



DESPLIEGUE DE APLICACIONES WEB

Práctica primer trimestre



DAVID ESPINOSA GONZÁLEZ
Desarrollo de aplicaciones web, IES Palomeras-Vallecas

Índice

Despliegue en un servidor remoto LAMP

Creación de una base de datos

Despliegue de la aplicación web

Acceso a la aplicación web

Despliegue en un servidor de hosting

Creación de una cuenta en el servidor de alojamiento, deploy de archivos de la app y acceso a la web

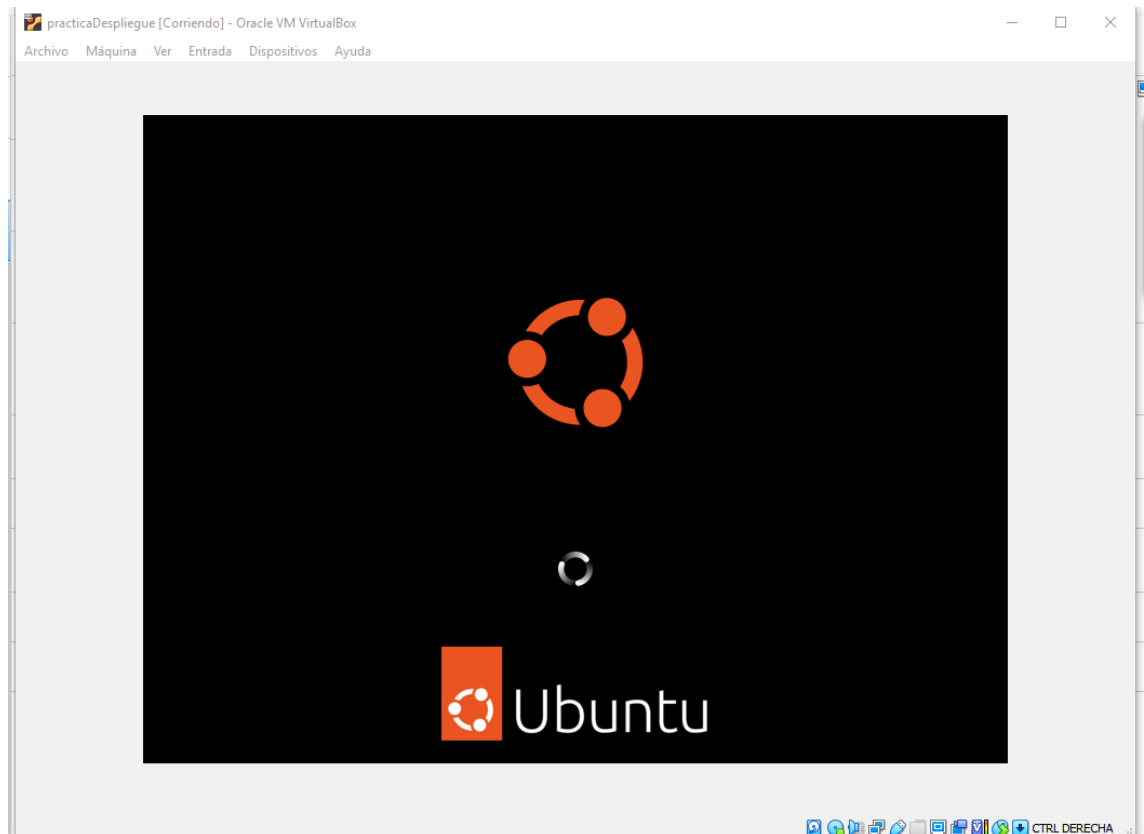
Despliegue web Python en Ubuntu

Despliegue web Python en Heroku

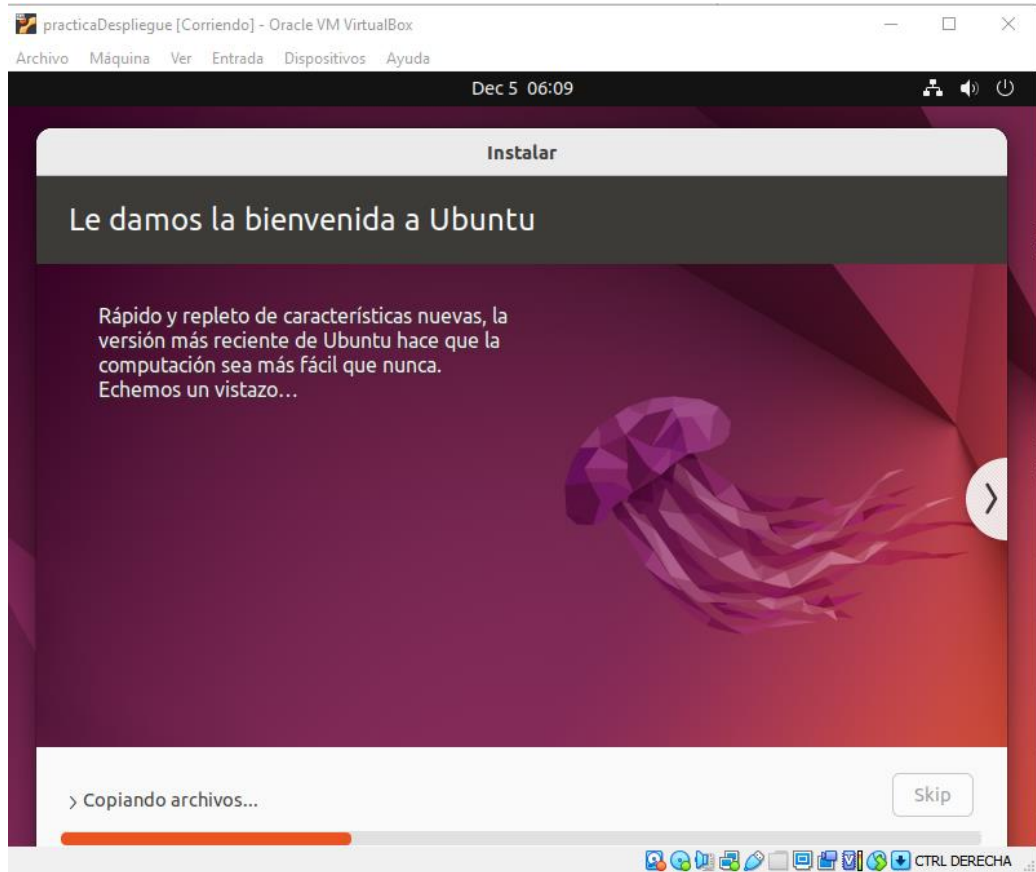
Despliegue en un servidor remoto LAMP

Para comenzar la práctica lo primero que haremos será instalar una máquina virtual y en ella el servidor LAMP. El programa que utilice para realizar la máquina virtual es el VirtualBox, y el entorno usado es Ubuntu.

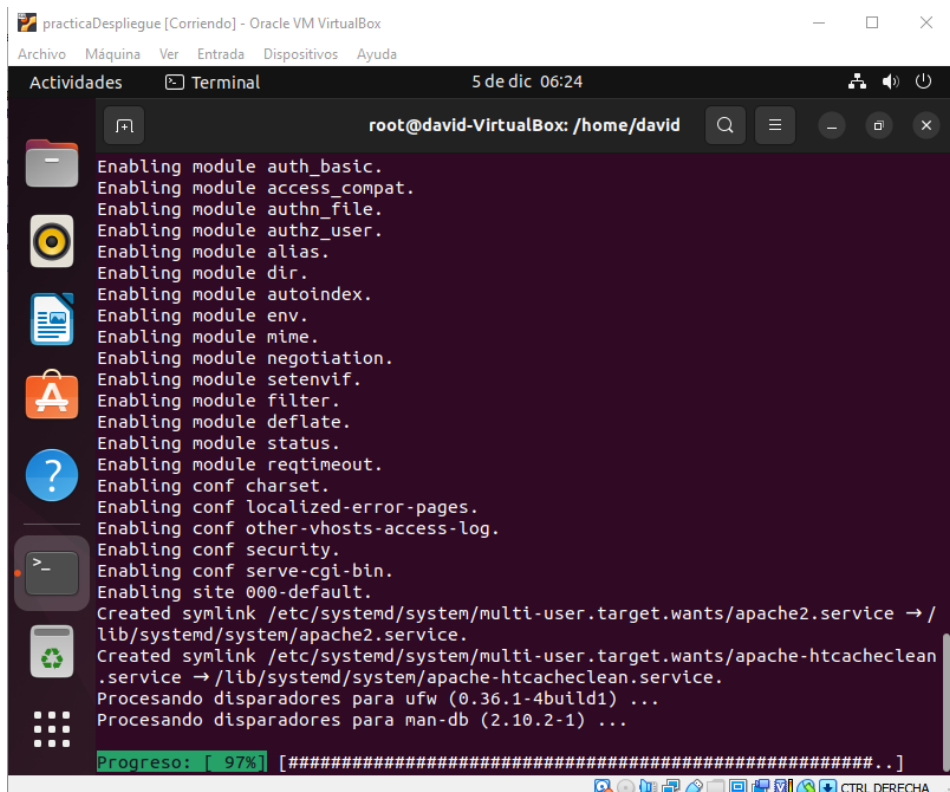
Una vez instalado Ubuntu, se instalará el servidor LAMP, para ello lo primero a instalar es “apache2”, luego se instalará “mariadb” y por último “php”. Una vez instalados todos estos componentes se procederá a instalar el phpmyadmin para poder manejar las bases de datos del entorno virtual. Todo ello lo trabajaremos en el Localhost, por lo que aún no se podrá acceder al entorno desde fuera del entorno. A continuación, se mostrarán unas imágenes del proceso seguido.



La siguiente imagen corresponde a la instalación de Ubuntu.



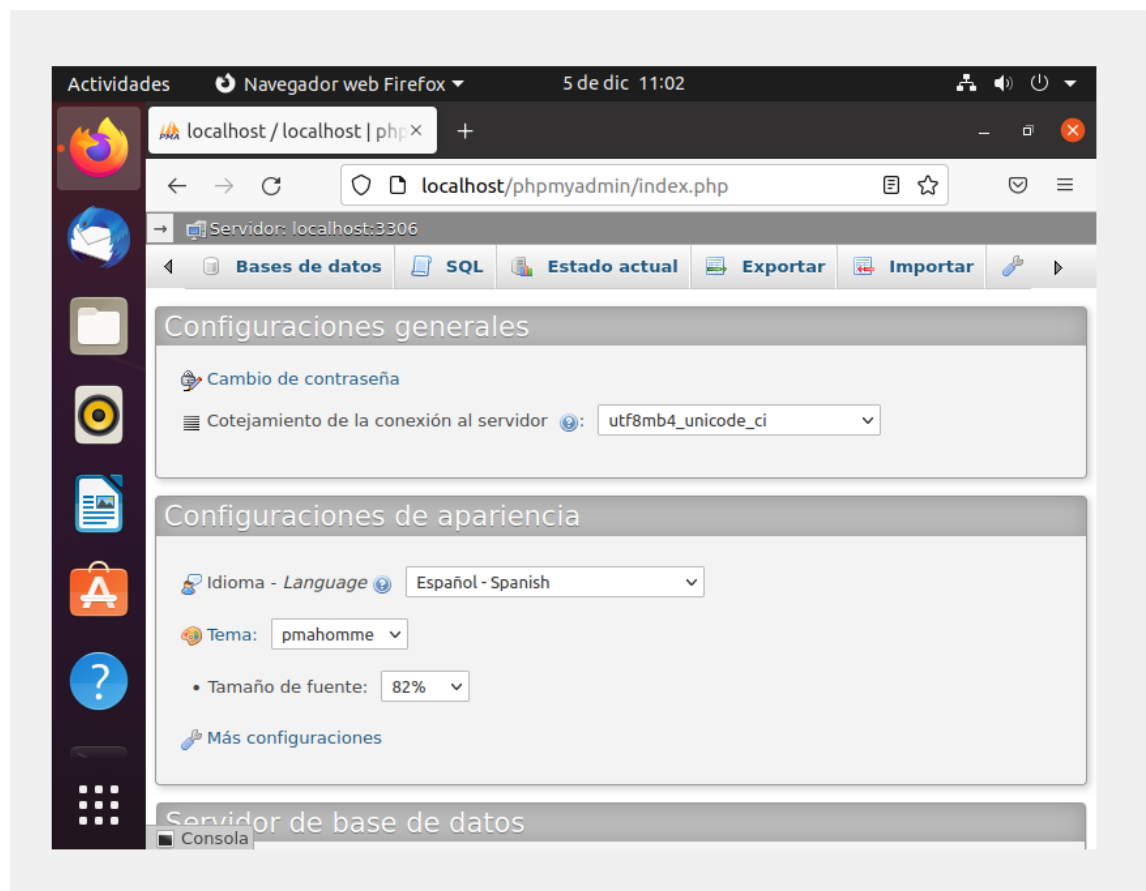
Una vez terminada la instalación se procede a instalar las Guest-Additions del entorno virtual para poder desarrollar mejor el trabajo en el entorno. Una vez instaladas comenzaremos a instalar el servidor LAMP.



A continuación, se muestra la versión de php en que se trabajara.

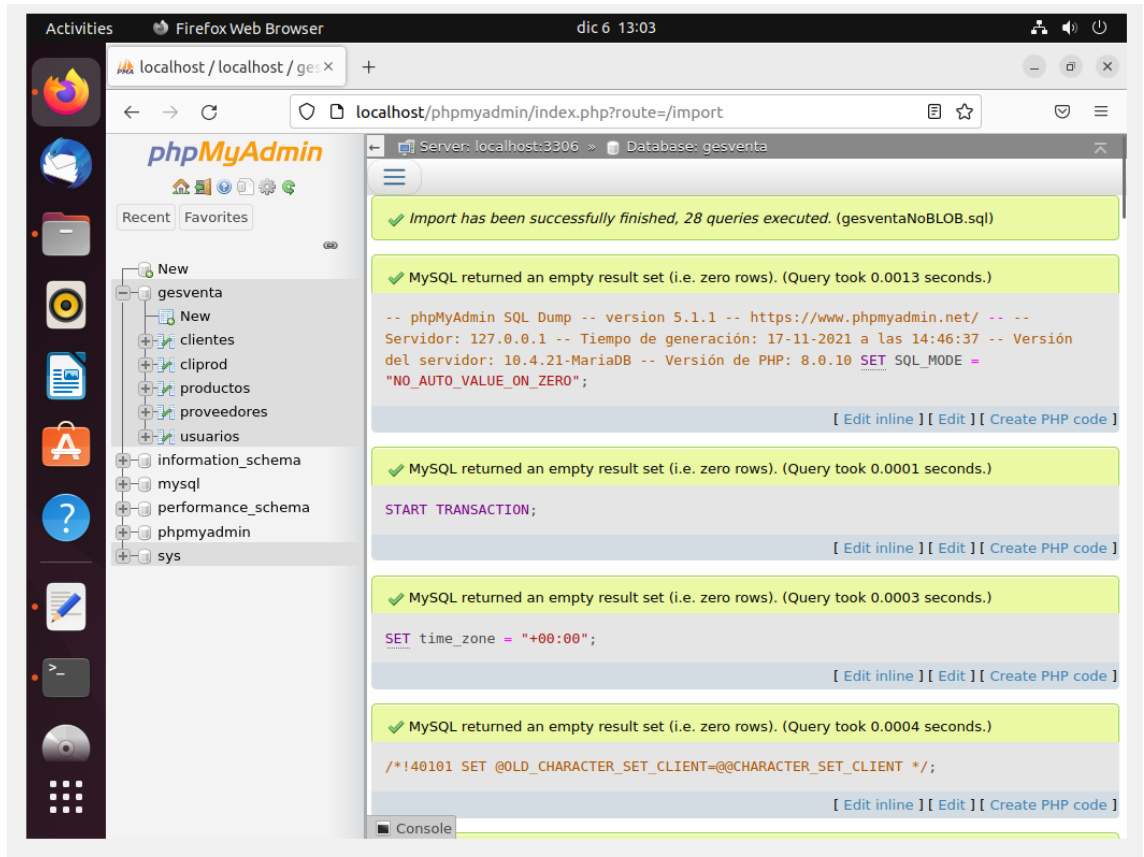
```
root@david-VirtualBox:/home/david# php -v
PHP 8.1.2-1ubuntu2.9 (cli) (built: Oct 19 2022 14:58:09) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.2, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.2-1ubuntu2.9, Copyright (c), by Zend Technologies
root@david-VirtualBox:/home/david#
```

Una vez instalado el resto del servidor de LAMP, si se ha hecho bien la instalación se podrá acceder a la dirección de localhost/phpmyadmin, y se observará una web como la siguiente



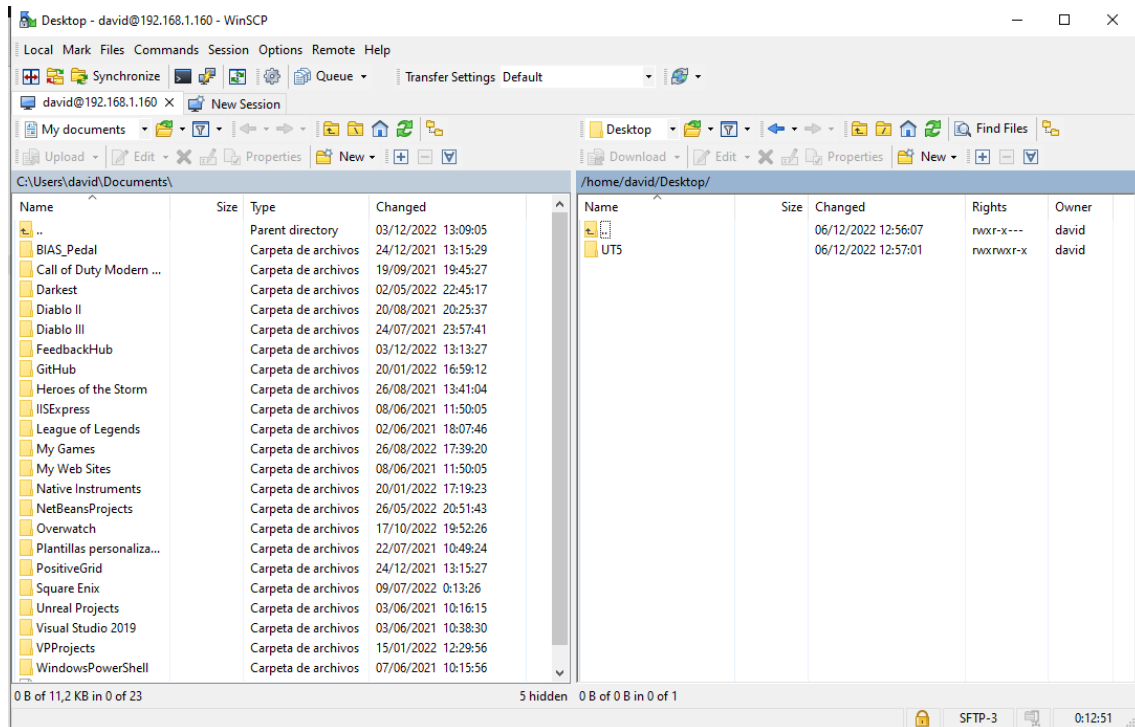
Creación de la base de datos

Siguiendo el guion dado en la práctica se creo un usuario nuevo para poder acceder al phpmyadmin diferente al root. En mi caso fue “david” el usuario y su contraseña es “1234”. Primero crearemos una base de datos que la llamaremos ‘gesventa’ de acuerdo con el guion. Luego importaremos la base de datos del ejercicio de la unidad 5 que se compone de 5 tablas.



Despliegue de la aplicación web

Tratando de seguir con la practica en cuestión sin una base de datos importada, se intentará hacer la conexión a través de WinSCP para poder hacer el paso de archivos entre la maquina virtual y el ordenador. A través de la ip, el usuario y la contraseña de acceso al ordenador, se realiza la conexión dando lugar a la siguiente imagen.



He pasado esa carpeta, la de la unidad 5 que tiene ya una base de datos y tiene también un ejercicio completo con Loguin incluido. Una vez importada la base de datos, se modificará el archivo de config-BD.php para poder acceder con el usuario creado



Acceso a la aplicación web

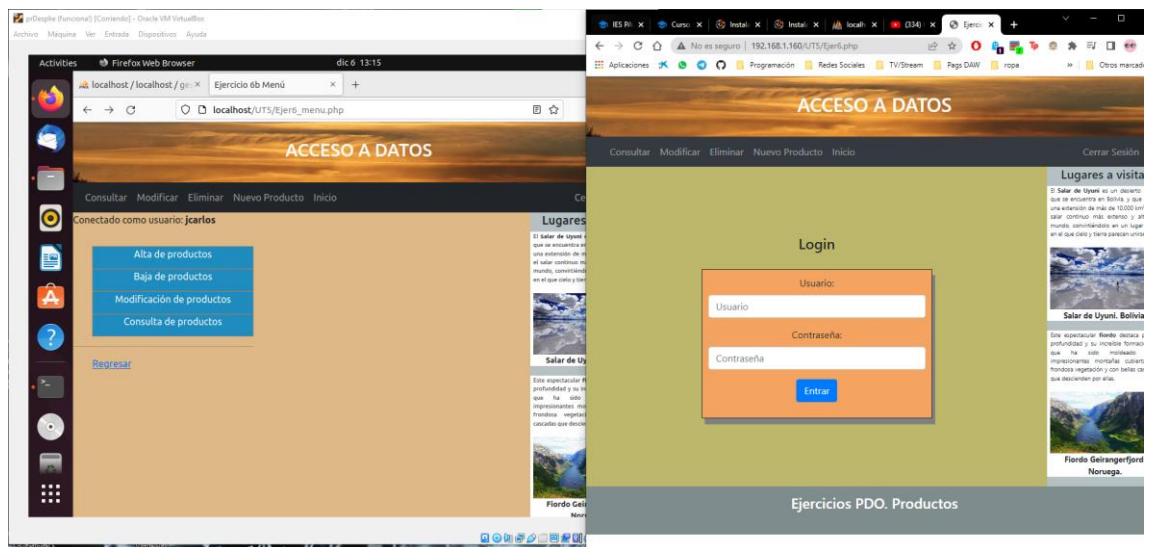
En este apartado se va a intentar acceder al servidor de localhost de la máquina virtual desde fuera de la máquina virtual. Lo primero a realizar es sacar la ip de nuestra máquina virtual para poder acceder a ella, aunque también podríamos acceder a través del nombre DNS del local host. En mi caso, como no he realizado los cambios en el archivo para cambiar y preseleccionar un nombre DNS accederé a través de la ip.

Primero ejecutare el comando de ifconfig en la consola recibiendo la siguiente información:

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.159 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::ec49:df9f:66b:2d83 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:2c:16:f8 txqueuelen 1000 (Ethernet)
    RX packets 327563 bytes 490115681 (490.1 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 25367 bytes 2148527 (2.1 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Bucle local)
    RX packets 2647 bytes 1761437 (1.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2647 bytes 1761437 (1.7 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Después se procederá a escribir la ip en la barra de búsqueda del navegador junto con la ruta del archivo:

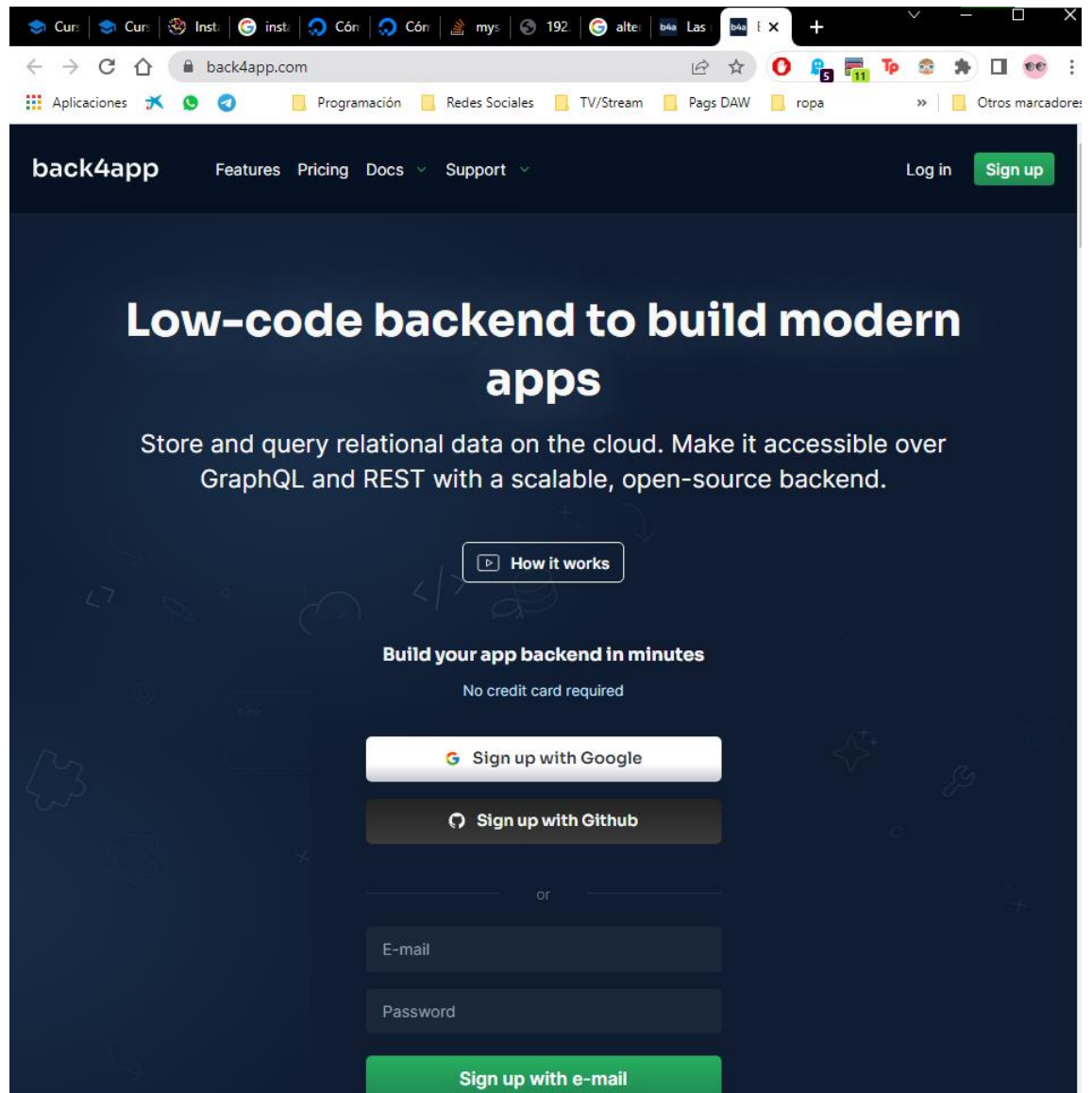


Despliegue en un servidor de hosting

Dado a que uno de los servidores de hosting dados es Heroku, y este deja de ser gratuito, se han barajado diferentes opciones para poder realizar los hostings.

Esas opciones han sido:

- Back4app



- Elastic Beanstalk

aws

Contacte con nosotros Support Español Mi cuenta Iniciar sesión [Cree una cuenta AWS](#)

re:Invent Productos Soluciones Precios Documentación Aprender Red de socios AWS Marketplace Ha > Q

AWS Elastic Beanstalk

[Información general](#) Características Precios Introducción Recursos Preguntas frecuentes

Socios

« Herramientas para desarrolladores

AWS Elastic Beanstalk

Despliegue y amplíe las aplicaciones web

[Introducción a AWS Elastic Beanstalk](#)

Cargue e implemente aplicaciones web de manera sencilla y

Céntrese en escribir código en lugar de aprovisionar y administrar

Seleccione y conserve el control completo de los recursos óptimos de

Utilice configuraciones ajustables para escalar su aplicación

Seleccione sus preferencias de cookies

Utilizamos cookies y herramientas similares para mejorar su experiencia, ofrecer nuestros servicios, hacerle llegar publicidad relevante y realizar mejoras. Algunos terceros aprobados también utilizan estas herramientas para ayudarnos a hacerle llegar publicidad y a proporcionar determinadas características del sitio.

[Personalizar](#) [Aceptar todas](#)

Casos de uso

Inicie de manera rápida aplicaciones web

Despliegue aplicaciones web escalables en minutos sin la complejidad de aprovisionar y administrar la infraestructura subyacente.

Cree backends de API móviles para sus aplicaciones

Utilice su lenguaje de programación favorito para crear backends API móviles y Elastic Beanstalk administrará los parches y las actualizaciones.

Redefina la plataforma de las aplicaciones empresariales críticas

Migre las aplicaciones con control de estado fuera de la infraestructura heredada a Elastic Beanstalk y conéctese de forma segura a su red privada.

- Google App Engine

cloud.google.com/appengine/

Google Cloud Información General Soluciones > Language Consola D

App Engine Contactar Empezar gratis

Aviso: En los próximos meses, reorganizaremos el sitio de documentación de App Engine para facilitar la búsqueda de contenido y alinearlo mejor con el resto de los productos de Google Cloud. El mismo contenido estará disponible, pero la navegación ahora coincidirá con el resto de los productos de Cloud. Si tienes comentarios o preguntas mientras navegas por el sitio, haz clic en Enviar comentarios.

No te pierdas los últimos anuncios de Google Cloud Next. [Ver ahora](#)

IR A

App Engine

Desarrolla sitios web monolíticos que se rendericen en servidores. App Engine es compatible con lenguajes de desarrollo conocidos y ofrece una gran variedad de herramientas para desarrolladores.

Los nuevos clientes reciben 300 USD en crédito gratis para utilizarlo en App Engine. Todos los clientes obtienen 28 instancias en un entorno estándar al día de forma gratuita (es decir, sin que se les descuente de su crédito).

[Probar App Engine gratis](#) [Contactar con Ventas](#)

✓ [Despliega un sitio web con App Engine siguiendo esta guía de inicio a fin.](#)

https://cloud.withgoogle.com/next?utm_source=cgc-site&utm_medium=et&utm_campaign=FY22-Q3-cgc-next22&utm_content=cgc-next22-banner&utm_term=-

VIDEO
App Engine en un minuto
1:12

CARACTERÍSTICAS PRINCIPALES

Características principales

Lenguajes de programación populares

Crea tu aplicación en Node.js, Java, Ruby, C#, Go, Python o PHP.

Totalmente gestionado

El hecho de utilizar un entorno totalmente gestionado te permite centrarte en el código mientras App Engine se encarga de gestionar la infraestructura.

- Dokku

github.com/dokku/dokku

Sign up

dokku / dokku Public

Sponsor Notifications Fork 1.8k Star 24k

<> Code Issues 39 Pull requests 5 Discussions Actions Projects 1 Wiki

master

Go to file Code

About

A docker-powered PaaS that helps you build and manage the lifecycle of applications

dokku.com

heroku docker kubernetes devops containers dokku paas buildpack nomad

Readme MIT license Code of conduct 24k stars 360 watching 1.8k forks

Releases 230

v0.28.4 Latest on Oct 28 + 229 releases

Sponsor this project

dokku Dokku 0

opencollective.com/dokku

File	Commit Message	Time Ago
josegonzalez	Merge pull request #5506 from dokku/...	13 hours ago
.circleci	tests: fix lint issues in yaml files	2 years ago
.devcontainer	feat: add fileutils plugin to the devcontainer	11 days ago
.github	chore(deps): bump hadolint/hadolint-action ...	24 days ago
contrib	fix: drop the app argument when calling stor...	19 days ago
debian	fix: add moby-compose as alternative for do...	29 days ago
docker	refactor: run crontab under sudo to support ...	9 months ago
docs	refactor: use a single implementation for che...	3 days ago
plugins	fix: add support for escaped plus (+) signs in...	14 hours ago
tests	fix: add support for escaped plus (+) signs in...	14 hours ago
.codacy.yml	tests: ignore duplication warnings in codacy	17 months ago
.dockerignore	docs: add code for building docs in ci	3 months ago
.editorconfig	chore: properly format go.mod files	17 months ago
.gitignore	feat: ensure related go source and vscode pl...	13 months ago
.shellcheckrc	feat: add .shellcheckrc	9 months ago
.stickler.yml	chore: remove all deprecated commands	3 years ago
CONTRIBUTING.md	docs: use updated url for gliderlabs slack inv...	3 months ago
Dockerfile	fix: install official docker package from dock...	29 days ago

What is Dokku?

The smallest PaaS implementation you've ever seen. Docker powered mini-Heroku in around 200 lines of Bash

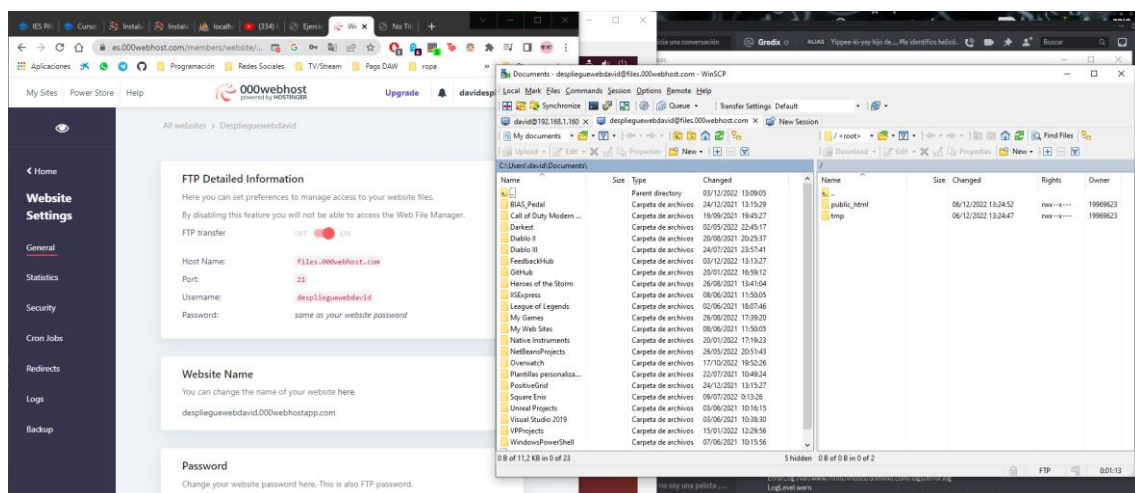
Creación de una cuenta en el servidor de alojamiento,
deploy de archivos de la app y acceso a la web

A continuación, se procederá a crear una cuenta en www.000webhost.com para poder luego alojar la app.




The image shows the registration page of 000webhost. At the top is the logo with the text "000webhost POWERED BY HOSTINGER". Below it is a large heading "Regístrate". The form consists of three input fields: "Correo electrónico" (Email) with the value "davidespinoso1993@gmail.com", "Contraseña" (Password) with masked dots, and "Repite la contraseña" (Repeat password) also with masked dots. A red button labeled "REGÍSTRATE" is at the bottom.

Una vez registrado, se acceden a las opciones y ahí se puede ver la información en general, esa información se usa en el WinSCP y como resultado obtendremos lo siguiente



Después de haber pasado los archivos, creamos una base de datos en el gestor de la propia web.



 000webhost

Create new database ✕

Database name

Database username

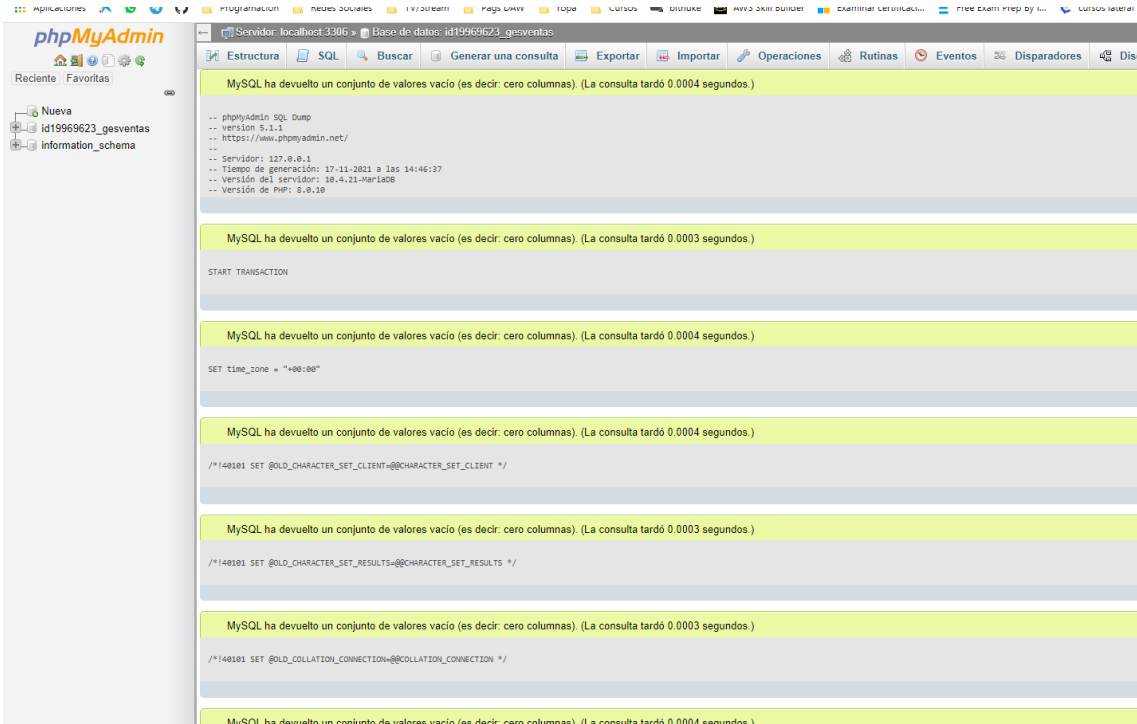
Password

 Hide Password
Randomize


Create

DB Name	DB User	DB Host	
id19969623_gesventas	id19969623_ventas	localhost	Manage ▾

Ahora le importamos la base de datos a través del phpmyadmin, y obtenemos lo siguiente:



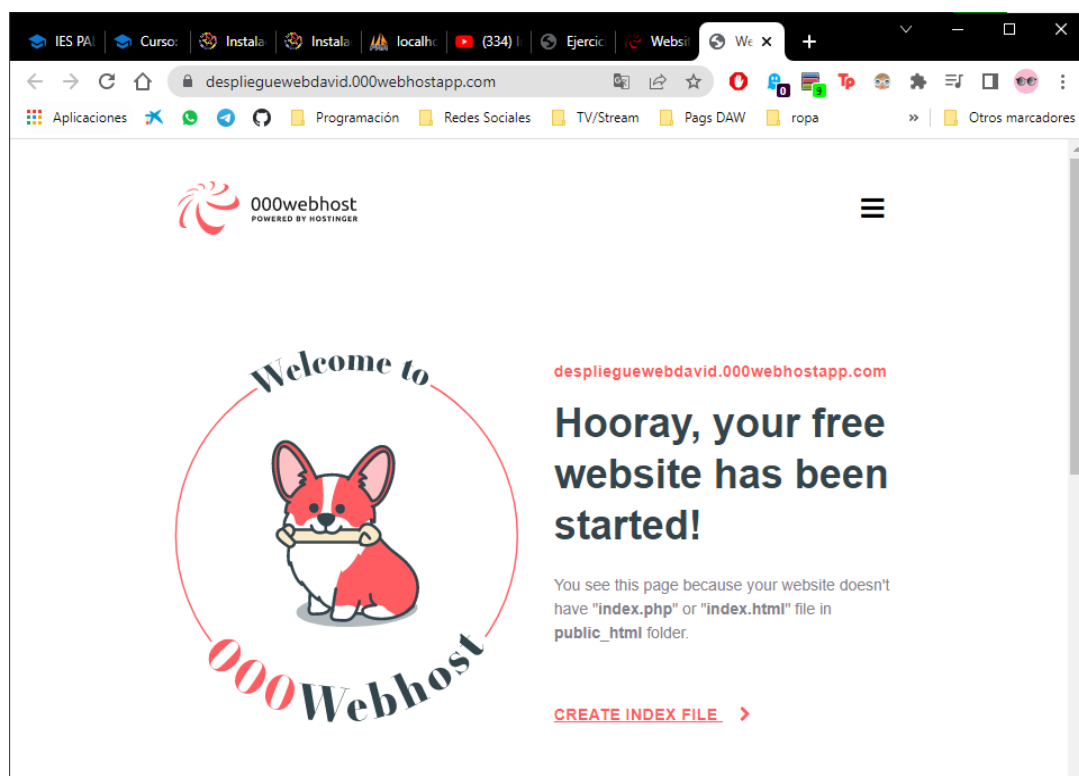
The screenshot shows the phpMyAdmin web interface. On the left sidebar, there's a tree view with 'Nueva' and 'id19969623_gesventas' (under 'information_schema'). The main panel shows the 'SQL' tab selected. The top navigation bar includes links like 'Estructura', 'SQL', 'Buscar', 'Generar una consulta', 'Exportar', 'Importar', 'Operaciones', 'Rutinas', 'Eventos', 'Disparadores', and 'Dis'. The main content area displays a series of status messages from MySQL, indicating that the import process is complete and successful. The messages are: 'MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0.0004 segundos.)', 'MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0.0003 segundos.)', 'MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0.0004 segundos.)', 'MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0.0004 segundos.)', 'MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0.0003 segundos.)', 'MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0.0003 segundos.)', and 'MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0.0004 segundos.)'.

Por último se modificará el archivo configdb para tenerlo con el nombre de la base de datos, el usuario y la pass de la web

Edit file

/public_html/UT5/config_BD.php

```
1 <?php
2
3 // Parámetros acceso a una BD
4 define('HOST', 'localhost');
5 define('BD', 'id19969623_gesventas');
6 define('USER', 'id19969623_ventas');
7 define('PASSWORD', 'Defender123456789.');
8 define('CHARSET', 'utf8');
9
10 ?>
11 |
12
```



Por último, añadiendo a la url de la web la ruta de la carpeta y el archivo en cuestión, llegamos a la web deseada: <https://desplieguewebdavid.000webhostapp.com/UT5/Ejer6.php>

Warning: session_start(): Cannot start session when headers already sent in /storage/ssd5/623/19969623/public_html/UT5/Ejer6.php on line 3

ACCESO A DATOS

Consultar Modificar Eliminar Nuevo Producto Inicio Cerrar Sesión

Login


Usuario:

Contraseña:

Entrar


Lugares a visitar

El Salar de Uyuni es un desierto de sal que se encuentra en Bolivia y que ocupa una extensión de más de 10.000 km². Es el salar continuo más extenso y alto del mundo convirtiéndolo en un lugar único en el que cielo y tierra parecen unirse.



Salar de Uyuni, Bolivia.

Este espectacular fiordo destaca por su profundidad y su increíble formación, ya que ha sido modelado entre impresionantes montañas cubiertas de frondosa vegetación y con ríos caudales que descienden por ellas.



Fiordo Geirangerfjord, Noruega.

Ejercicios PDO. Productos

Despliegue de una aplicación web Python en Ubuntu

Para realizar este paso se ha seguido un tutorial de cómo hacer un deploy de una aplicación flask en Ubuntu.

Lo primero a realizar es instalar apache2, que como ya lo tenemos instalado con antelación de la práctica no será necesario. A continuación, hay que instalar Python3, y python3-pip. Una vez tenemos instalados estos dos programas, hay que instalar una librería de apache que es: libapache2-mod-wsgi-py3.

```
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
david@david-VirtualBox:~$ sudo apt install libapache2-mod-wsgi-py3  
Reading package lists... Done
```

Después, hay que instalar flask, esto lo haremos a través de otro comando sudo:

```
david@david-VirtualBox:~$ sudo -H pip3 install flask  
Collecting flask  
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)  
    101.5/101.5 KB 2.0 MB/s eta 0:00:00  
Collecting Jinja2>=3.0  
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)  
    133.1/133.1 KB 18.5 MB/s eta 0:00:00  
Collecting itsdangerous>=2.0  
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
```

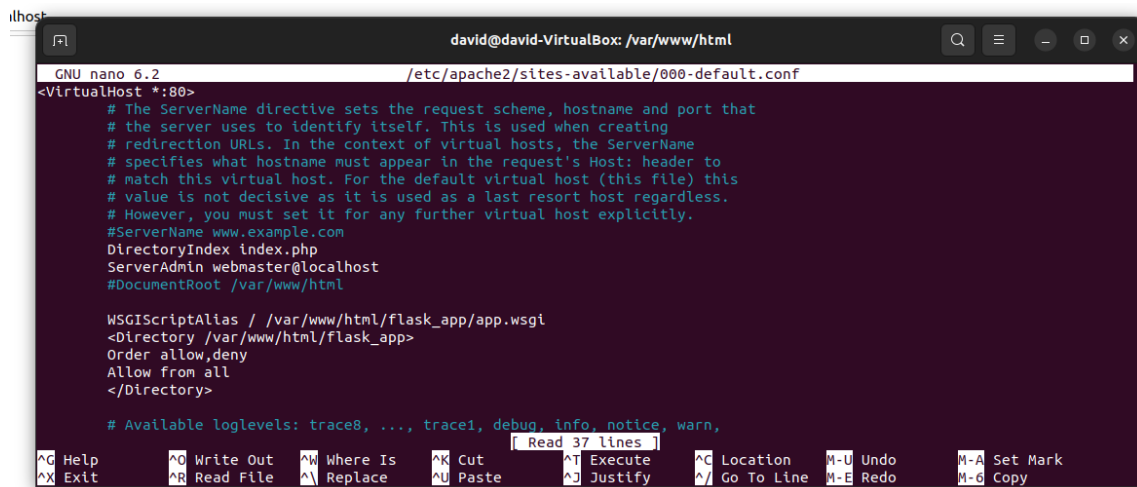
A continuación, se crearan un par de carpetas en el directorio home para poder modificar los archivos antes de subirlos a la carpeta de destino. Los archivos que se crean son: un directorio flask_app, y dos archivos, uno .py y otro .wsgi

```
david@david-VirtualBox:~/proyectos$ mkdir flask_app  
david@david-VirtualBox:~/proyectos$ cd flask_app/  
david@david-VirtualBox:~/proyectos/flask_app$ nano myapp.py  
david@david-VirtualBox:~/proyectos/flask_app$ cat myapp.py  
from flask import Flask  
app = Flask(__name__)  
  
@app.route('/')  
def app():  
    return "Aplicacion flask en ubuntu"  
  
if __name__ == '__main__':  
    app.run()
```

```
david@david-VirtualBox:/var/www/html$ cat flask_app/app.wsgi  
import sys  
  
sys.path.insert(0, '/var/www/html/flask_app')  
  
from myapp import app as application
```

A continuación se moverá la carpeta que contiene esos dos archivos, que es la carpeta flask_app, a la ruta que se muestra en app.wsgi, que es /var/www/html

Una vez que esta en ese directorio, hay que modificar el archivo de 000-default.conf de apache y se dejara de la siguiente manera:

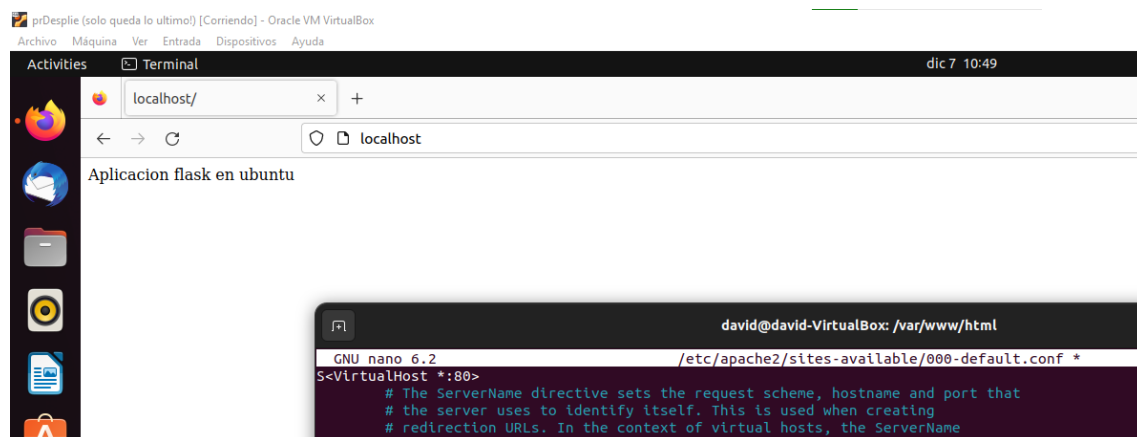


```
GNU nano 6.2 /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com
DirectoryIndex index.php
ServerAdmin webmaster@localhost
#DocumentRoot /var/www/html

WSGIScriptAlias / /var/www/html/flask_app/app.wsgi
<Directory /var/www/html/flask_app>
Order allow,deny
Allow from all
</Directory>

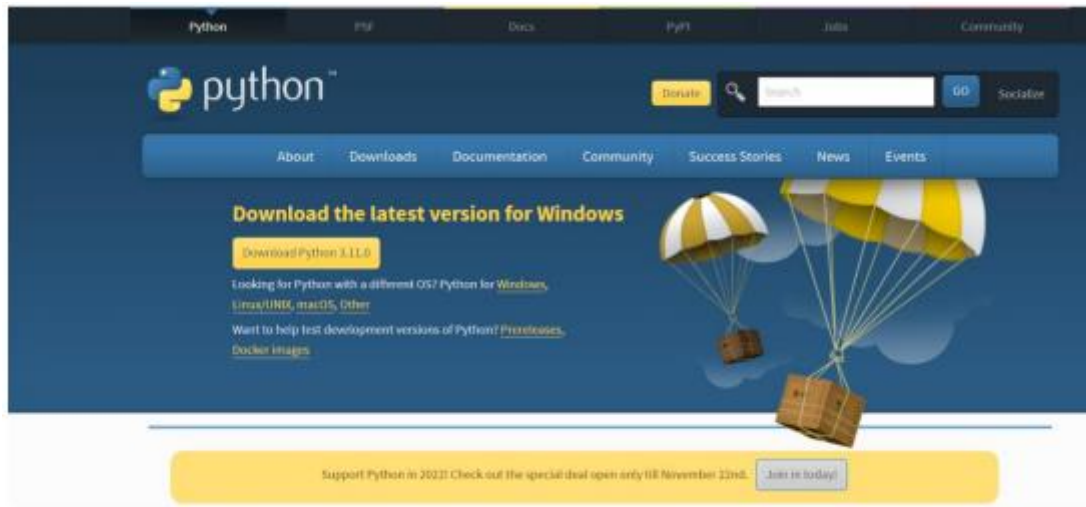
# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
```

Después de esto, hay que reiniciar el servidor de apache2, y cuando introducimos localhost en la barra de búsqueda del navegador nuestra aplicación deberá de funcionar:

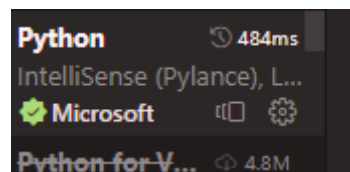


Despliegue de una aplicación web Python en Heroku

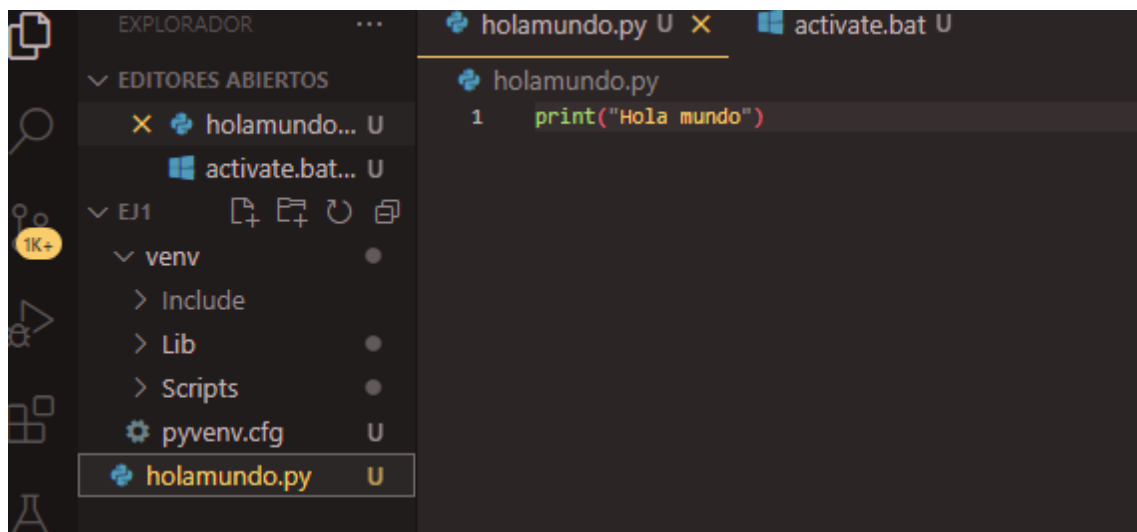
Instalación de Python:



Instalación de la extensión en visual studio code:



Primero se crea un archivo con la terminación .py. Luego pondremos echo ("hola mundo") y cuando ejecutamos el termina se printea

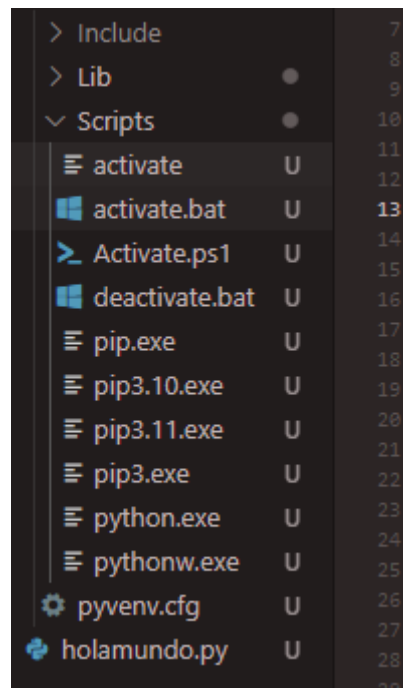


```
PS C:\Users\Admin\Desktop\gitDaw\2 Daw\Python\Ej1> & 'C:\Users\Admin\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.python-2022.18.2\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '63951' '--' 'c:\Users\Admin\Desktop\gitDaw\2 Daw\Python\Ej1\holamundo.py'
Hola mundo
```

Ejecutar en ese mismo terminal :

Py -m venv c:/desktop/gitdaw/2Daw/EJ1

Se crea una carpeta en el directorio donde se pueden ver varios subdirectorios:



Ejecutamos el archivo active.bat, y nos sale un prompt:

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw\Python\Ej1>
```

Copiamos el código del ejercicio:

```
holamundo.py 1, U X activate.bat U Ex ▶
holamundo.py > ...
1  # Importar la biblioteca Flask del entorno flask
2  from flask import Flask
3  # Inicializar Flask obteniendo un objeto para manejar
4  app = Flask(__name__)
5  # Crear rutas utilizando decoradores
6  @app.route('/')
7  def inicio():
8      return 'Primera aplicación Web con Python y flask'
9  # Hacemos que aplicación se mantenga a la escucha de
10 # comprobando primero si estamos en el archivo principal
11 if __name__ == '__main__':
12     app.run()
```

Y luego ejecutamos el comando **pip freeze** en el terminal. Dara un error asi que se tiene que instalar con el comando **pip install flask**

```
\Python\Ej1> pip install flask
Collecting flask
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
    |#####| 101.5/10... 2.9 MB/s eta 0:00:00
Collecting Werkzeug>=2.2.2
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
    |#####| 232.7/23... 2.9 MB/s eta 0:00:00
Collecting Jinja2>=3.0
```

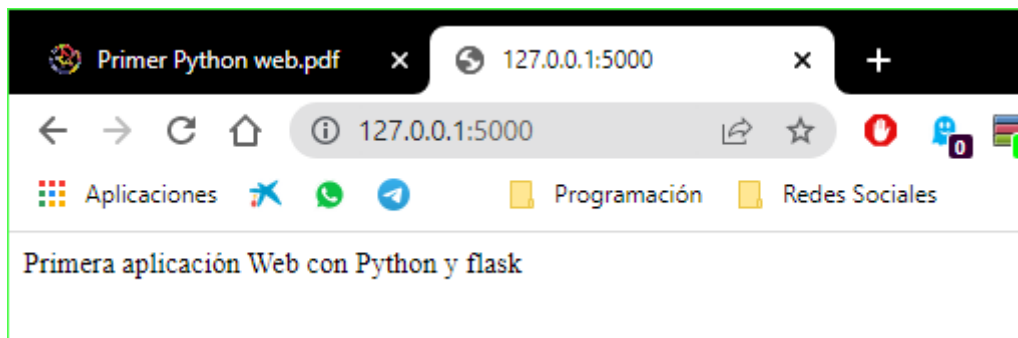
Luego se ejecuta de nuevo el comando **pip freeze** y esta vez obtenemos el siguiente resultado:

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1> pip freeze
click==8.1.3
colorama==0.4.6
Flask==2.2.2
itsdangerous==2.1.2
Jinja2==3.1.2
MarkupSafe==2.1.1
Werkzeug==2.2.2
```

Ejecutamos py holamundo.py

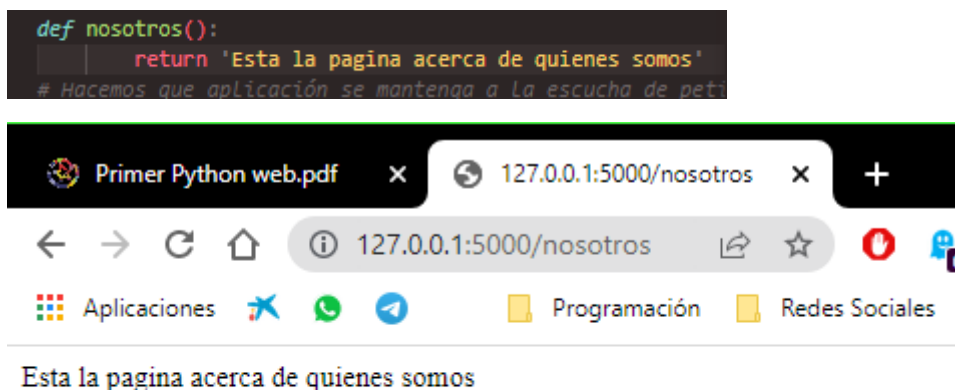
```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1> py holamundo.py
* Serving Flask app 'holamundo'
* Debug mode: off
WARNING: This is a development server. Do not
use it in a production deployment. Use a pro
duction WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Cuando accedemos a esa url comprobamos que esta cargando la app bien:



- Creando más rutas:

Ruta nosotros:



Primero creamos el nuevo directorio, y lo llamaremos “templates”

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1> mkdir templates

Directorio:
C:\Users\Admin\Desktop\gitDaw\2
Daw\Python\Ej1

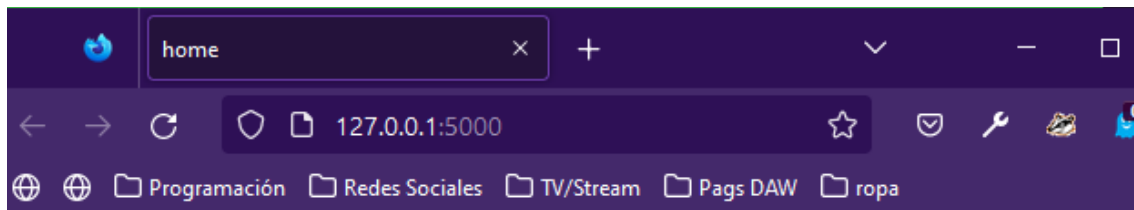
Mode                LastWriteTime         Length
----                -
d-----          23/11/2022         15:06

(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1> 
```

Luego modificamos el archivo holamundo.py a app.py, y le cambiamos tambien un par de líneas tal y como viene en la guía.

```
# Importar la biblioteca Flask del entorno flask
from flask import Flask, render_template
# Inicializar Flask obteniendo un objeto para manejar la aplicación
app = Flask(__name__)
# Crear rutas utilizando decoradores
@app.route('/')
def inicio():
    return render_template('home.html')
```

Esto produce que ahora lo que se cargue es el html que hemos creado:



Web principal home.html es renderizada en Python

Primero se crean la carpeta de static, y dentro de ella la de css.

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1> mkdir static/css

Directorio:
C:\Users\Admin\Desktop\gitDaw\2
Daw\Python\Ej1\static

Mode                LastWriteTime         Length
----                -
d-----          23/11/2022         15:20

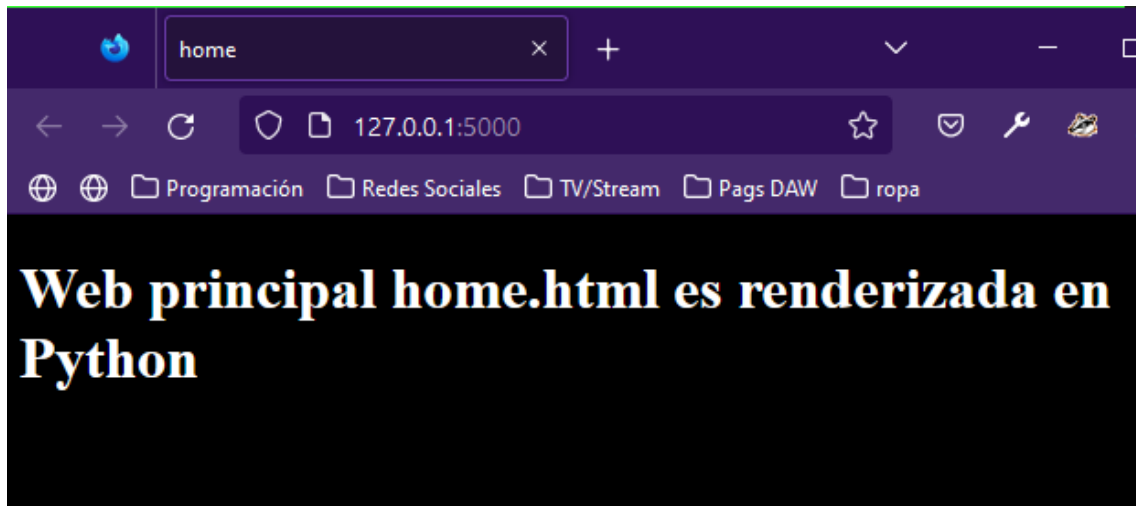
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1> 
```

Dentro se crea el archivo css.css y se pone un fondo negro con la letra blanca.

Después se linkea al archivo html a través de una url.

```
<link rel="stylesheet" href="{{url_for('static',
filename='css.css')}}">
<meta name="viewport" content="width=device-width,
```

Y comprobamos que funciona:



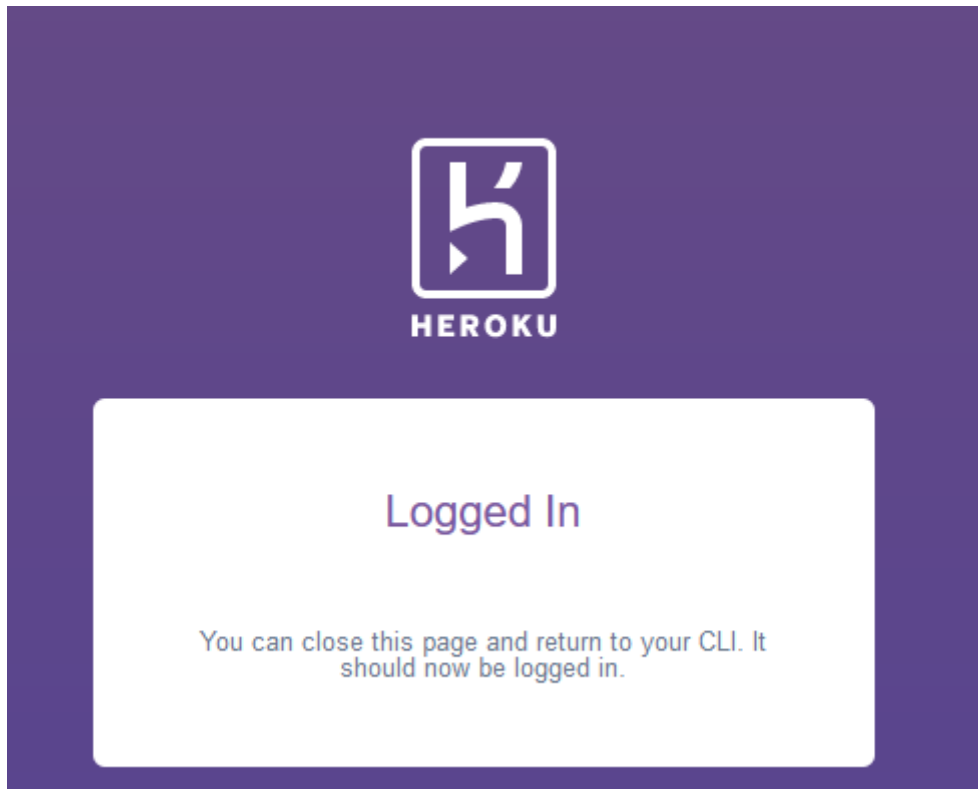
Solo tendremos que añadir dentro del `app.run()` del archivo `app.py` un `debug=true`. Despues reiniciamos el servidor y comprobamos que esta funcionando correctamente

```
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 925-977-586
```

Se modifiko un poco la web para hacerla mas atractiva

Despues nos creamos la cuenta de heroku, le pondremos una contraseña a través del link que hemos recibido al correo.

Seguimos los pasos descritos en el guion (ejecución de Loguin de heroku en cmd) y llegaremos a esta pantalla:

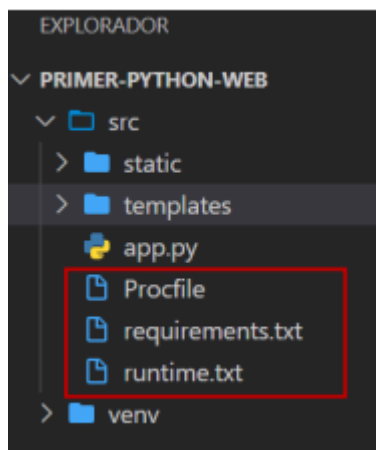


Luego instalaremos con el comando pipi instal gunicorn

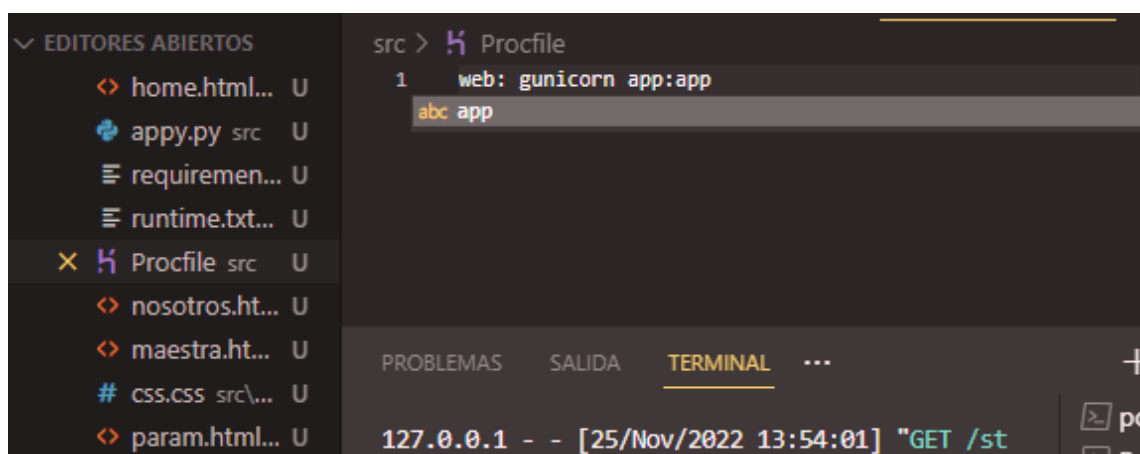
```
\Python\Ej1> pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-20.1.0-py3-none-any.whl (79 kB)
    ━━━━━━━━━━━ 79.5/79.5 2.2 MB/s eta 0:00:00
Requirement already satisfied: setuptools>=3.0 in c:\users\admin\desktop\gitdaw\2 daw\python\ej1\venv\lib\site-packages (from gunicorn) (65.5.0)
Installing collected packages: gunicorn
Successfully installed gunicorn-20.1.0

[notice] A new release of pip available: 22.3 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw \Python\Ej1> █
```

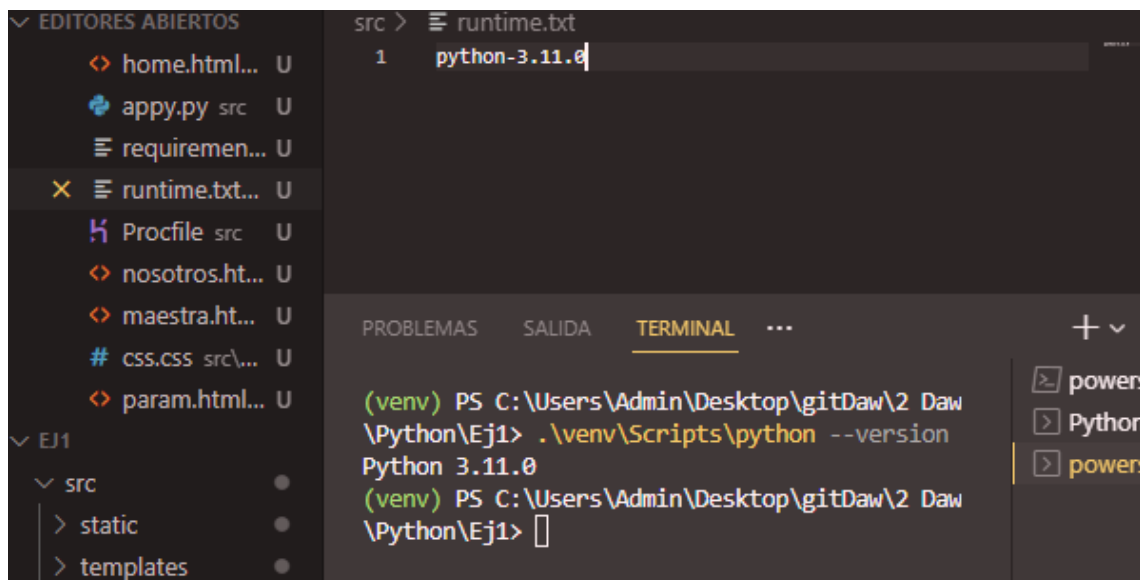
Después creamos los archivos requirements.txt, runtime.txt y procfile.



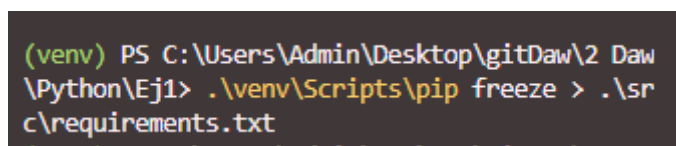
Luego modificamos el Procfile a:



Modificamos el archivo runtime.txt



Luego se ejecuta la lista de comandos en la terminal:



Y como resultado nos dará en el requirements.txt lo siguiente:

```
src > requirements.txt
1  autopep8==2.0.0
2  click==8.1.3
3  colorama==0.4.6
4  Flask==2.2.2
5  gunicorn==20.1.0
6  itsdangerous==2.1.2
7  Jinja2==3.1.2
8  MarkupSafe==2.1.1
9  pycodestyle==2.10.0
10 tomli==2.0.1
11 Werkzeug==2.2.2
12
```

Creamos un repositorio Git y lo iniciamos

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1> cd src
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1\src> git init
Initialized empty Git repository in C:/Users/
Admin/Desktop/gitDaw/2 Daw/Python/Ej1/src/.gi
t/
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1\src>
```

Luego hacemos un git add . y un git status y recibimos por consola lo siguiente:

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1\src> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage
  )
        new file:   Procfile
        new file:   appy.py
        new file:   requirements.txt
        new file:   runtime.txt
        new file:   static/css.css
        new file:   static/css/css.css
        new file:   templates/home.html
        new file:   templates/maestra.html
        new file:   templates/nosotros.html
        new file:   templates/param.html
```

Y luego realizamos la subida:

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1\src> git commit -m "primera subida
a a heroku"
[master (root-commit) f2b5e44] primera subida
a a heroku
10 files changed, 138 insertions(+)
create mode 100644 Procfile
create mode 100644 appy.py
create mode 100644 requirements.txt
create mode 100644 runtime.txt
create mode 100644 static/css.css
create mode 100644 static/css/css.css
create mode 100644 templates/home.html
create mode 100644 templates/maestra.html
create mode 100644 templates/nosotros.html
create mode 100644 templates/param.html
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1\src>
```

Luego creamos el nombre de la aplicación:

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1\src> git config --global user.ema
il "davidespinosa1993@gmail.com"
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1\src> git config --global user.nam
e "david"
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
```

Después de configurar con el mail, y con el nombre, se sincronizara mediante el comando heroku git:remote nombreakcion

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1\src> heroku git:remote dawpalomer
asdavid
set git remote heroku to https://git.heroku.c
om/dawpalomerasdavid.git
```

Despues se desplegara la aplicación con el comando git push heroku rama_git, que en este caso al ser master será así: git push heroku master

```
(venv) PS C:\Users\Admin\Desktop\gitDaw\2 Daw
\Python\Ej1\src> git push heroku master
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (15/15), 3.07 KiB | 628
.00 KiB/s, done.
Total 15 (delta 1), reused 0 (delta 0), pack-
reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-22 stac
k
remote: -----> Determining which buildpack to
use for this app
remote: -----> Python app detected
remote: -----> Using Python version specified
in runtime.txt
remote: -----> Installing python-3.11.0
remote: -----> Installing pip 22.3.1, setupto
ols 63.4.3 and wheel 0.37.1
remote: -----> Installing SQLite3
remote: -----> Installing requirements with p
ip
remote:          Collecting autopep8==2.0.0
remote:          Downloading autopep8-2.0.0-p
y2.py3-none-any.whl (45 kB)
```

Al final del todo, se podrá ver la url donde se ha desplegado la app, en nuestro caso es:

```
remote: -----> Launching...
remote:          Released v3
remote:          https://dawalomerasdavid.hero
kuapp.com/ deployed to Heroku
remote:
remote: Starting November 28th, 2022, free He
roku Dynos, free Heroku Postgres, and free He
```

Finalmente para guardar los cambios que se han realizado se hará con los comandos git add .
 NOTA IMPORTANTE, EL PUNTO AL FINAL ES IMPORTANTE

Para poder ir viendo los cambios se van a realizar se podrá con el comando git status

Para poder lanzar la app se realizara con el comando heroku open, y este nos podrá automáticamente la url del siguiente modo:

<https://dawalomerasdavid.herokuapp.com>