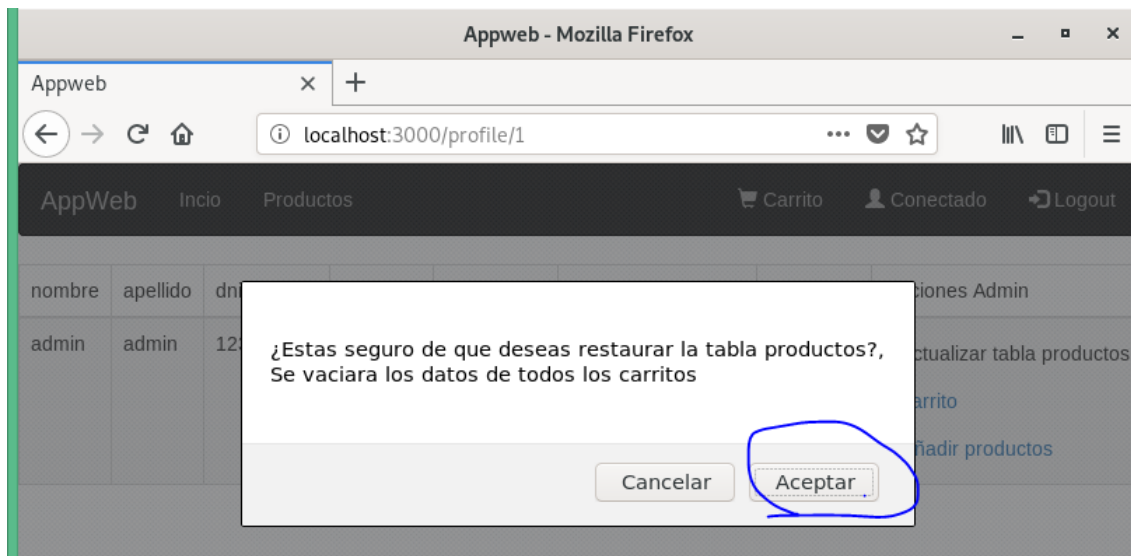
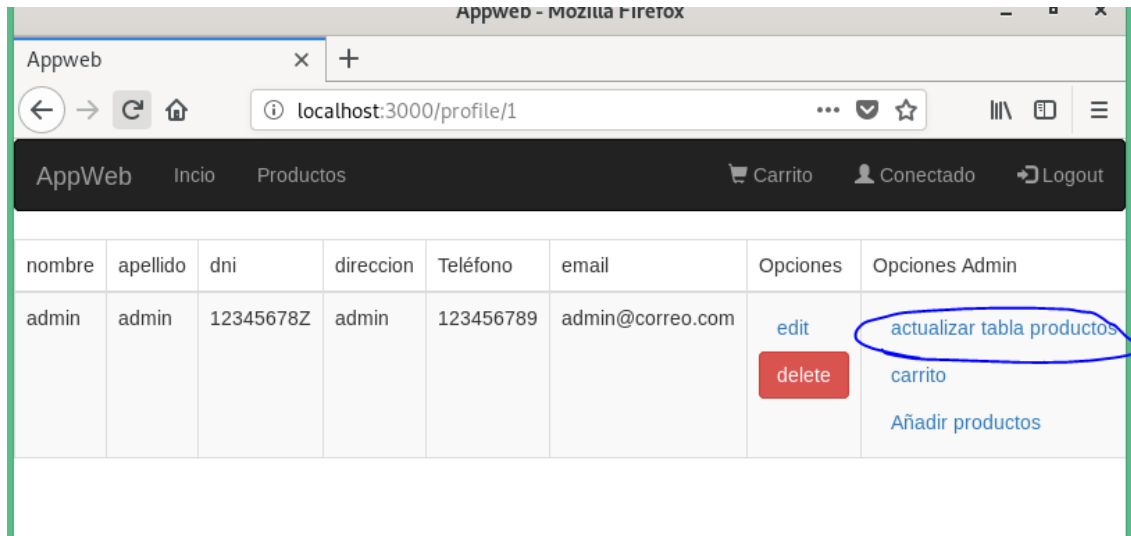
```

Debian 10 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Aplicaciones Lugares Terminal mar, 21 de jul, 13:51
usuario@debian: ~

Archivo Editar Ver Buscar Terminal Ayuda
root@debian:/home/usuario/Descargas# chmod +x appweb/src/app.py
root@debian:/home/usuario/Descargas# python3 appweb/src/app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:3000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-078-800

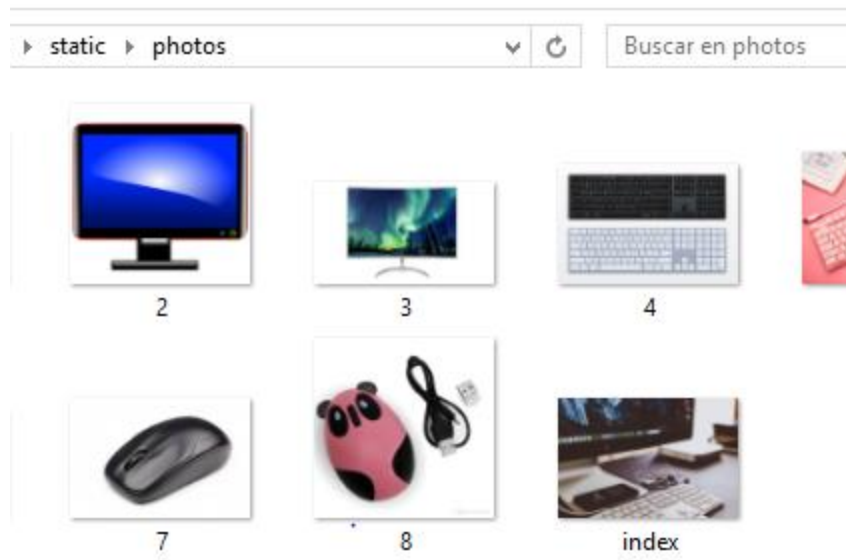
```

Iniciamos sesión con el usuario admin (Email: admin@correo.com , Password: 123456). Cuando hemos iniciado sesión pulsamos en el enlace de actualizar tabla productos, para cargar los datos de los productos en la base de datos



Con esto se da por finalizado la instalación y configuración de la aplicación WEB. Cada vez que se añada un producto, se debe añadir una imagen JPG en la carpeta photos

con el nombre de archivo <id>.jpg, para que cuando muestres los detalles del producto se muestre correctamente la imagen



Guía de Usuario

Tras la instalación se procede a explicar las distintas partes del código y como entra en funcionamiento con la aplicación.

Se comienza a explicar el archivo “app.py”, puesto que, el archivo estructuras.sql se explicará con más profundidad en el apartado donde se desarrollará el diagrama entidad-relación y el relacional.

El archivo app.py comienza importando las librerías necesarias, inicializando el módulo para saber dónde se encuentra y configurar las rutas de los archivos que ofrecerá el servidor, el módulo secret_key se encargará de mantener los datos seguros y por último especificamos los datos necesarios para conectarse con la base de datos.

```
from flask import Flask, render_template, request, redirect, url_for, flash, session, logging, json, jsonify
from flask_mysql import MySQL

import os
from functools import wraps

app = Flask(__name__)

app.secret_key = os.urandom(24)

# MySQL configuracion
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'tienda'

mysql = MySQL(app)
```

A continuación se utilizan decoradores para comprobar si se ha iniciado sesión o no, un decorador son funciones que añaden funcionalidades a otras funciones, la estructura de un decorador, es la de 3 funciones (A, B y C) donde A recibe como parámetro la función

B para retornar la función C, es decir, un decorador devuelve una función un ejemplo sería lo siguiente:

```
usuario@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 3.2 decorador.py

def funcion_a(funcion_b):
    def funcion_c():
        print('Antes de la ejecucion')
        funcion_b()
        print('Despues de la ejecucion')
    return funcion_c

@funcion_a
def prueba():
    print('hola')

prueba()
```

```
usuario@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@debian:/home/usuario# python3 decorador.py
Antes de la ejecucion
hola
Despues de la ejecucion
root@debian:/home/usuario#
```

Una vez entendido como funciona un decorador se procede a explicar el decorador usado en el script

```
'''Decoradores para comprobar si estan logueados'''
def not_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return redirect(url_for('index'))
        else:
            return f(*args, *kwargs)

    return wrap

def is_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return f(*args, *kwargs)
        else:
            return redirect(url_for('login'))

    return wrap
```


Wrap es una un decorador en sí mismo encargado de copiar sobre el mismo el nombre de la función, en definitiva cada decorador (not_logged_in, is_logged_in) va a recibir la situación de la página en cada momento e interpretar si ha iniciado o no sesión dependiendo de cuál de las dos utilices.

A continuación se empieza a definir las rutas en las cuales se especificará que tareas realizará el servidor y que devolverá a las peticiones que hacen los clientes un ejemplo sería:

```
'''Registro commit usuario'''
@app.route('/process', methods=['POST'])
def process():
    # connect

    cursor=mysql.connection.cursor()
    msg = ''
    name = request.form['name']
    apellido = request.form['apellido']
    dni = request.form['dni']
    direccion = request.form['direccion']
    phone = request.form['phone']
    email = request.form['email']
    password = request.form['password']

    if name and apellido and dni and email and password:
        try:
            cursor.execute('INSERT INTO clientes VALUES (NULL, %s, %s, %s, %s, %s, %s)', (name, apellido, dni, direccion, phone, email, password))
            mysql.connection.commit()
            cursor.close()
        except:
            return jsonify({'error': 'Revisa el DNI o ese correo no está disponible'})
        msg = 'Registrado ahora puedes hacer login'
        return jsonify({'name': msg})

    return jsonify({'error': 'Error al enviar los datos'})
```

En este caso la ruta “/process”, recibe los datos de un formulario y los procesa y devolverá a esta petición un mensaje en formato JSON que se encargará de procesarlo el cliente.

Otro ejemplo de rutas sería el siguiente:

```
'''carrito usuario'''
@app.route('/carrito/<id>')

def carrito(id):
    correo=session['_s_name']
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM pedidos WHERE emailcliente=%s", [correo])
    datos = cursor.fetchall()
    cursor.close()

    return render_template('carrito.html', pedidos = datos)

'''borrar pedido'''
```

El cual nos hacen una petición GET de una página y devolvemos el archivo.html y un array pedidos que contiene los datos que se han obtenido como respuesta a esa consulta SQL.

El resto del archivo app.py funciona de manera muy similar pero hay que explicar porque hay que poner 3000 para acceder al sitio y es debido a esta parte:

```
if __name__ == '__main__':
    app.run(port= 3000, debug= True)
```

En port le indicamos el puerto que estará a la escucha de las peticiones y con debug que cada vez que se realice un cambio en el archivo app.py se reinicie la ejecución del mismo.

Con esto se acaba la explicación del archivo app.py, debido a que el resto del archivo es muy similar a lo que ya se ha explicado y se procede a explicar la parte del front-end centrándose en jinja2 y en algunos scripts de jquery que se han usado.

Se empieza explicando por ejemplo carrito.html para explicar lo que devuelve el servidor de python, es decir, el archivo html y el array de datos.

```

</thead>
<tbody>
  {% for pedido in pedidos %}

    <tr>

      <td>{{pedido.1}}</td>
      <td>{{pedido.2}}</td>
      <td>{{pedido.3}}</td>
      <td>{{pedido.4}}</td>

      <td>
        <a href="/DeletePedido/{{pedido.0}}" class="btn btn-danger btn-delete">Borrar pedido</a>

      </td>
    </tr>

  </tbody>
</table>
{% endfor %}
<a href="/profile/{{session.uid}}" class="btn btn-secondary">Volver</a>

```

Como podemos ver en jinja2 la forma de recorrer arrays es muy similar en python, el array pedidos es lo que devuelve el servidor junto con el archivo html, crea la vista en el servidor y envía el archivo html ya con los datos correspondientes al cliente que es lo que ve en el navegador

Appweb

- Inicio
- Productos
- Carrito
- Conectado
- Logout

Correo_cliente	id_producto	nombre_producto	fecha_pedido	
admin@correo.com	1	monitor a	2020-07-21 15:19:47	Borrar pedido

```

<!DOCTYPE html>
<html lang="en">
<head></head>
<body>== $@
  <nav class="navbar navbar-inverse"></nav>
  <div class="main">
    <table class="table table-striped table-hover table-bordered table-sm bg-white">
      <thead></thead>
      <tbody>
        <tr>
          <td>admin@correo.com</td>
          <td>1</td>
          <td>monitor a</td>
          <td>2020-07-21 15:19:47</td>
          <td></td>
        </tr>
      </tbody>
    </table>
    <a href="/profile/1" class="btn btn-secondary">Volver</a>
  </body>

```

Otra parte de jinja2 que cabe explicar es la parte de reutilización de código html

```

</div>
</nav>
{% block body%}
<div class="text-center jumbotron">
  <h3 class="text-center titulo">Appweb</h3>
  
</div>

{% endblock%}

<div class="footer">
  <p>página de prueba de AppWeb</p>
</div>

```

Mediante las etiquetas block body y endblock le indicamos la parte que podrá cambiar de una página a otra donde se decida extender la plantilla por ejemplo en carrito.html:

```

{% extends "index.html" %}
{% block body %}

<div class="main">

<table class="table table-striped table-hover table-bordered table-sm bg-white">
<thead>
<tr>
<td>Correo_cliente</td>
<td>id_producto</td>
<td>nombre_producto</td>
<td>fecha_pedido</td>
</tr>
</thead>
<tbody>
{% for pedido in pedidos %}
<tr>
<td>{{pedido.1}}</td>
<td>{{pedido.2}}</td>
<td>{{pedido.3}}</td>
<td>{{pedido.4}}</td>
<td>
<a href="/DeletePedido/{{pedido.0}}" class="btn btn-danger btn-delete">Borrar ped
</td>
</tr>
</tbody>
</table>
<a href="/profile/{{session.uid}}" class="btn btn-secondary">Volver</a>
<a href="#" class="btn btn-secondary">Comprar</a>
</div>
{% endblock %}

```

Le indicamos de donde tiene que extender el html, en este caso es desde index y le indicamos en la parte que puede cambiar ({% block body%} y {%endblock%}) definimos los cambios que habrá respecto a la página Index.html

Con esto acaba la explicación del uso general que se le ha dado a jinja2 para crear vistas. A continuación se analiza algunos scripts de JQuery.

El primer script que se va a analizar es el script encargado de recoger los datos de la página login.html y de analizar la respuesta del servidor

```

$(document).ready(function() {
    $(".btn-enviar").click(function(){
        var email = $('#email').val();
        var pass = $('#password').val();
        var ExpressionEmail=/^[a-zA-Z0-9\._-]+@[a-zA-Z0-9-]{2,}\.[a-zA-Z]{2,4}$/;
        var Expressionpassword=/^[A-Za-z0-9$]{3,25}$/g
        if (!ExpressionEmail.test(email)){
            $('#errorAlert').text('pon un correo válido').show();
            return false;
        } else {
            if (!Expressionpassword.test(pass)) {
                $('#errorAlert').text('pon una contraseña (solo texto)').show();
                return false;
            } else {
                return true;
            }
        }
    });
    $('#form').on('submit', function(event) {
        $.ajax({
            data : {
                email : $('#email').val(),
                password : $('#password').val()
            },
            type : 'POST',
            url : '/login'
        })
        .done(function(data) {
            if (data.error) {
                $('#errorAlert').text(data.error).show();
                $('#cform')[0].reset();
            }
            else {
                window.location.href = "http://" + window.location.host + "";
            }
        });
        event.preventDefault();
    });
});

```

El script se ejecuta cuando el documento ya se ha cargado completamente debido a la primera línea del script, la segunda función se ejecuta cuando se pulsa en el botón que tiene la clase btn-enviar, y está realiza la validación de los campos para que concuerde con la expresión regular y en caso de que no sea así muestra mensaje en un div que inicialmente tenía la opción de `display:none;` en el css y con el “.show” cambia el css para que ese div se empiece a mostrar si no se cumple.

El script sigue con una función que se ejecuta cuando se produce el envío de datos en el cual se envía los datos recogidos en el campo email y password, se envía con el método “post” a la URL “/login”. El servidor procesa los datos en la dirección “/login” y si la respuesta que devuelve el servidor contiene el campo “error” borra el contenido de todos los campos del formulario y hace visible un “div” con un mensaje de error, en caso de que se no devuelva condición de error redirige la página ya que window.location.href no es un método, es una propiedad que le indicará la ubicación actual de la URL del navegador. Cambiar el valor de la propiedad redirigirá la página.

El resto de scripts tienen un funcionamiento similar a excepción de los scripts encargados de cargar los filtros desde datos devueltos por el servidor en formato JSON

```
$(document).ready(function () {

    $.ajax({
        url: '/mp',
        dataType: 'json',
        type: 'POST',

        success: function(payload) {
            var data = JSON.parse(payload);
            console.log(data);

            for (var i in data){
                var person=data[i];

                $("tbody").append('<tr><td>'+person.nombre+'</td><td>'+person.precio+'</td><td>'+person.cantidad+'</td><td><a href="/'+person.id +"'>'+class
```

En estos script los datos que devuelve el servidor son en formato JSON, y así se lo debemos indicar a Ajax si el envío de los datos ha tenido éxito se ejecuta una función en el que se le pasa como parámetro dichos datos en JSON, se guardan en una variable, se muestran en consola (Solo durante la etapa de desarrolló) y acto seguido se recorre como un array con un “for” y se añade en el html por debajo del “tbody” especificado en el html.

El último tipo de script que se debe analizar es el de confirmación que impedirá eliminar pedidos, clientes o productos sin una confirmación previa

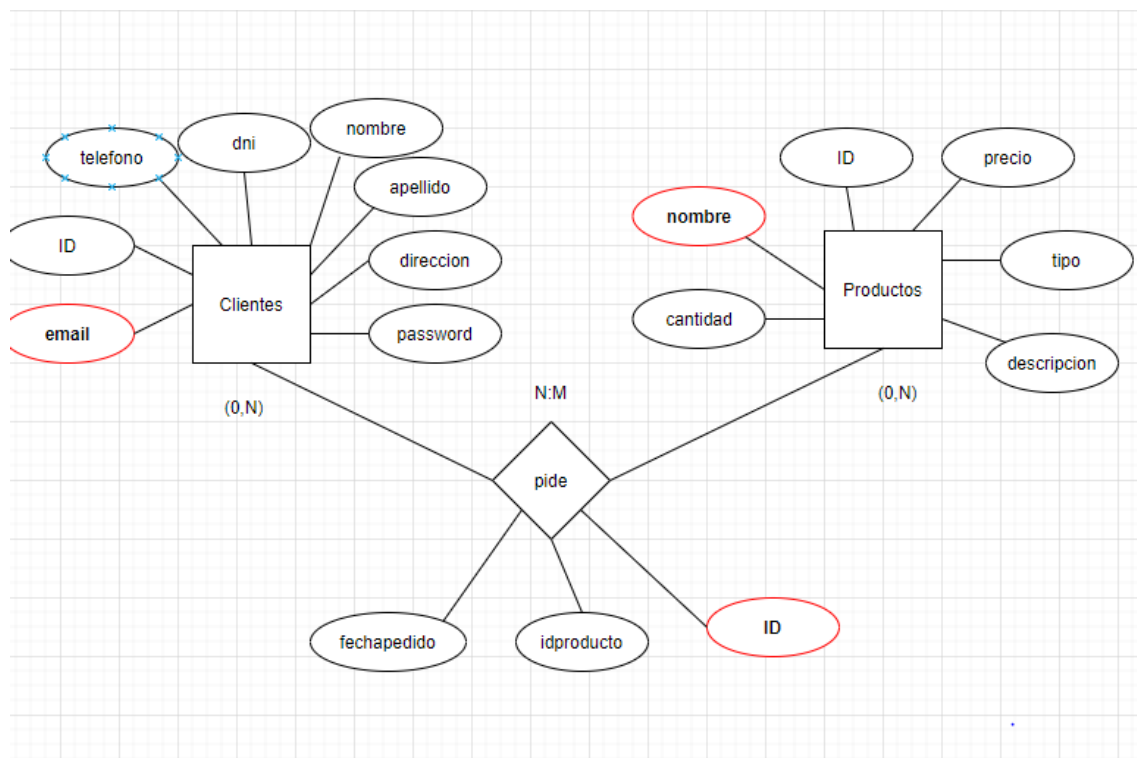
```
$(document).ready(function () {
    $(".btn-delete").click(function () {
        if (!confirm("¿Estas seguro de que deseas borrarlo?")){
            return false;
        }
    });
    $(".btn-delete-tabla").click(function () {
        if (!confirm("¿Estas seguro de que deseas restaurar la tabla productos?, Se vaciara los datos de todos los carritos")){
            return false;
        }
    });
    $(".btn-delete-product").click(function () {
        if (!confirm("¿Estas seguro de que deseas borrar el producto?")){
            return false;
        }
    });
});
});
```

Las distintas funciones se ejecutarán cuando se pulsen sobre los botones con ese nombre de clase, en caso de que no se pulse aceptar en el alert que saldrá en pantalla no se eliminará ese cliente, producto o pedido.

Tras explicar los scripts principales se concluye la guía del usuario en el que se le hace una breve introducción al código de la aplicación y de cómo funciona todo en conjunto.

Diagrama Entidad-Relación y Modelo Relacional

El diagrama Entidad-Relación empleado para desarrollar sería el siguiente:



La explicación de este diagrama es la siguiente: Un producto siempre es pedido por cero o varios cliente (0, N) y un cliente puede no pedir, o pedir muchos productos (0, N). Al obtener los máximos de cada participación da como resultado una cardinalidad N:M. En este caso necesitamos una clave en la relación, ya que los productos no se agrupan en un solo pedido, sino que se van listando. Con lo que la relación tiene un atributo que será clave en la futura clave compuesta.

Una vez realizado el modelo entidad-relación, se procede a realizar el modelo relacional. Se observa se obtiene una cardinalidad N:M, las cardinalidades N:M dan lugar a tablas, la tabla estará formada por las claves primarias de las entidades que relaciona la relación y por los atributos de la relación. En este caso la clave estará formada por las claves primarias de las entidades clientes y productos y por el atributo id.

- Clientes: *id*, nombre, apellido, DNI, dirección, teléfono, **email**, password
- Productos: *id*, **nombre**, precio, tipo, descripción, cantidad

- Pedidos: **id**, **emailcliente**, idproducto, **nombre_producto**, fechapedido

Las claves primarias de las tablas Clientes y Productos se pasan como claves foráneas a la tabla pedidos y forman junto con id la clave primaria.

Conclusión

El proyecto me ha parecido muy instructivo a la vez que interesante, debido a que he tenido que informarme y leer bastante documentación sobre tecnologías que no dominaba. Además este proyecto me ha servido para crear mi primera aplicación web encargándome del desarrollo back-end y front-end, cosa que no había hecho antes de una forma tan completa. Me ha interesado mucho el desarrollo web a raíz de este proyecto y me ha hecho darme cuenta de mi nivel actual y de cuanto me falta por aprender ya que he tenido que revisar bastante documentación para resolver un problema que no era demasiado complejo.

Debido al limitado tiempo del que disponía esta aplicación lejos de estar completa debería buscar el mejorar el planteamiento de las bases de datos, la codificación en sí, ya que no es que aporte mucha seguridad, debido por ejemplo a que no se ha implementado ningún tipo de control a nivel de bases de datos. El CSS de las páginas se debería mejorar mucho y en sí el control de las propias páginas.

Se puede concluir que en la tecnología en las que peor me he desenvuelto es en el CSS ya que es un campo en el que trabajado muy poco.