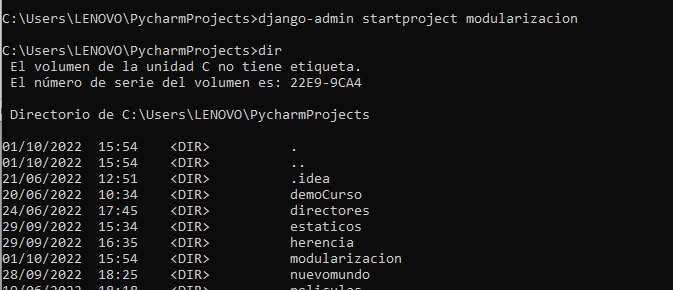
Doc Django

[**https://docs.djangoproject.com/**](https://docs.djangoproject.com/)

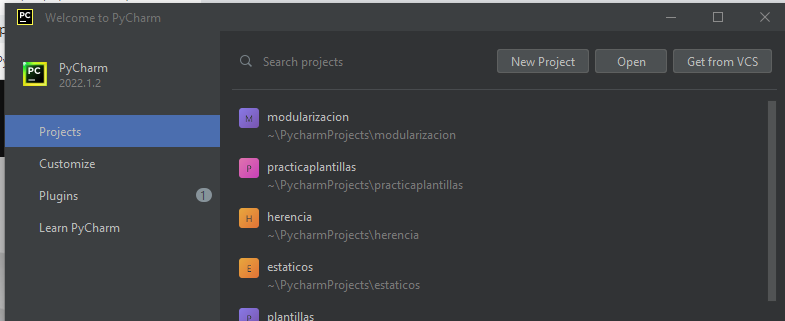
[**https://docs.djangoproject.com/en/4.1/topics/db/queries/**](https://docs.djangoproject.com/en/4.1/topics/db/queries/)

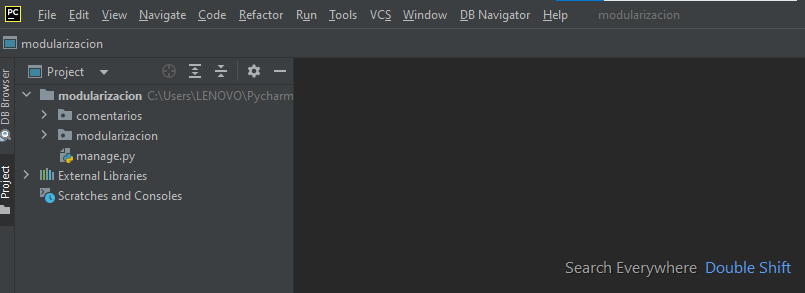
En CMD generamos el nuevo proyecto (Modularizacion) en carpeta PycharmProyect con

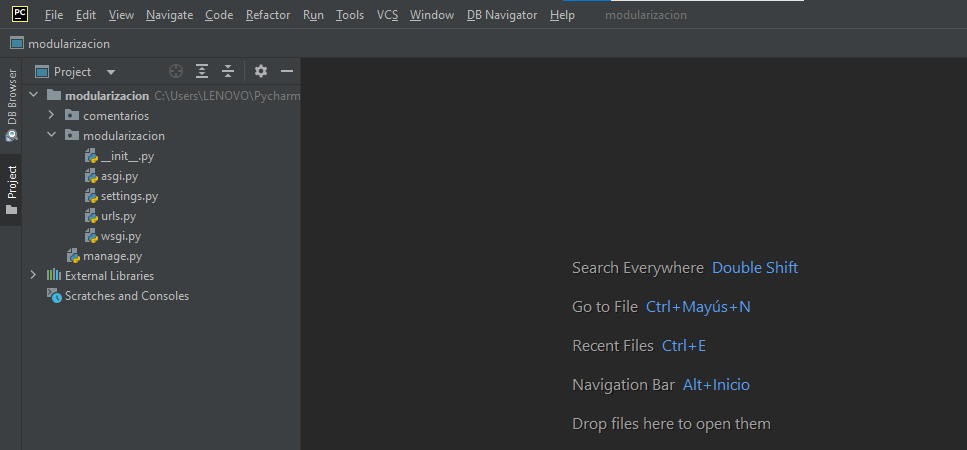
**django-admin startproject nombredel proyecto**



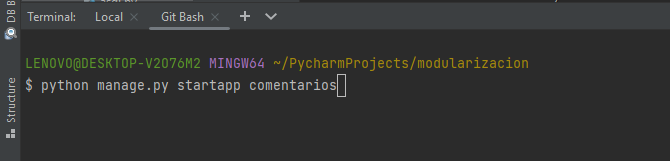
Abrimos el proyecto en PyCharm

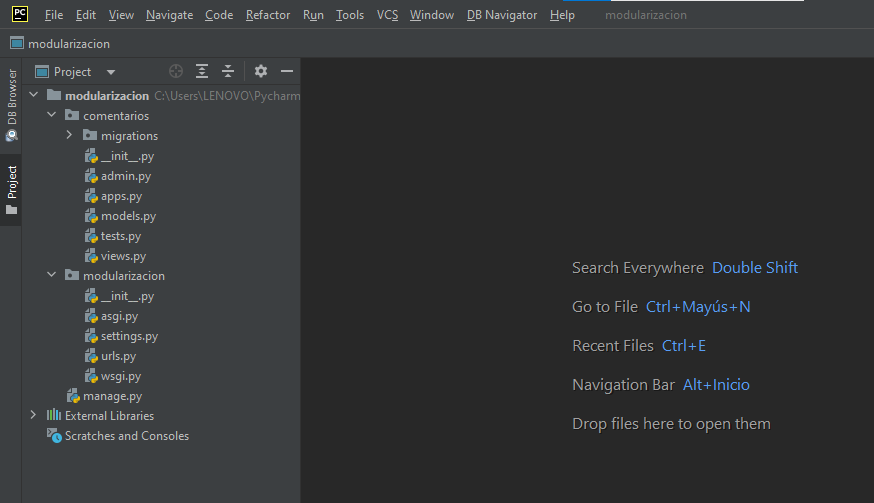




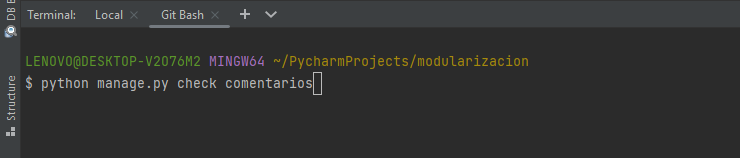


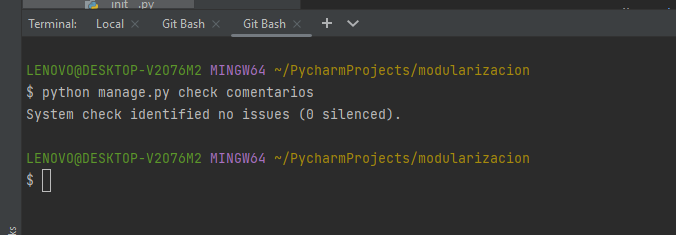
Dentro del proyecto modularización creamos una app. Comentarios. Desde gitbash con:



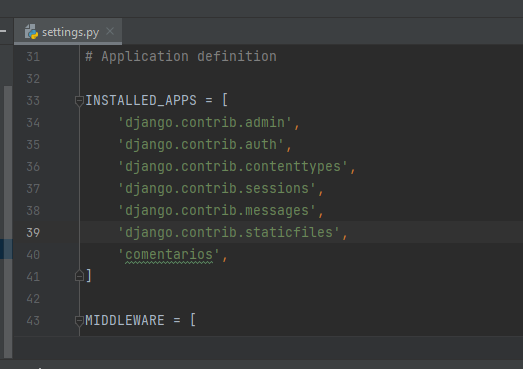


Verificamos en gitbash su creación

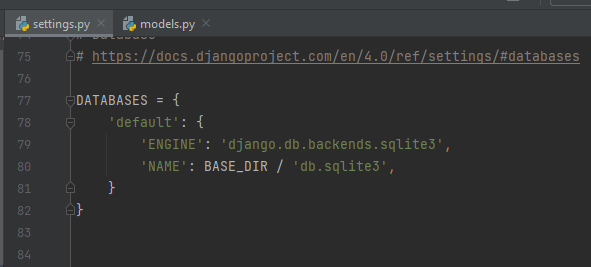




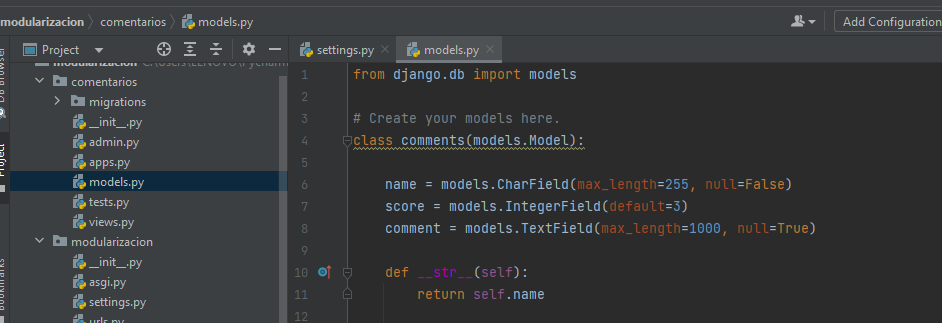
Agregamos la app creada en archivo setting.py del proyecto modularización



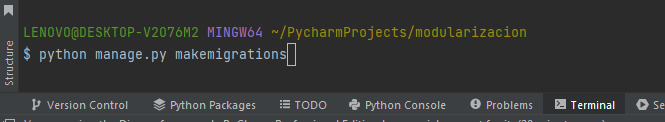
Django abstrae todo el manejo de la BD. Mantiene separado la app del motor de base de datos por lo que podemos cambiarlo de acuerdo a la necesidad del proyecto. En este caso el motor es sqlite3 y lo vemos en el archivo setting.py

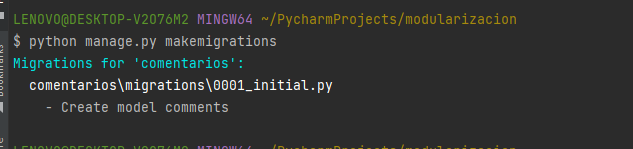


Comenzamos con el modelado de los datos.. Creamos clases con los atributos que necesitamos que persistan en nuestra BD

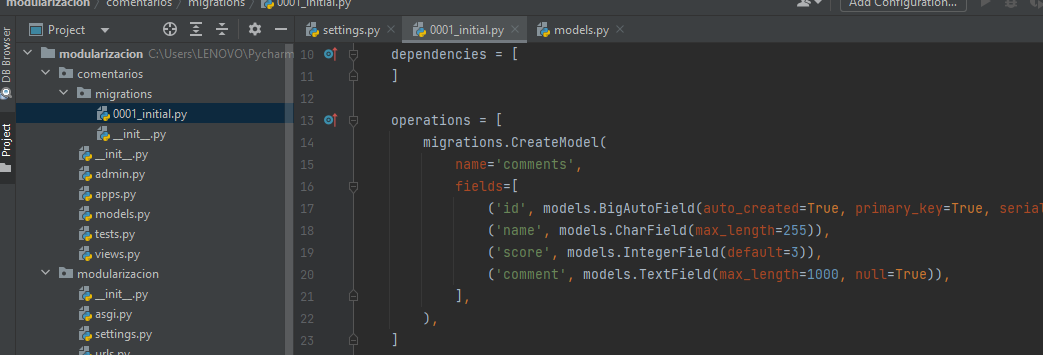


Solicitamos que los modelos estén en un formato SQL

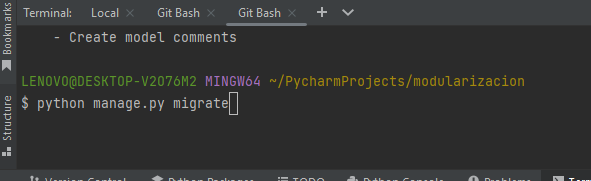


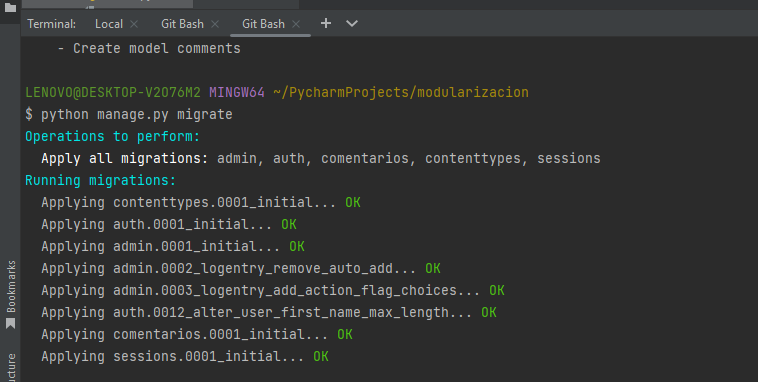


En el proyecto nos crea un archivo de migración inicial para iniciar la ejecución del código.

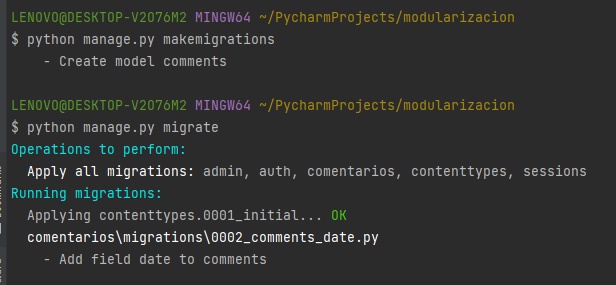


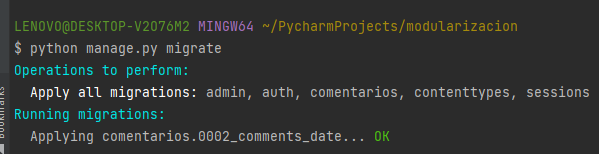
Luego ejecutamos la migración. Con esto se genera la BD



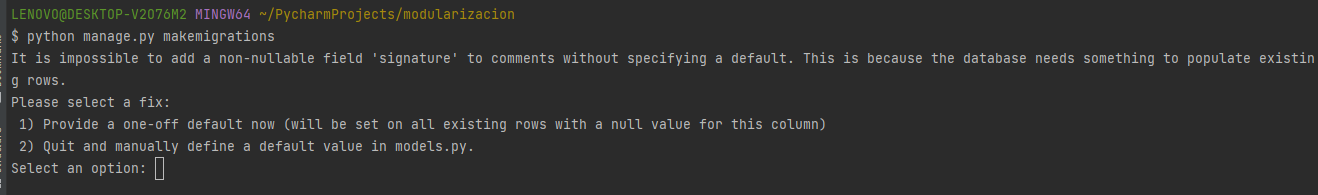


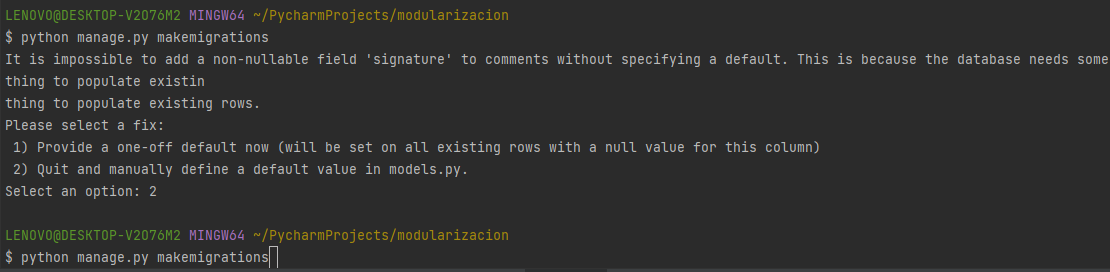
Cada vez que modifique el modelo tendre que hacer makemigrations y migrate.





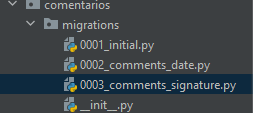
En caso de modificaciones del modelo. Si rompemos la integridad de los datos. Ejemplo cargamos un campo que no especificamos si acepta datos nulos habiendo datos ya cargados. Esto puede generar comportamientos inesperados. Por ello django con makemigration lo detecta y nos avisa







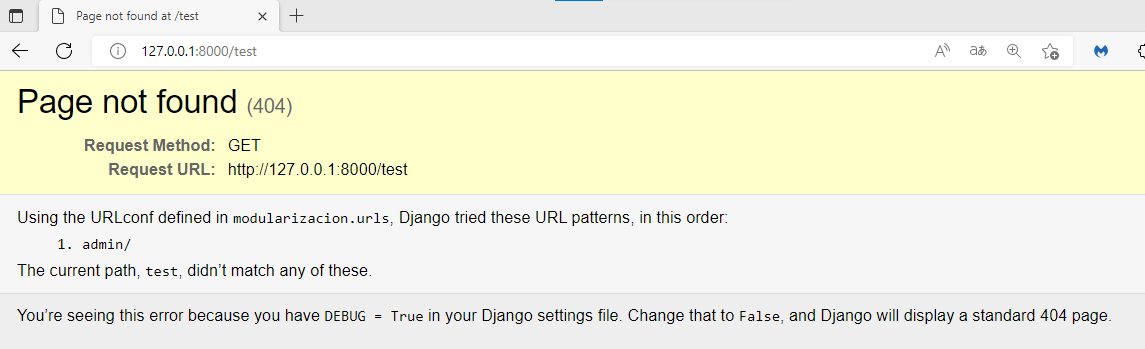
Aquí podemos ver las versiones y los cambios



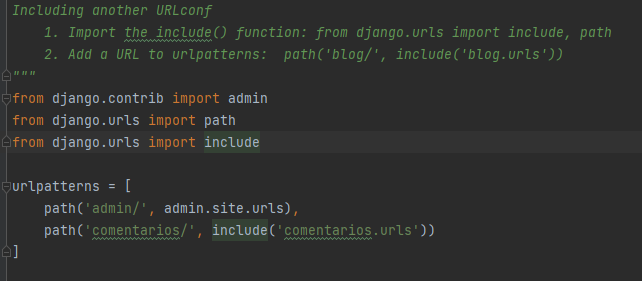
Como comunicamos a nuestro proyecto general con el contenido (en este caso app comentarios)

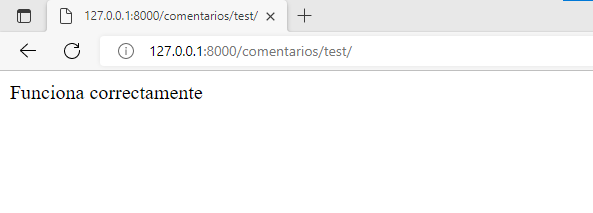
Genero un archivo urls.py en la carpeta de la app. (en este caso app comentarios). Tiene sus propias urls que se comunican con sus propias views.

Si levantamos el servidor con **$ Python manage.py runserver** . Todavia no lo vamos a poder ver



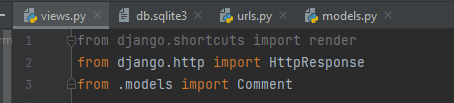
En el archivo urls del proyecto agregamos el path comentarios

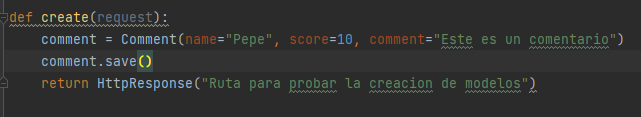




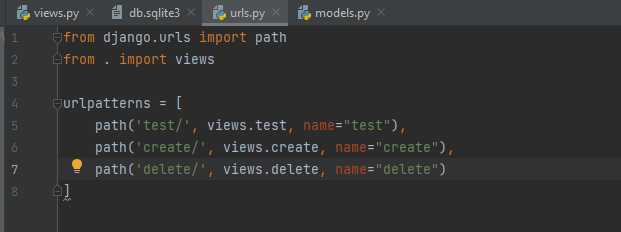
Creamos un objeto y lo guardamos en BD. Comentario

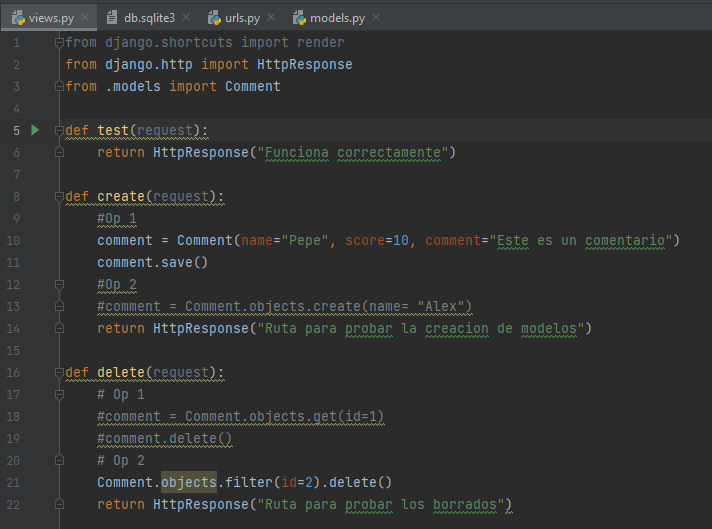
Importamos el modelo a las views





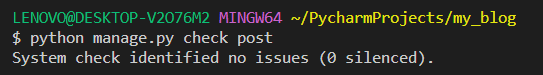
Crear (Inserción) y borrar un objeto

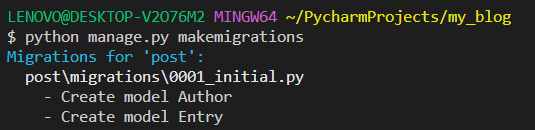


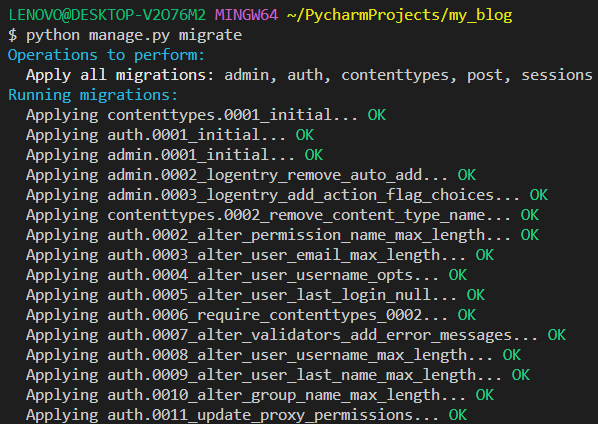


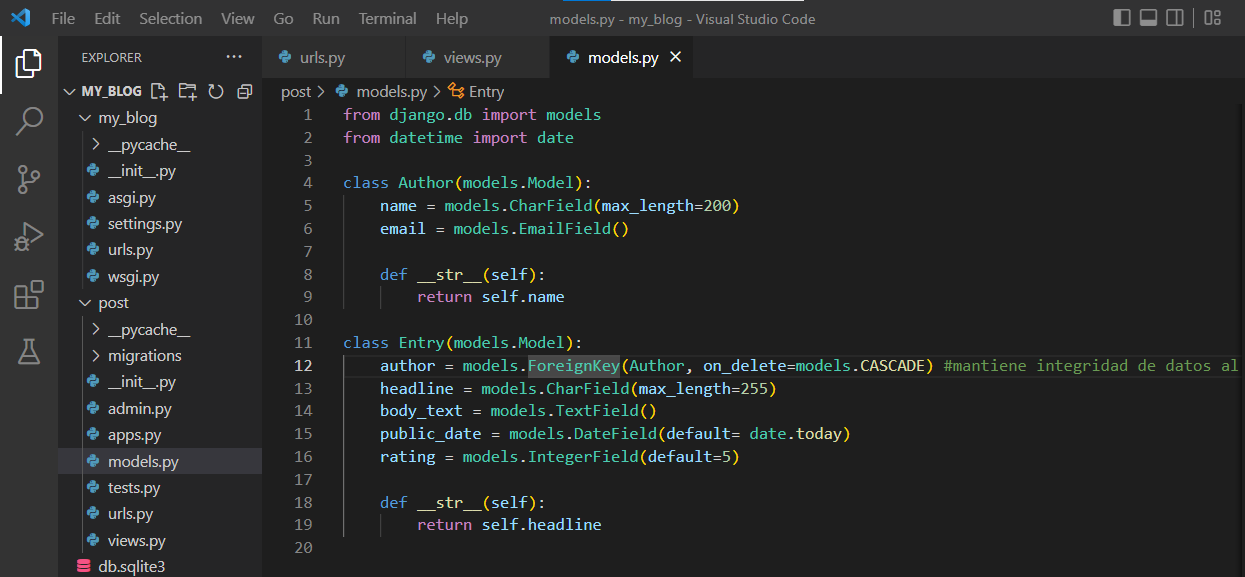
Estructura y claves foráneas. Creamos modelo con relación por campo autor (En VSC)





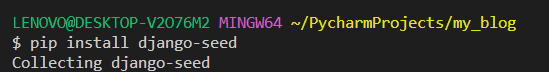


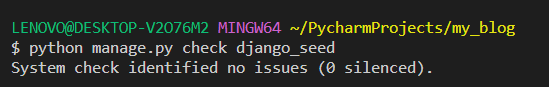




Seeding y paquetes para poblar la BD

Instalamos django-seed. Librería para poblar la base de datos

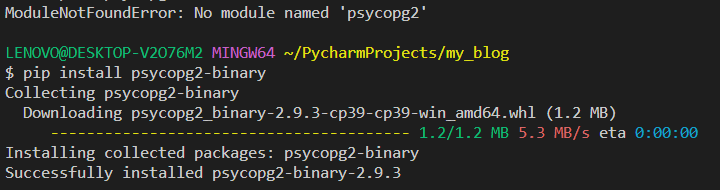


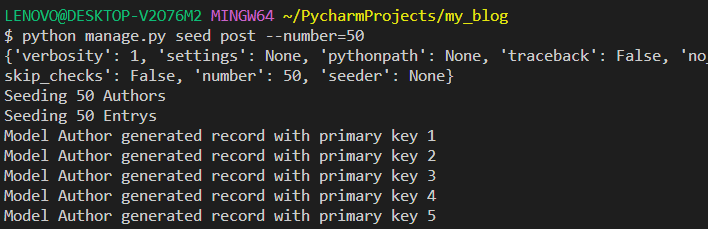


Generamos un seed de 50 datos

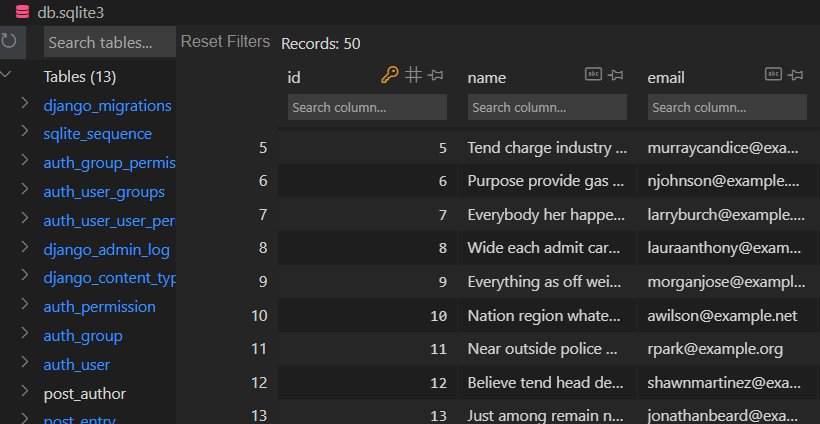


Nos dio error porque falto un modulo. Lo instalamos





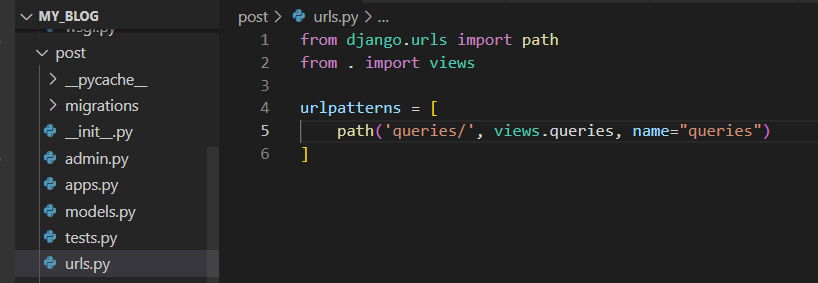
EN BD sqlite3 en este caso visto en VSC



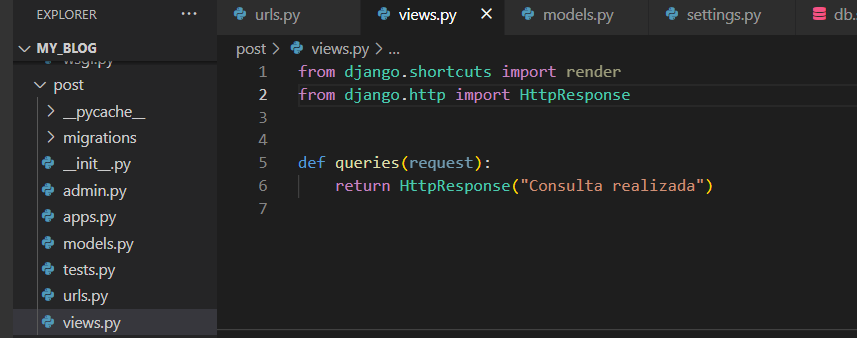


Consulta de datos I

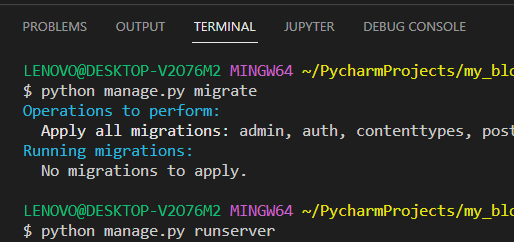
Generamos url en la app

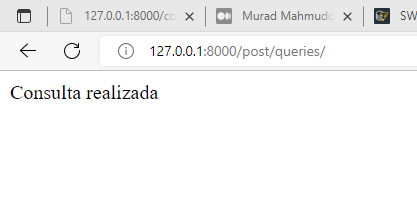


Generamos la view en la app

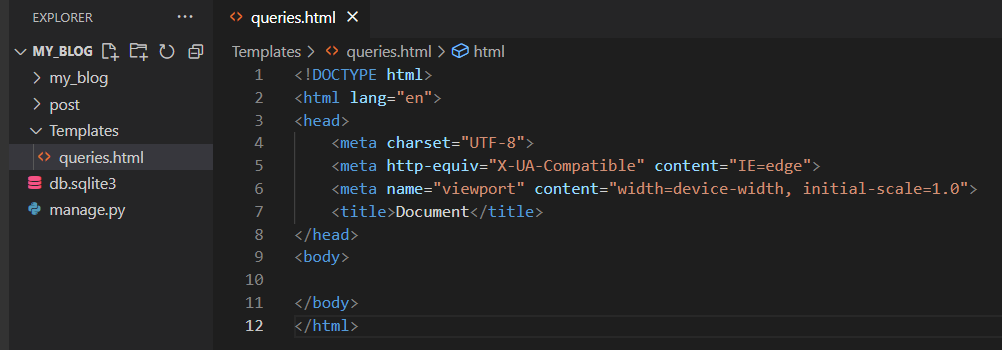


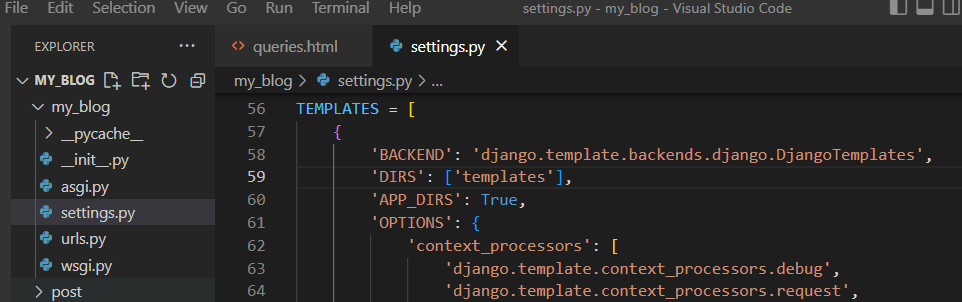
Verificamos la migración y iniciamos el server



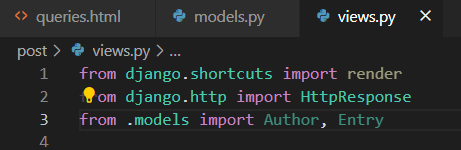


Generamos el tamplate para ver las consultas que generamos. Cargamos en setting del proyecto





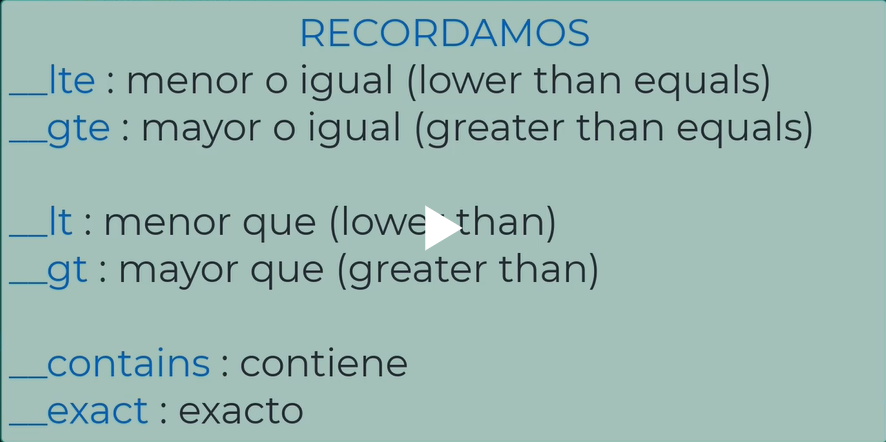
Importamos los modelos en las views



post/ queries/

**Consulta datos I y II**

Operadores



<https://docs.djangoproject.com/en/4.1/topics/db/queries/>

**Actualización de Datos**

**Relación uno a uno**

**Relación muchos a uno**

**Relación muchos a muchos**

[**https://docs.djangoproject.com/en/4.1/topics/db/examples/**](https://docs.djangoproject.com/en/4.1/topics/db/examples/)

**Conexión PostgreSQL**

**https://docs.djangoproject.com/en/4.1/ref/settings/#databases/**

[**https://docs.djangoproject.com/en/4.1/ref/databases/**](https://docs.djangoproject.com/en/4.1/ref/databases/)

**https://docs.djangoproject.com/en/4.1/topics/db/multi-db/**

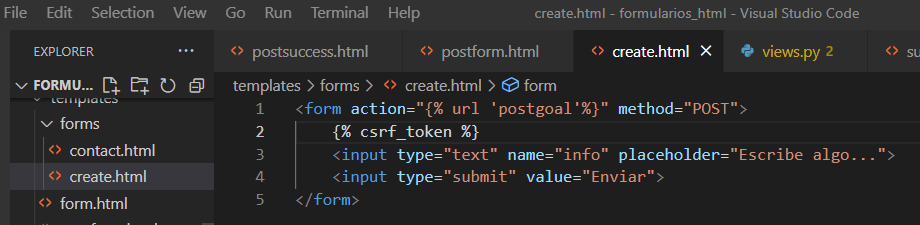
Formularios HTML Get

Get no se utiliza para envio de información sensible del usuario (ej contraseñas, credenciales)

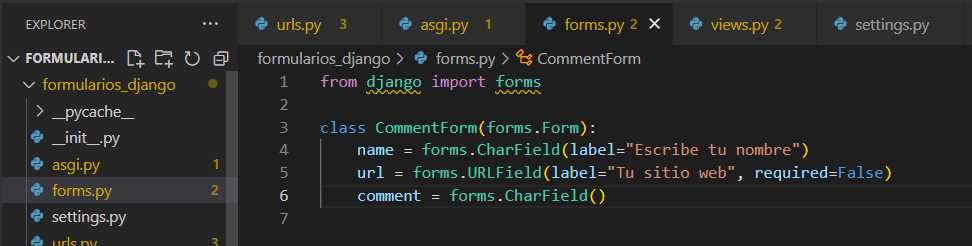
Post para envio de información sensible. Va codificada

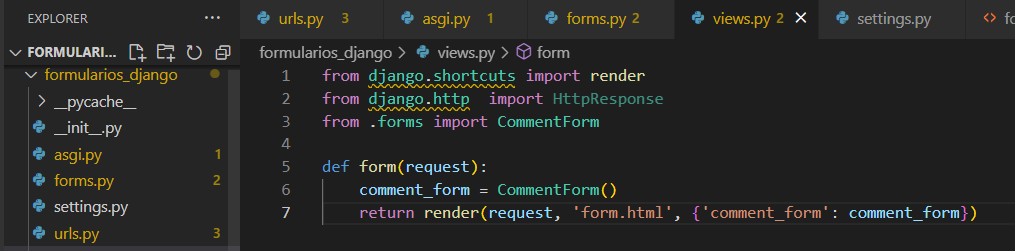
Formularios HTML Post

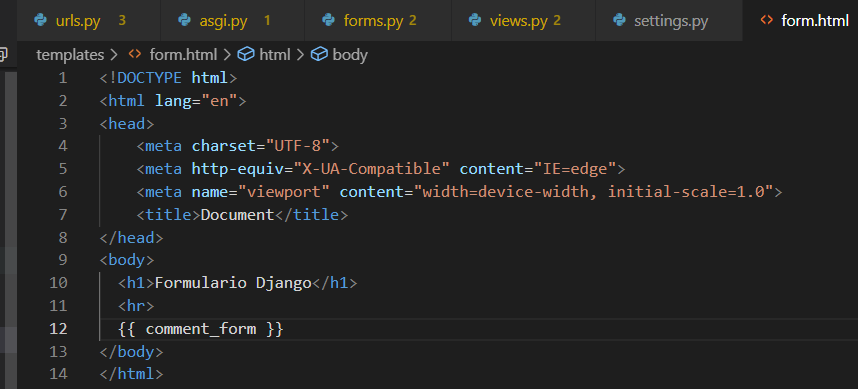
CSRF código de seguridad contra el robo de sesiones. Es una protección que se hace a través de un token. Cada vez que se hace un envio por POST se debe incluir.

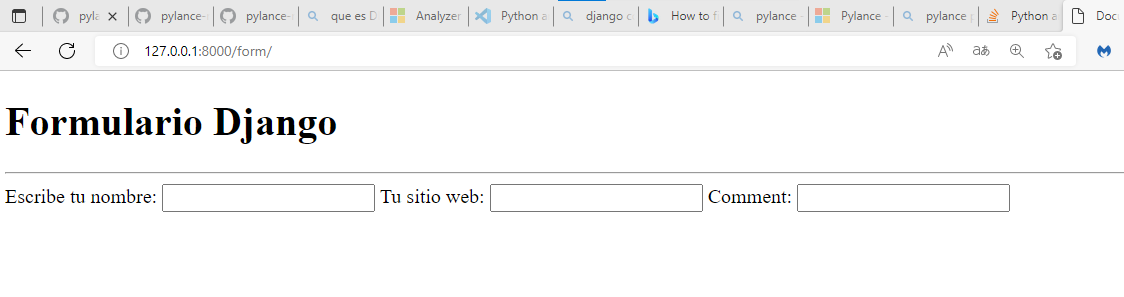


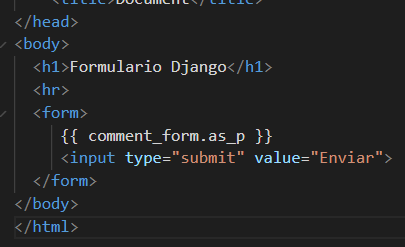
**Formularios a traves de clases (clase form de django )**

****

****

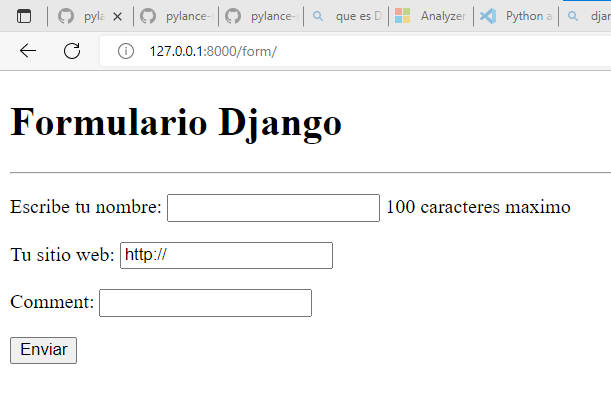
****

****

****

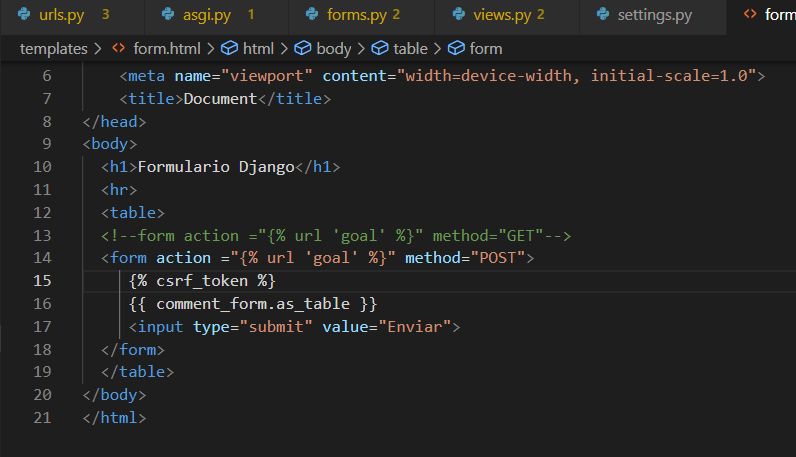
**as\_p -> párrafo**

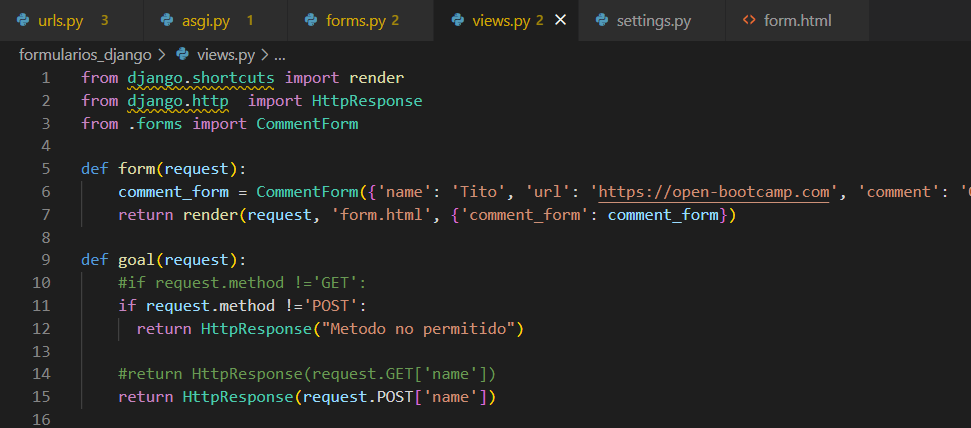
**as\_table -> en tabla**

****

****

**GET o POST**

****

****