

# Examen Programación

Carlota Martín-Anero Mateos

21 diciembre 2022

## 1 Ejercicio 1

### 1.1 Apartado A

Realiza un fork del repositorio de GitHub: <https://github.com/rubences/TutorialBasico>

Hacer un fork de un repositorio consiste en obtener una copia exacta del repositorio original en tu cuenta. Podrás hacer cambios en tu fork sin afectar al repositorio original. Para hacerlo tenemos que ir al repositorio a copiar y hacer clic en el botón *Fork*. Github nos creará un fork del repositorio en nuestra cuenta

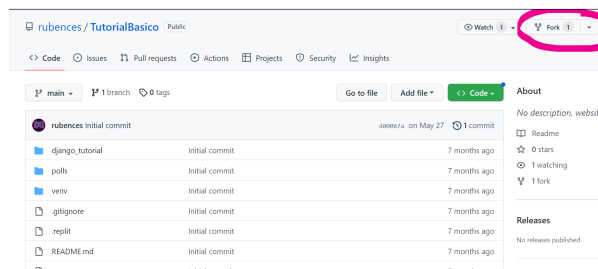


Figure 1: Hacer un fork

### 1.2 Apartado B

*Crea un entorno virtual de python3 e instala las dependencias necesarias para que funcione el proyecto.*

Lo primero que he hecho, es eliminar el anterior entorno virtual que se llamaba *venv*. Después, he seguido los siguientes pasos:

1. Abre una consola o terminal y asegúrate de estar en el directorio donde quieres crear el entorno virtual.

```
python -m venv nombre_del_entorno
```

En este caso hemos nombrado como el anterior *venv*. Nos aparecerá una nueva carpeta con el nombre del entorno virtual y un conjunto de archivos necesarios para que el entorno funcione.:

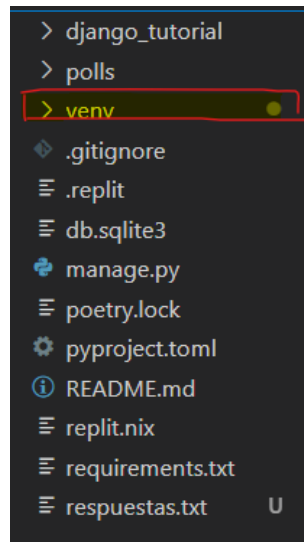


Figure 2: Entorno virtual

2. Para activar el entorno virtual, ejecuta el siguiente comando en la terminal:

```
source venv/bin/activate
```

Una vez lo activemos, obtendremos esto

```
(venv) PS C:\Users\Carlota Martin-Anero\OneDrive\Documentos\UAX\Tercero de carrera  
GMAT\5º cuatri\Desarrollo orientado a objetos\Examen 3\Despliegue-y-CMS-de-Aplica  
ciones-en-Python>
```

Figure 3: Activado el entorno virtual

3. Una vez que hayas activado el entorno virtual, puedes instalar las dependencias necesarias para que el proyecto funcione. Primero, asegúrate de tener el archivo *requirements.txt* en el directorio raíz del proyecto. Este archivo debería contener una lista de todas las dependencias que necesitas instalar. Para instalar las dependencias, ejecuta el siguiente comando:

```
pip install -r requirements.txt
```

Nos crea el el directorio principal, el siguiente archivo:

### 1.3 Apartado C

*Comprueba que vamos a trabajar con una base de datos sqlite. ¿Qué fichero tienes que consultar?*

Para comprobar que estamos trabajando con una base de datos sqlite, debemos consultar el archivo "settings.py" del proyecto. Este archivo contiene la configuración de la base de datos y nos indicará qué tipo de base de datos estamos utilizando. La base de datos predeterminada de Django es sqlite3, aun así, lo comprobamos:

*/djangotutorial/settings.py*

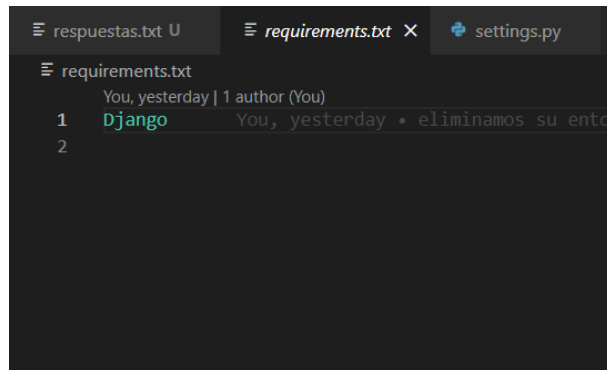


Figure 4: requirements.txt

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': BASE_DIR / 'db.sqlite3',  
    }  
}
```

¿Cómo se llama la base de datos que vamos a crear? En este caso, la base de datos se llama *default*. Esto se puede ver en la clave *default* del diccionario *DATABASES*. La base de datos se almacena en un archivo llamado *db.sqlite3* que se encuentra en el mismo directorio que la variable *BASEDIR*.

Django utiliza el motor de base de datos *"django.db.backends.sqlite3"* para conectarse a la base de datos *default*, que es una base de datos *SQLite*.

## 1.4 Apartado D

*Crea la base de datos. A partir del modelo de datos se crean las tablas de la base de datos*

1. Accedemos al directorio del proyecto con el siguiente comando en la terminal:

```
cd Despliegue-y-CMS-de-Aplicaciones-en-Python
```

2. Creamos una aplicación de Django con el siguiente comando:

```
python manage.py startapp polls
```

3. Ahora tenemos que definir los modelos de datos en el archivo *models.py* de la aplicación, crear las vistas en el archivo *views.py* y las plantillas HTML en la carpeta *templates*, y configurar las URLs en el archivo *urls.py*. En este caso, como hemos hecho un fork del repositorio, estos *templates*, así como la base de datos y las tablas necesarias para almacenar los datos de la aplicación ya los tenemos. Ahora, necesitamos configurar la conexión a la base de datos en el archivo *settings.py* del proyecto. Para ello añadimos: */djangotutorial/urls.py*

```
urlpatterns = [
    path('polls/', include('polls.urls')), #NEW
    path('admin/', admin.site.urls),
    path("", views.index, name="frontpage"),
]
```

*/djangotutorial/settings.py*

```
INSTALLED_APPS = [
    'polls.apps.PollsConfig', #NEW
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

4. Luego, podemos usar el siguiente comando para crear las tablas en la base de datos:

```
python manage.py makemigrations
python manage.py migrate
```

5. Una vez que hayamos creado las tablas en la base de datos, podemos probar la aplicación ejecutando el siguiente comando:

```
python manage.py runserver
```

## 1.5 Apartado E y F

*Crea un usuario administrador. Ejecuta el servidor web de desarrollo y entra en la zona de administración (/admin) para comprobar que los datos se han añadido correctamente.*

1. Abre una consola o terminal y accede al directorio del proyecto de Django. Ejecuta el siguiente comando para crear un usuario administrador:

```
python manage.py createsuperuser
```

2. Seguimos las instrucciones en pantalla para ingresar el nombre de usuario, correo electrónico y contraseña del usuario administrador. Nos aseguraremos de elegir una contraseña segura.

```
torial> python manage.py createsuperuser
>>
Username (leave blank to use 'carlotamartin-anero'):
Email address: cmartmat@myuax.com
Password:
Password (again):

Superuser created successfully.
```

Figure 5: createsuperuser

Una vez que hayas creado el usuario administrador, podrás acceder al panel de administración de Django con nuestro nombre de usuario y contraseña. Para ello, tras ejecutar nuestro proyecto (*python manage.py runserver*), la dirección para acceder a la zona de administración sería *http://127.0.0.1:8000/admin*.

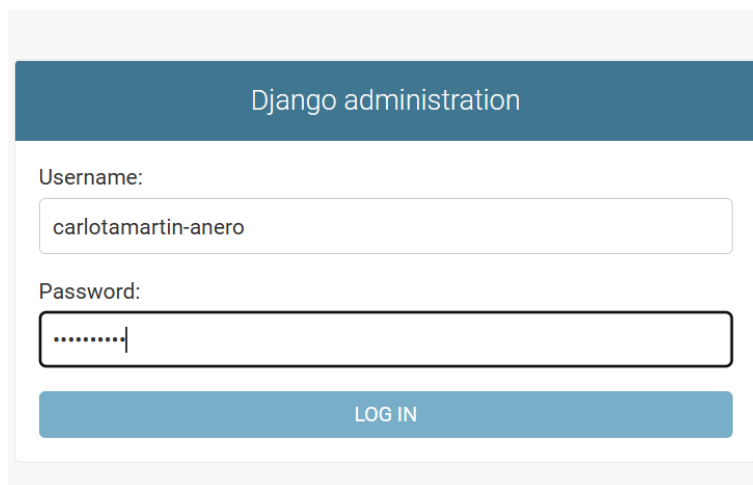


Figure 6: panel de administación

Desde el panel de administración, podrás gestionar usuarios, grupos, permisos y otras configuraciones del proyecto.

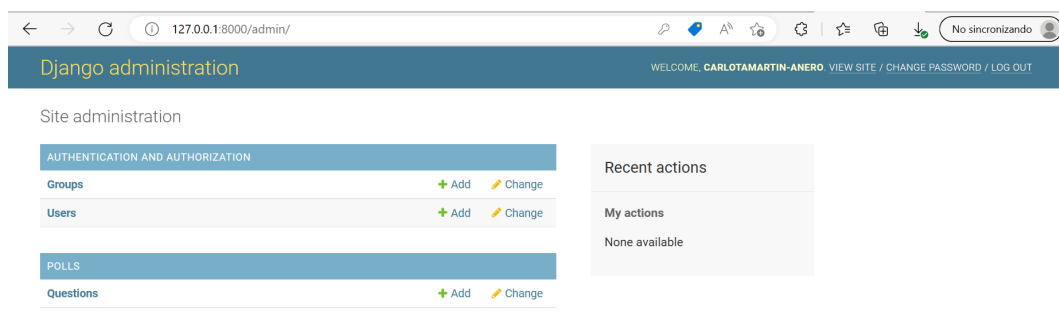
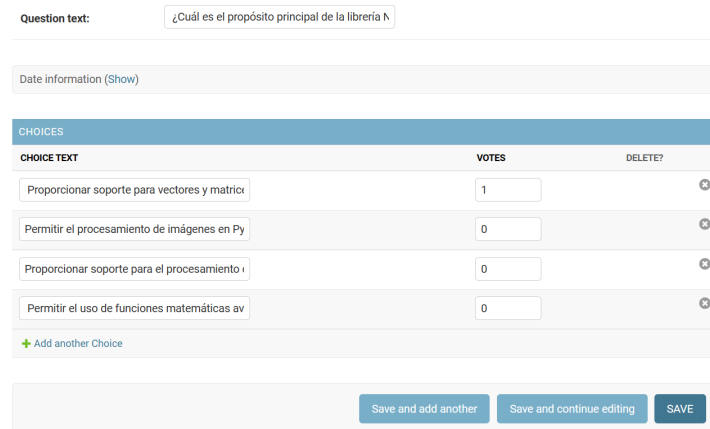


Figure 7: panel de administación

## 1.6 Apartado G

*Crea dos preguntas, con posibles respuestas.* Para crear preguntas, en la *figura 7*, se puede ver que en la parte inferior, hay una apartado de *Questions*, en el recuadro de *Polls*. Para añadir una nueva le damos a *Add*.

Añadimos la primera:



CHOICE TEXT	VOTES	DELETE?
Proporcionar soporte para vectores y matrices	1	X
Permitir el procesamiento de imágenes en Py	0	X
Proporcionar soporte para el procesamiento de lenguaje natural	0	X
Permitir el uso de funciones matemáticas avanzadas	0	X

+ Add another Choice

Save and add another Save and continue editing SAVE

Figure 8: Primera pregunta

La pregunta y la respuesta son:

Pregunta 1: ¿Cuál es el propósito principal de la librería NumPy en Python?

Posibles respuestas:

- Proporcionar soporte para vectores y matrices de números en Python
- Permitir el procesamiento de imágenes en Python
- Proporcionar soporte para el procesamiento de lenguaje natural en Python
- Permitir el uso de funciones matemáticas avanzadas en Python

Le damos a *Save and add another*

## 1.7 Apartado H

*Comprueba en el navegador que la aplicación está funcionando, accede a la url*

Question text:

---

Date information

**Date published:** Date:  Today   
Time:  Now   
Note: You are 1 hour ahead of server time.

---

CHOICE TEXT	VOTES	DELETE?
<input type="text" value="C++"/>	<input type="text" value="0"/>	
<input type="text" value="Python"/>	<input type="text" value="1"/>	
<input type="text" value="Java"/>	<input type="text" value="0"/>	
<input type="text" value="Assembly"/>	<input type="text" value="0"/>	

[+ Add another Choice](#)

Figure 9: Segunda pregunta

Figure 10: Segunda pregunta

## 2 Ejercicio 2

*Vamos a realizar el despliegue de nuestra aplicación en un entorno de producción, para ello vamos a utilizar nuestro VPS, sigue los siguientes pasos*

- Clona el repositorio en el VPS.
- Crea un entorno virtual e instala las dependencias de tu aplicación.
- Instala el módulo que permite que python trabaje con mysql: `(env) pip install mysqlclient`
- Crea una base de datos y un usuario en mysql.
- Configura la aplicación para trabajar con mysql, para ello modifica la configuración de la base de datos en el archivo settings.py:
- Como en la tarea 1, realiza la migración de la base de datos que creará la estructura de datos necesarias. Comprueba que se han creado la base de datos y las tablas.

Para poder desplegar nuestra aplicación en un entorno virtual, lo primero que hemos hecho es cambiar la base de datos SQLite3 a una base de datos MySQL con XAMPP. Hemos seguido los

siguientes pasos:

1. Primero tenemos que desactivar el entorno virtual, ya que con la nueva actualización, no te deja instalar nuevos módulos. Para desactivarlo: **venv-Scripts-desactivate**. Entonces, instalamos el módulo *mysqlclient*

```
pip install mysqlclient
```

2. Tras instalarlo, actualizamos los *requirements* y volvemos a activar el entorno. (Explicado como realizarlo en el ejercicio 1.

```
112 mysql==0.0.3
113 mysql-connector==2.2.9
114 mysql-connector-python==8.0.31
115 mysqlclient==2.1.1
116 nest-asyncio==1.5.1
117 netaddr==0.8.0
```

Figure 11: Instalado *mysqlclient* en los *requirements*

3. Ahora nos hemos intentado conectar a mysql desde la terminal, utilizando el siguiente comando pero ha sido imposible:

```
mysql -u root -p
```

La solución ha sido utilizar la app de *XAMPP*. Lo abrimos y activamos *SQL*

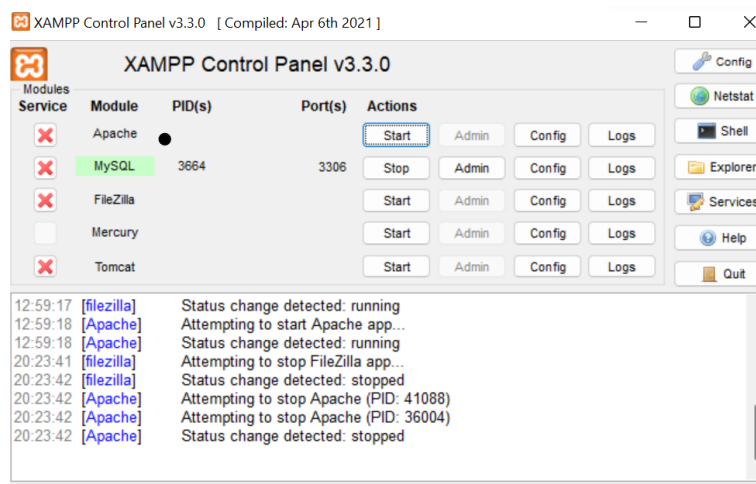


Figure 12: *MySQL* activado

4. Ahora creamos la base de datos a través de la terminal de *XAMPP*. Para ello introducimos los siguientes comandos:

```
mysql -u root
```



```

Carlota Martin-Anero@DESKTOP-14A57JC c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 55
Server version: 10.4.27-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.

```

Figure 13: Hemos entrado a MYSQL

```
CREATE DATABASE examen
```

```

MariaDB [(none)]> CREATE DATABASE examen;
Query OK, 1 row affected (0.002 sec)

```

Figure 14: Base de datos creada

Lo comprobamos en *PHPMYADMIN*



Figure 15: Base de datos creada: examen

- Ahora tenemos que introducir esta base de datos creada en nuestra app Django. Para ello vamos a *settings.py* y cambiamos la base de datos. No le hemos puesto contraseña y el puerto es el que nos da *XAMPP*.

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'examen',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}

```

- Ahora al ejecutar la app. Nos funciona, pero nos salta un aviso de que tenemos que hacer las migraciones. Estas migraciones significan crear archivos de migración para las tablas de

base de datos. Estos archivos describen los cambios que se deben hacer en la base de datos, como la creación o modificación de tablas, y se utilizan para actualizar la base de datos de acuerdo con los cambios en el modelo de la aplicación. Ejecutando en la línea de comandos:

```
python manage.py makemigrations
python manage.py migrate
```

7. Una vez hemos hecho las migraciones, si nos vamos a nuestra base de datos, vemos que se han hecho correctamente. Los comprobamos: Si ejecutamos la app, podemos ver que funciona

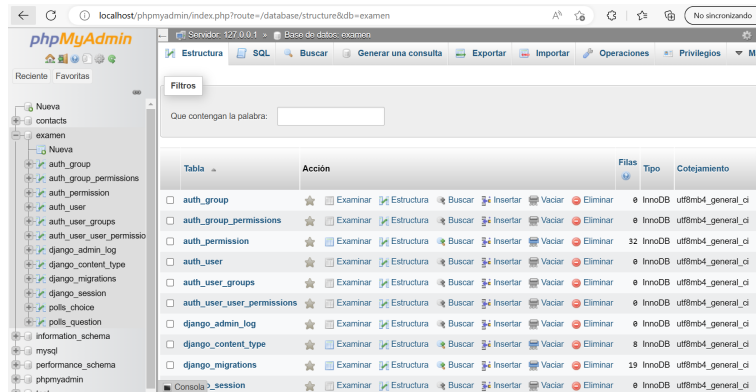


Figure 16: Migraciones hechas

perfectamente:

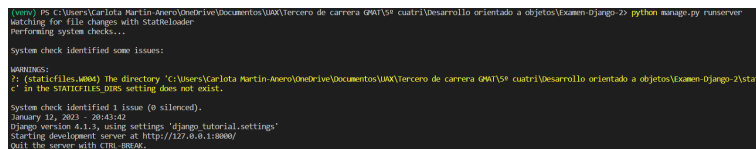


Figure 17: Prueba de la app

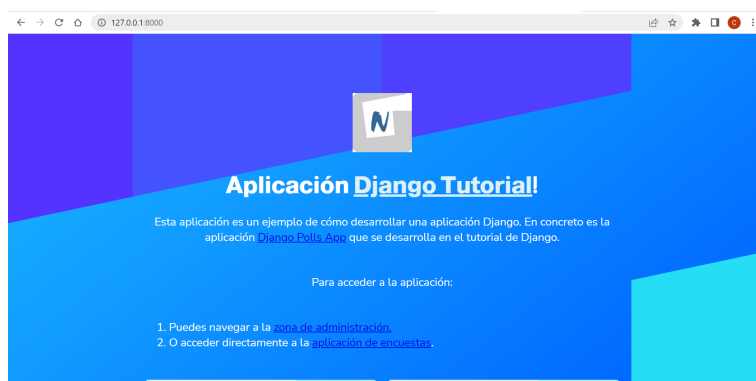


Figure 18: Prueba de la app con MySQL

8. Ahora el siguiente paso que queremos hacer es subirlo a Railway. Para ello, lo primero que tenemos que hacer es instalar el paquete *gunicorn*. (EL ENTORNO DESACTIVADO!!)

```
pip install gunicorn
```

9. Creamos el archivo *Procfile*, que se utilizará para conectar la app a railway.

```
web: python manage.py migrate && gunicorn django_tutorial.wsgi
```

10. Ahora lo que hacemos es irnos a nuestra cuenta de Railway. Allí, nos creamos una cuenta vinculada con github y empezamos un nuevo proyecto. Este proyecto lo vinculamos con nuestro repositorio. El problema llega cuando se va a conectar que tras intentarlo repetidas veces, no funciona:

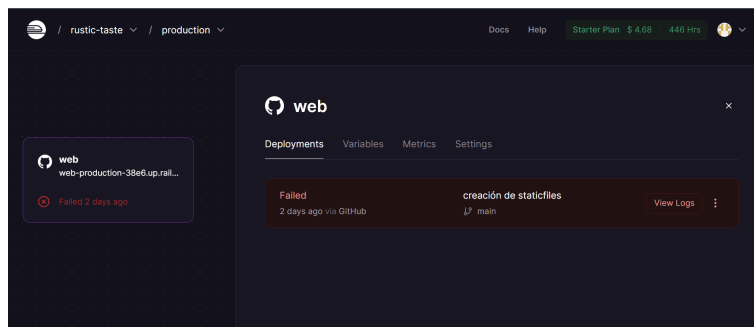


Figure 19: Prueba de la app con MySQL