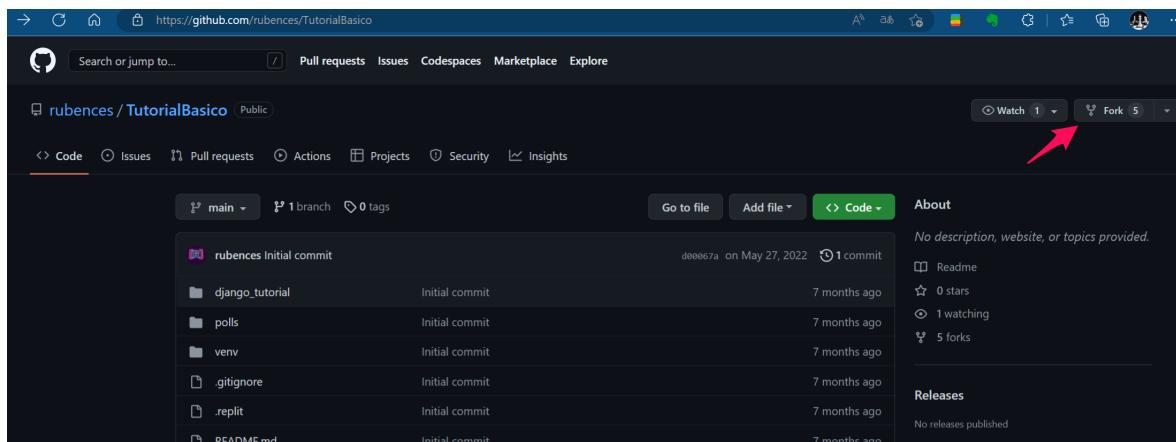


Despliegue y CMS

NOTA: Todo hecho sobre Windows

Tarea 1: Entorno de desarrollo

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



Una vez hecho el fork clonamos el repositorio en vscode para trabajar con el

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

Abrimos un terminal con la siguiente dirección C:\Users\... Path ...\\TutorialBasico
En nuestro caso no hace falta crear un entorno porque ya existe, solo tendríamos que activarlo, pero en caso de tener que crear uno introducimos el siguiente comando en la terminal:

```
>> py -m venv 'Nombre del entorno'
```

Para activarlo:

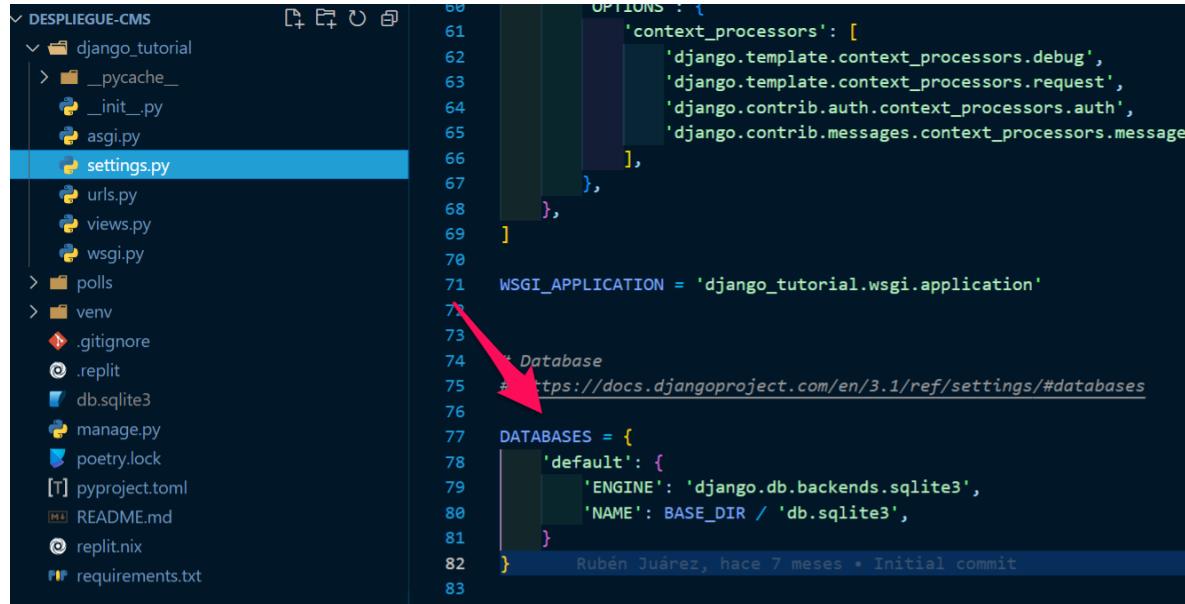
```
>> .\env\Scripts\activate
```

Ahora solo nos queda instalar las dependencias, introducimos los siguientes comandos:

```
>> py manage.py makemigrations  
>> py manage.py migrate
```

- Comprueba que vamos a trabajar con una base de datos sqlite.
¿Qué fichero tienes que consultar?. ¿Cómo se llama la base de datos que vamos a crear?

Consultamos el archivo settings.py



```

DESPLIEGUE-CMS
  |- django_tutorial
    |- __pycache__
    |- __init__.py
    |- asgi.py
    |__ settings.py
    |- urls.py
    |- views.py
    |- wsgi.py
  |- polls
  |- venv
  |- .gitignore
  |- .replit
  |- db.sqlite3
  |- manage.py
  |- poetry.lock
  |- pyproject.toml
  |- README.md
  |- repl.it.nix
  |- requirements.txt

```

```

OPTIONS : [
    'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages'
    ],
],
WSGI_APPLICATION = 'django_tutorial.wsgi.application'

# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

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

```

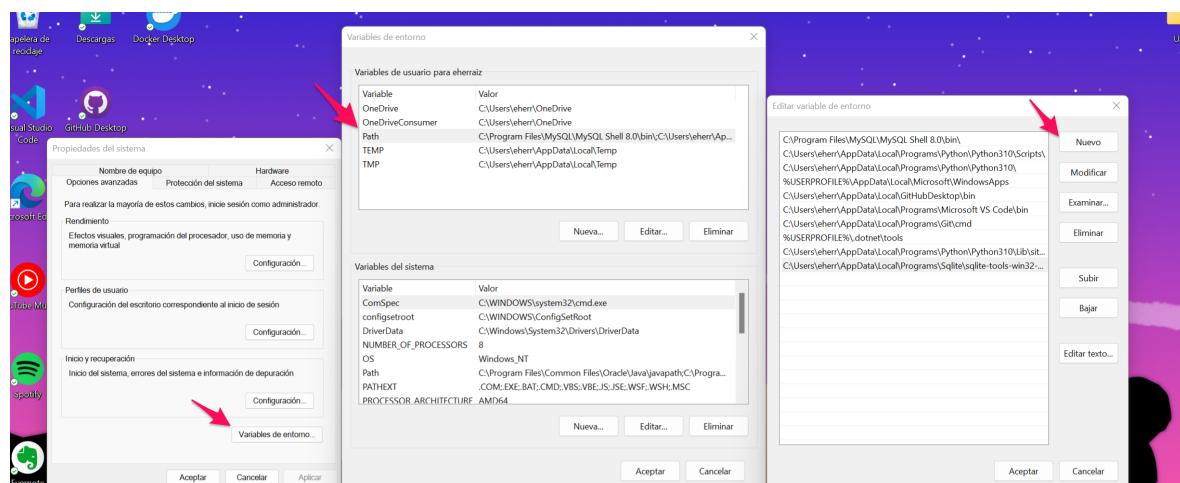
Rubén Juárez, hace 7 meses • Initial commit

Vemos que utilizamos Sqlite3, utilizada por defecto en Django

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

A través de este enlace instalamos Sqlite3 <https://www.sqlite.org/download.html> y lo añadimos a las variables de sistema para poder trabajar con el.

Nos metemos en la configuración avanzada del sistema



Añadimos la ruta donde tenemos Sqlite3 y aceptamos todo

Volviendo al terminal, lo reiniciamos e introducimos el siguiente comando:

```
>> sqlite3 db.sqlite3  
>> .tables
```

De esta manera podremos trabajar con tablas en la base de datos, con .help vemos una ayuda con opciones que podemos hacer

- Crea un usuario administrador.

```
>> python manage.py createsuperuser
```

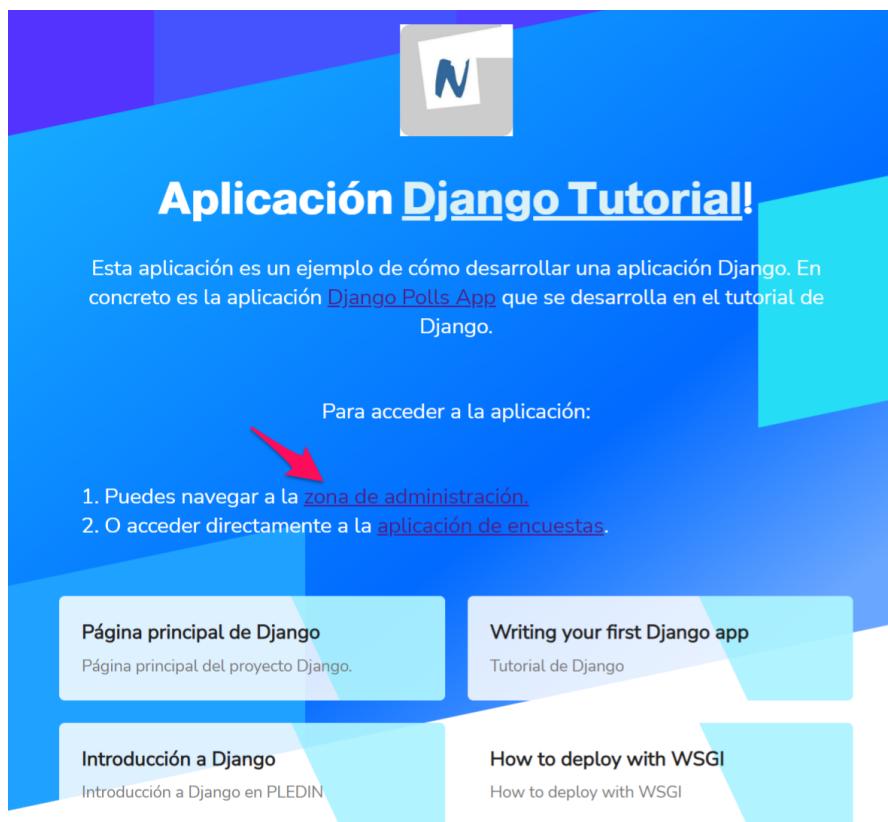
He introducidos los datos que se nos piden

- Ejecuta el servidor web de desarrollo y entra en la zona de administración (`\admin`) para comprobar que los datos se han añadido correctamente.

```
>> python manage.py runserver
```

Abrimos la siguiente dirección

```
System check identified no issues (0 silenced).  
January 05, 2023 - 12:45:21  
Django version 4.1.4, using settings 'django_tutorial.settings'  
Starting development server at http://127.0.0.1:8000/ ↗  
Quit the server with CTRL-BREAK.  
[05/Jan/2023 12:45:23] "GET / HTTP/1.1" 200 5864
```



Observamos como se ha creado

Action:	Go	0 of 1 selected
<input type="checkbox"/>	USERNAME	EMAIL ADDRESS
<input type="checkbox"/>	eherraiz	quiquemh2001@gmail.com

- Crea dos preguntas, con posibles respuestas.

The screenshot shows the Django administration interface at the URL `127.0.0.1:8000/admin/polls/questions/`. The top navigation bar includes links for Home, Polls, and Questions. Below this, there is a search bar labeled "Start typing to filter...". The main content area is divided into sections: "AUTHENTICATION AND AUTHORIZATION" (Groups, Users) and "POLLSS" (Questions). The "Questions" section contains a list of poll questions and a "+ Add" button. A pink arrow points to this "+ Add" button. The detailed view for one question, "¿Tortilla con o sin cebolla?", is shown below, featuring fields for question text, choice creation, and save options.

Introducimos los datos necesarios en las casillas marcadas

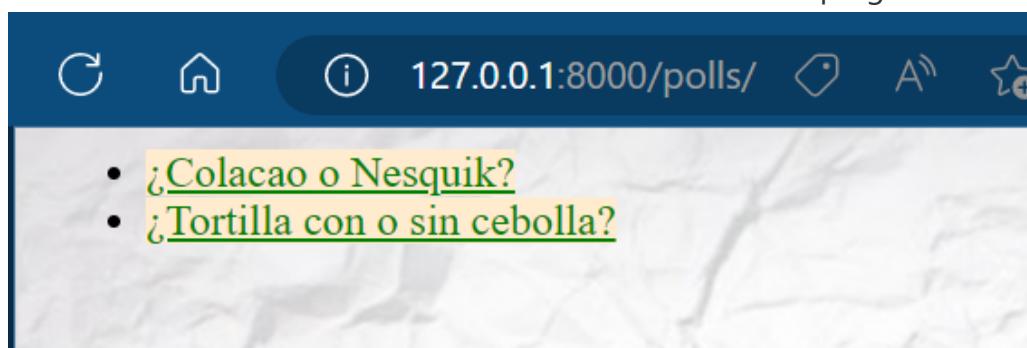
This screenshot shows the "Create a new question" form. The question text is set to "¿Colacao o Nesquik?". Two choices are listed: "Colacao" and "Nesquik". Each choice has a text input field and a vote counter. The "DELETE?" column contains checkboxes. At the bottom, there are buttons for "Delete", "Save and add another", "Save and continue editing", and a large blue "SAVE" button. Three pink arrows highlight the question text field, the "Colacao" choice input field, and the "Nesquik" choice input field.

- Comprueba en el navegador que la aplicación está funcionando, accede a la url `\polls`.

Volvemos a la pagina inicial <http://127.0.0.1:8000/>

The screenshot shows the homepage of the "Aplicación Django Tutorial!". At the top, there's a blue header with a white 'N' logo. Below it, the title "Aplicación Django Tutorial!" is displayed in large, bold, blue text. A subtext explains that it's an example of developing a Django application, specifically the "Django Polls App" from the Django tutorial. A section titled "Para acceder a la aplicación:" lists two options: "1. Puedes navegar a la [zona de administración](#)." and "2. O acceder directamente a la [aplicación de encuestas](#)." A red arrow points to the second option. Below this, there are four cards: "Página principal de Django" (with a link to "127.0.0.1:8000/"), "Writing your first Django app" (with a link to "127.0.0.1:8000/tutorial/"), "Introducción a Django" (with a link to "127.0.0.1:8000/intro/"), and "How to deploy with WSGI" (with a link to "127.0.0.1:8000/deploy/").

Observamos como se han añadido correctamente las dos preguntas



Tarea 2: Entorno de producción

- Clona el repositorio en el VPS.

En este caso vamos a utilizar Railway para gestionar nuestro servidor en la nube.

Creamos un nuevo proyecto y lo enlazamos con nuestro Github

Para que nos cargue correctamente tenemos que ajustar los requirements, donde las librerías que necesita railway para subirlo a la nube.

```

requirements.txt
You, hace 16 minutos | 2 a
1 asgiref
2 Django
3 gunicorn
4 psycopg2
5 sqlparse
6
7

```

Además le tenemos que permitir el acceso al host, donde con un asterisco se lo permitimos a cualquiera

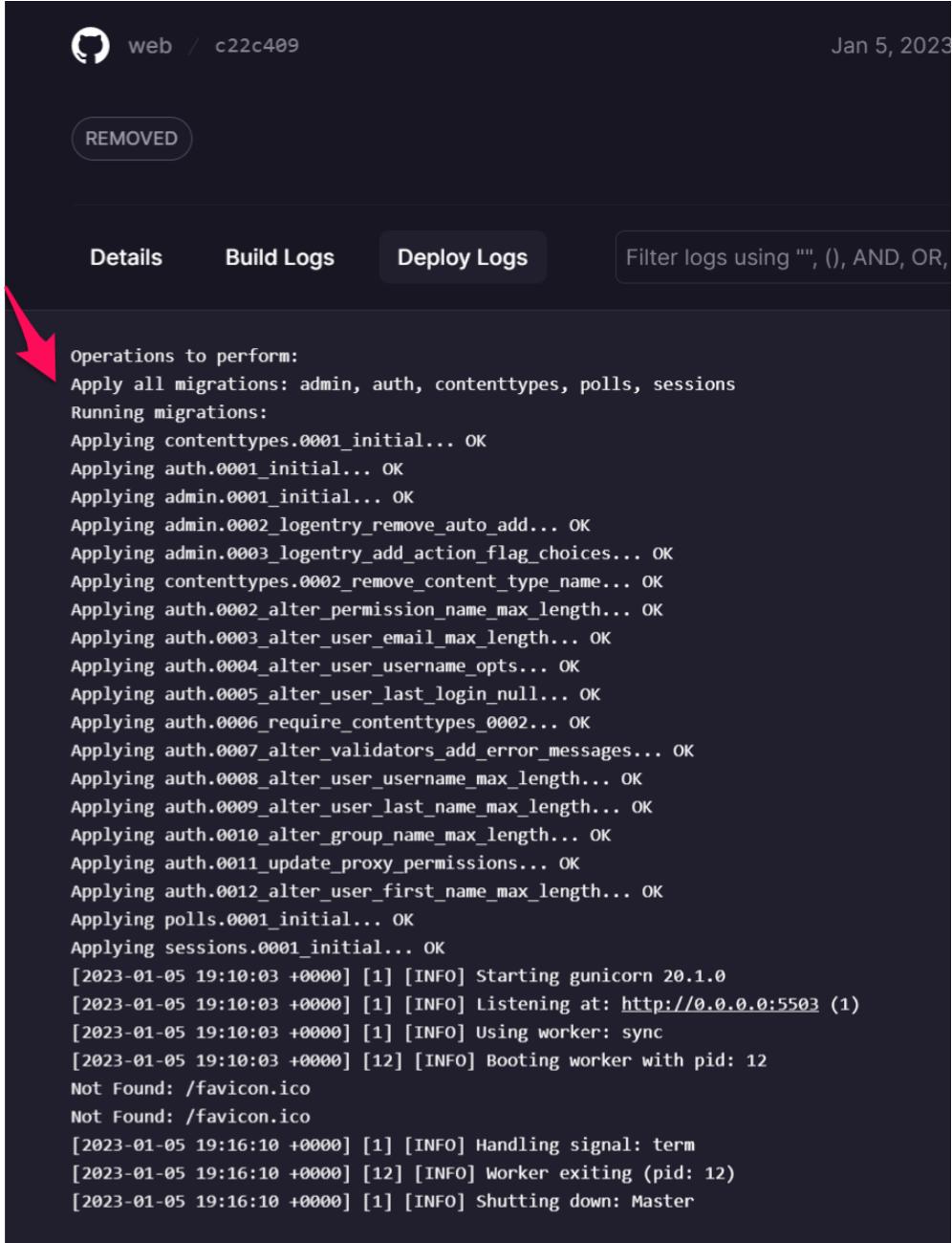
```

settings.py
...
20 # See https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = '9f0h)gozf$g%6igo8&767w1xro0adm+)msx'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = ['*']
29
30 # Application definition
31
32 INSTALLED_APPS = [
33     'polls.apps.PollsConfig',
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40 ]
41
42 MIDDLEWARE = [
43     'django.middleware.security.SecurityMiddleware',
44     ...
45 ]

```

- Crea un entorno virtual e instala las dependencias de tu aplicación.

Gracias a que hemos utilizado railway nos ahorraremos este paso



The screenshot shows a Railway deployment interface. At the top, there's a logo, the word "web", a slash, and the ID "c22c409". To the right is the date "Jan 5, 2023". Below this, a red arrow points down to the "Deploy Logs" tab, which is currently selected. To the left of the logs, a button labeled "REMOVED" is visible. On the far right, there's a search/filter bar with the placeholder "Filter logs using '', (), AND, OR,". The logs themselves begin with "Operations to perform:" followed by a list of migrations being applied, each marked as "OK". This includes migrations for admin, auth, contenttypes, polls, and sessions. After the migrations, the logs show the startup of gunicorn, including listening on port 5503, using a sync worker, and booting a worker with pid 12. It also handles a signal for term and shows the master shutting down.

```
Operations to perform:
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying polls.0001_initial... OK
  Applying sessions.0001_initial... OK
[2023-01-05 19:10:03 +0000] [1] [INFO] Starting gunicorn 20.1.0
[2023-01-05 19:10:03 +0000] [1] [INFO] Listening at: http://0.0.0.0:5503 (1)
[2023-01-05 19:10:03 +0000] [1] [INFO] Using worker: sync
[2023-01-05 19:10:03 +0000] [12] [INFO] Booting worker with pid: 12
Not Found: /favicon.ico
Not Found: /favicon.ico
[2023-01-05 19:16:10 +0000] [1] [INFO] Handling signal: term
[2023-01-05 19:16:10 +0000] [12] [INFO] Worker exiting (pid: 12)
[2023-01-05 19:16:10 +0000] [1] [INFO] Shutting down: Master
```

- Instala el módulo que permite que python trabaje con mysql:

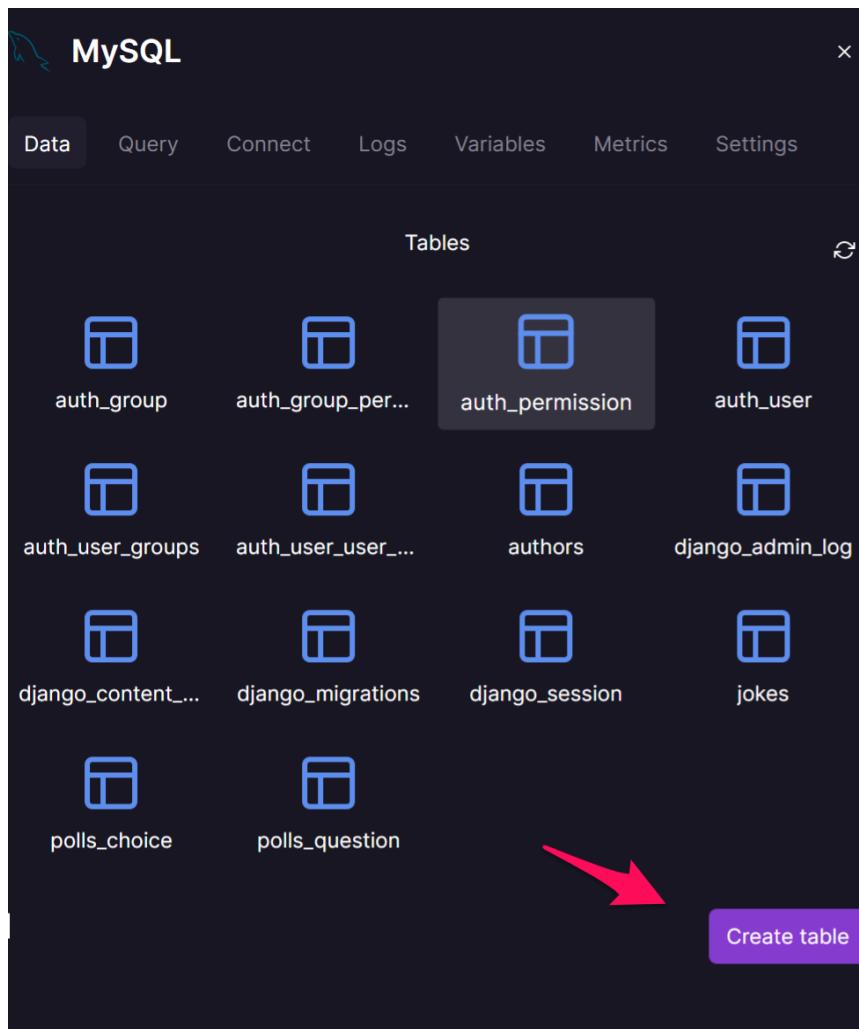
En la terminal lo haríamos mediante el siguiente comando:

>> pip install mysqlclient

Nos vale con incluirlo en los requirements.txt

- Crea una base de datos y un usuario en mysql.

Creamos la base en railway y ya podemos trabajar por ella y crear tablas



Authors y Jokes es un ejemplo que nos puede poner railway

- Configura la aplicación para trabajar con mysql, para ello modifica la configuración de la base de datos en el archivo [settings.py](#):

Copiamos los datos que se nos generaron en la tabla de datos

The screenshot shows the MySQL Workbench interface. On the left, the 'Variables' tab displays environment variables:

- MYSQLDATABASE: railway
- MYSQLHOST: containers-us-west-98.railway.app
- MYSQLPASSWORD: ar849qIyZnNtthKPLXMyd
- MYSQLPORT: 6088
- MYSQLUSER: root
- MySQL_URL: mysql://\${{ MYSQLUSER }}:\${{ MYSQLPORT }}@\${{ MYSQLHOST }}

On the right, a code editor window shows a Python file with configuration settings:

```

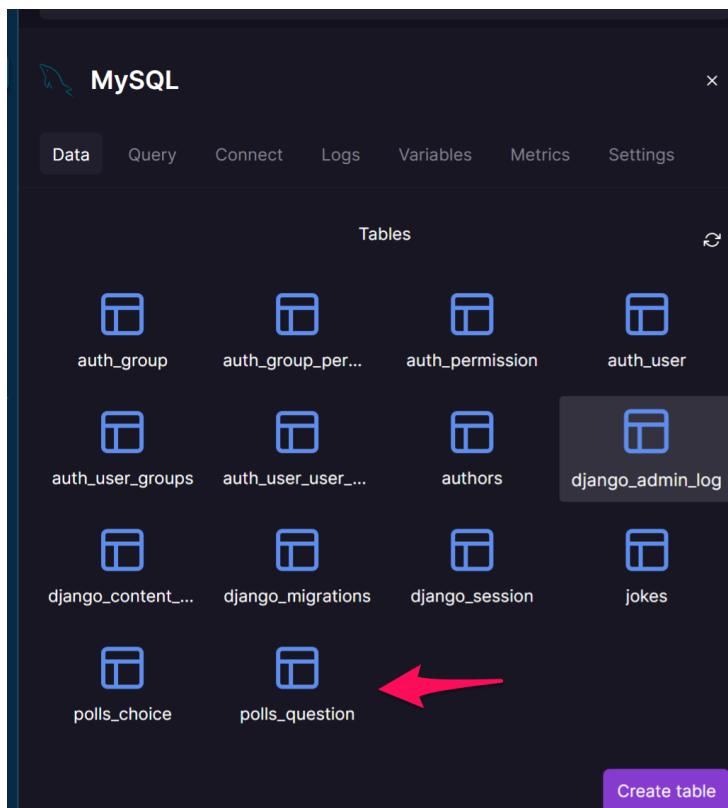
67     ],
68   },
69 ]
70 WSGI_APPLICATION = 'django_tutorial.wsgi.application'
72
73 # Database
74 # https://docs.djangoproject.com/en/3.1/ref/settings/#databases
75
76 DATABASES = {
77     'default': {
78         'ENGINE': 'django.db.backends.mysql',
79         'NAME': 'railway',
80         'USER': 'root',
81         'PASSWORD': 'ar849qIyZnNtthKPLXMyd',
82         'HOST': 'containers-us-west-98.railway.app',
83         'PORT': '6088',
84     }
85 }
86
87     Rubén Juárez, hace 7 meses • Initial commit ...
88     # Password validation
89     # https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validation
90
91 AUTH_PASSWORD_VALIDATORS = [
92     {
93         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',

```

Below the code editor are tabs for PROBLEMAS, SALIDA, TERMINAL, GITLENS, COMENTARIOS, and CONSOLA DE DEPURACIÓN.

- Como en la tarea 1, realiza la migración de la base de datos que creará la estructura de datos necesarias. Comprobamos que se han creado la base de datos y las tablas.

```
>> py manage.py makemigrations
>> py manage.py migrate
```

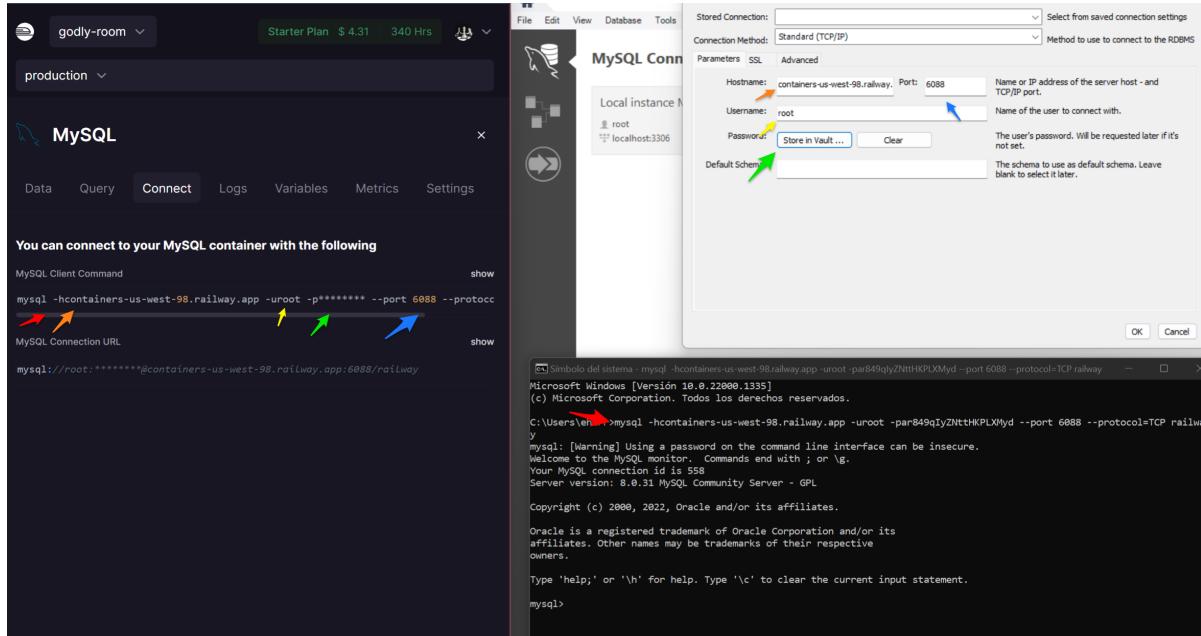


Nos fijamos como se han importado todas las tablas necesarias para nuestro proyecto.

- Crea un usuario administrador.

Para crear un usuario administrador en mysql tenemos que ingresar la aplicación a las variables de entorno de windows al igual que hicimos con sqlite.

Y ahora podemos trabajar desde nuestro cmd, aunque tambien podemos trabajar desde Mysql workbench.



Para aprender unos conceptos básicos de Mysql viene muy bien el siguiente video .

<https://youtu.be/uUdKAYI-F7g>

En la siguiente tabla se encuentran los usuarios administradores:

```
mysql> show tables;
+-----+
| Tables_in_railway |
+-----+
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
authors
django_admin_log
django_content_type
django_migrations
django_session
jokes
polls_choice
polls_question
+-----+
14 rows in set (0.20 sec)
```

Ahora mediante >> `INSERT INTO auth_user (propiedades) VALUES (datos_de_cada_propiedad)` podemos introducir los usuarios que queramos.
Lo podemos hacer tambien desde la terminal de python:

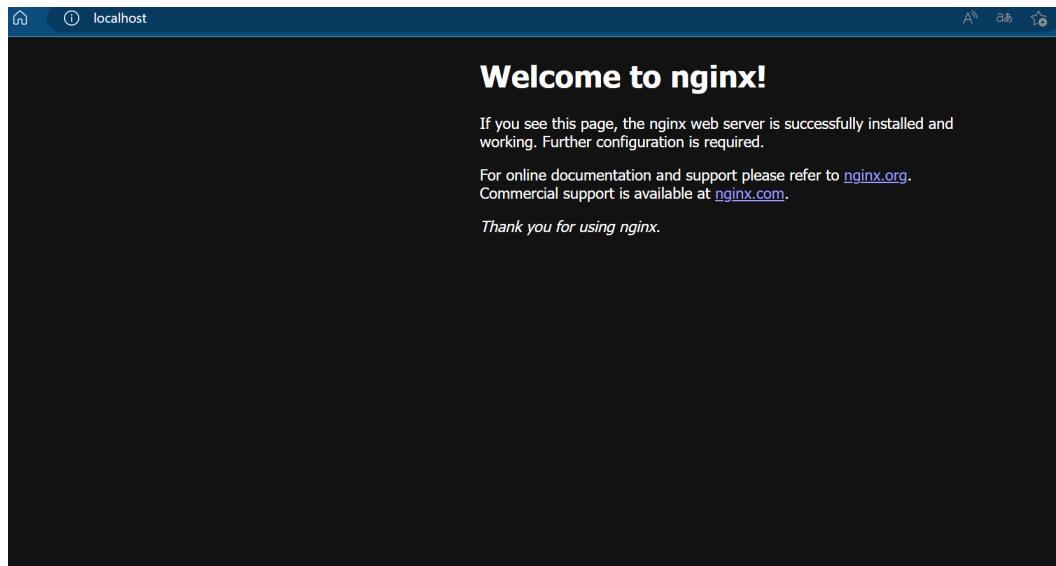
```
>> py manage.py createsuperuser
```

- Elige un servidor de aplicaciones python y configura nginx como proxy inverso para servir la aplicación.

El servidor elegido es gunicorn, que ya lo hicimos anteriormente. Para configurar nginx en windows lo hacemos a traves de la página oficial: [nginx: download](#).

Descomprimimos y lo guardamos en la ubicación que queramos dentro de C y lo añadimos al path como venimos haciendo anteriormente.

Para comprobar que lo hemos hecho bien podemos buscar localhost en nuestro navegador



Ahora abrimos nuestra carpeta de nginx en el editor de código. Buscamos sincronizarlo con nuestra aplicación.