*Solemne 2*

Tabla de contenido

[CREACION DE PROYECTO 3](#_Toc55548171)

[CREACION DE PROYECTO DENTRO DEL AMBIENTE VIRTUAL CREADO 5](#_Toc55548172)

[GUARDAR VERSIONES DEL PROYECTO EN GITHUB 9](#_Toc55548173)

[CREACION DE CRUD PARA LAS TABLAS CREADAS 14](#_Toc55548174)

[IMPLEMENTACION DE CASILLAS DE BUSQUEDA 17](#_Toc55548175)

[PRUEBAS UNITARIAS 18](#_Toc55548176)

[CREACION DE API REST 19](#_Toc55548177)

# CREACION DE PROYECTO

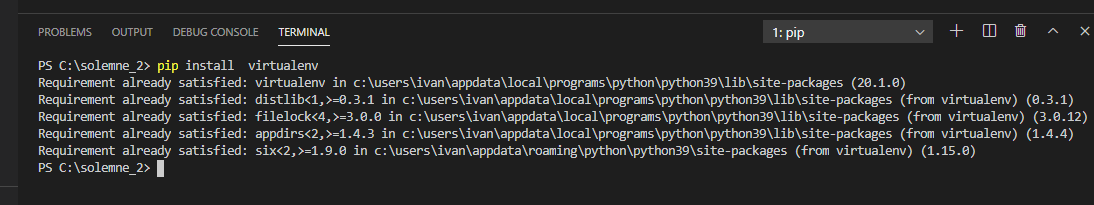
*Se creara una carpeta en C:, donde quedara el proyecto. En este caso se llamara solemne\_2*

Paso 1: INICIO DE CREACION DE VERSIONES GIT

*PS C:\solemne\_2> git init*

Paso 2: INTALACION DEL AMBIENTE VIRTUAL PARA SOLEMNE\_2

*PS C:\solemne\_2> pip install virtualenv*



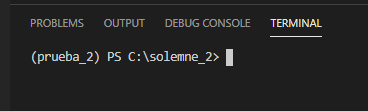
Paso 3: CREACION DEL AMBIENTE VIRTUAL EL CUAL SE LLAMARA prueba\_2

PS C:\solemne\_2> virtualenv prueba\_2

Paso 4: ACTIVACION DEL AMBIENTE VIRTUAL CREADO

*PS C:\solemne\_2> prueba\_2\Scripts\activate*

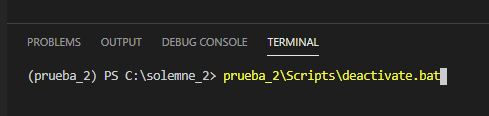
*Se vera de esta manera*



PARA

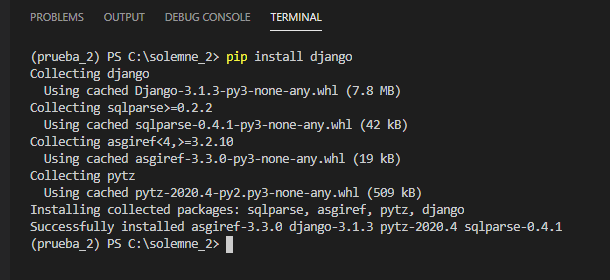
PARA DESACTIVAR EL AMBIENTE VIRTUAL

*(prueba\_2) PS C:\solemne\_2> prueba\_2\Scripts\deactivate.bat*

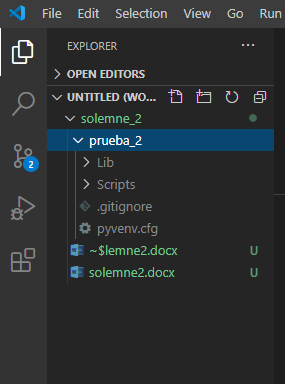


Paso 5: INSTALACION DE DJANGO DENTRO DEL AMBIENTE VIRTUAL CREADO

*(prueba\_2) PS C:\solemne\_2> pip install django*



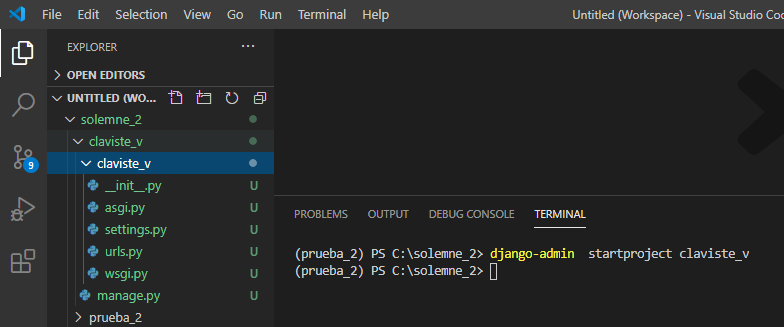
*De esta manera, debería quedar la carpeta virtual con django instalado.*



# CREACION DE PROYECTO DENTRO DEL AMBIENTE VIRTUAL CREADO

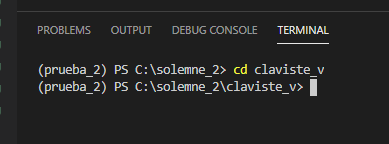
Paso 1: CREACION DEL PROYECTO, DENTRO DEL ALBIENTE VIRTUAL

*(prueba\_2) PS C:\solemne\_2> django-admin startproject claviste\_v* ***(puede ser cualquier nombre)***

*Se creara una carpeta con el nuevo proyecto*

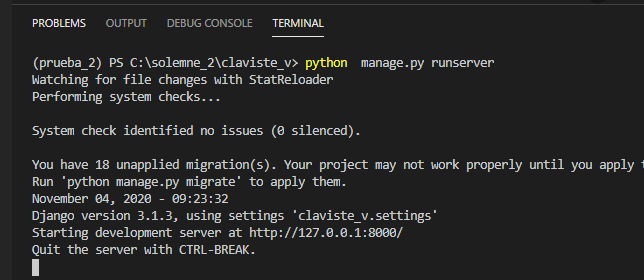
Paso 2: INGRESAR A LA CARPETA DEL PROYECTO CREADO

*(prueba\_2) PS C:\solemne\_2> cd claviste\_v*



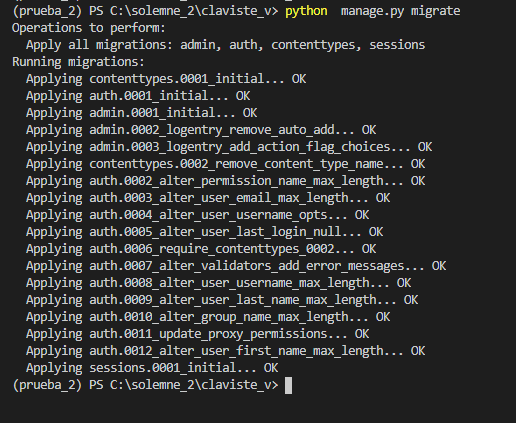
Paso 3: PROBAR SI EL PROYECTO LEVANTA

*(prueba\_2) PS C:\solemne\_2\claviste\_v>python manage.py runserver*



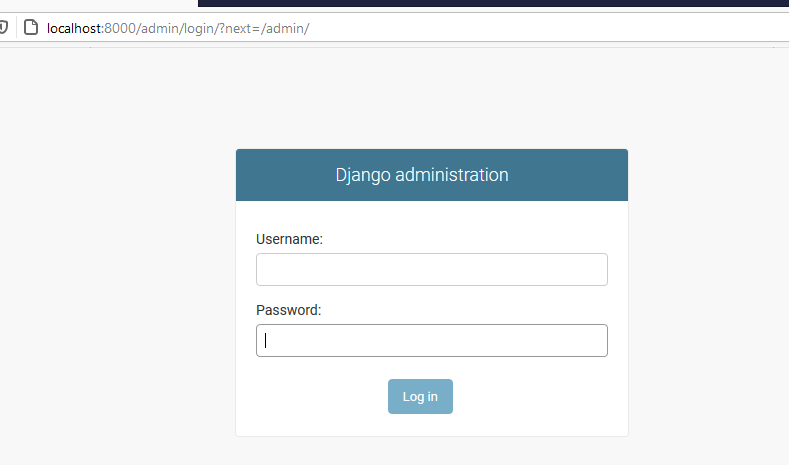
Paso 4: GENERAR LA MIGRACION DE LAS TABLAS POR DEFECTO DE DJANGO

*(prueba\_2) PS C:\solemne\_2\claviste\_v> python manage.py migrate*



*Paso 5:* Entrar al login ya creado en la migración y crear por línea de comando el usermane y password por defecto

***localhost:8000/admin*** y luego dar enter.

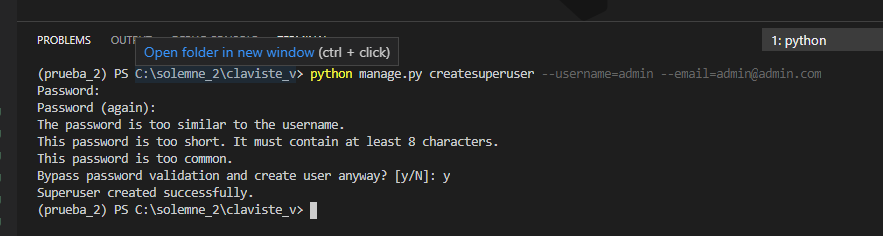


Por defecto no viene con usuario y password.

Ver paso 6

*Paso 6: Creación del usuario y password con súper usuario*

*(prueba\_2) PS C:\solemne\_2\claviste\_v> python manage.py createsuperuser --username=admin --email=admin@admin.com*



*Password: admin*

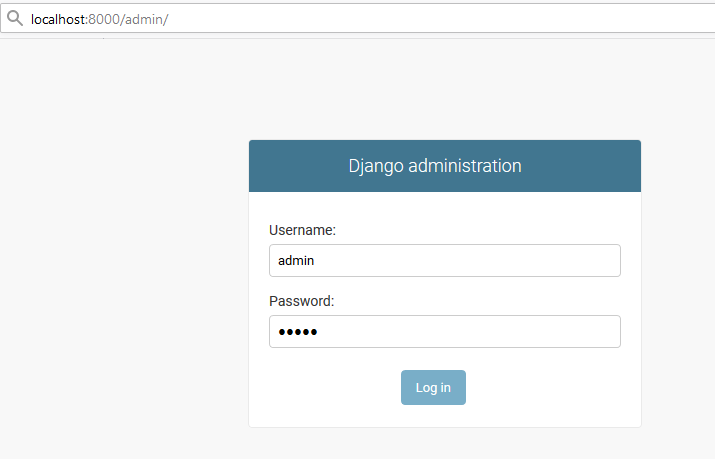
*Password (again): admin*

Paso 7: PROBAR SI EL PROYECTO LEVANTA

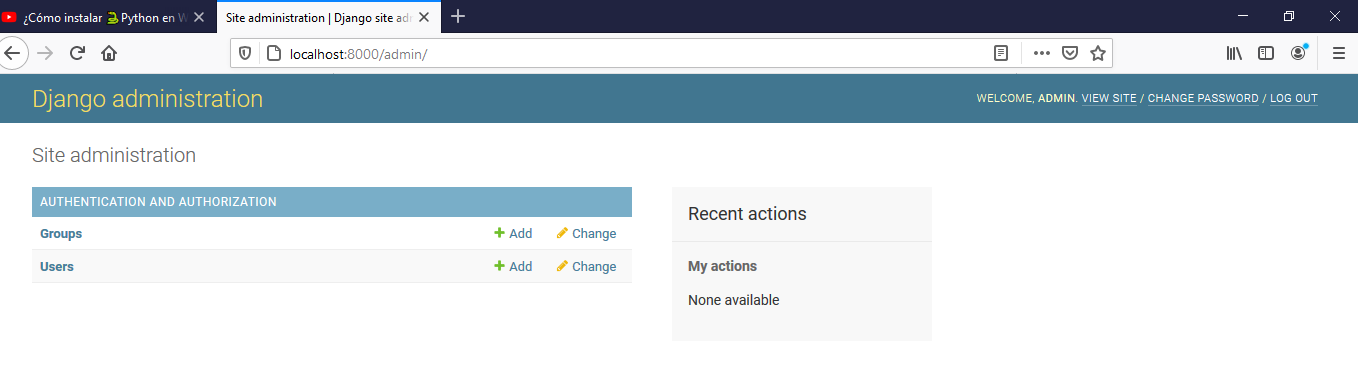
*(prueba\_2) PS C:\solemne\_2\claviste\_v>python manage.py runserver*

***localhost:8000/admin/***

en username y password escribir admin

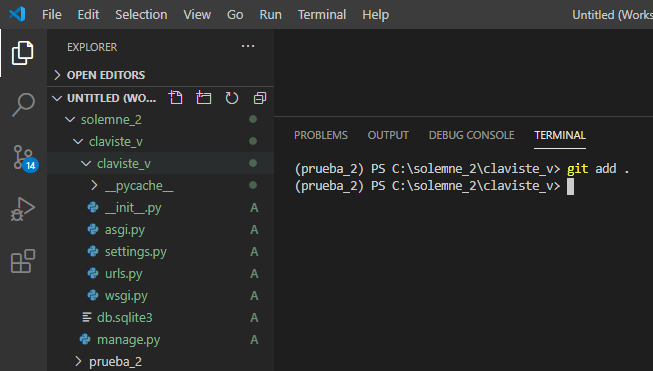


*Se debe abrir esta página. Con esto ya quedo el proyecto en el entorno virtual con control de usuarios.*



Paso 8: Se guardan los cambios en el control de versión con GIT

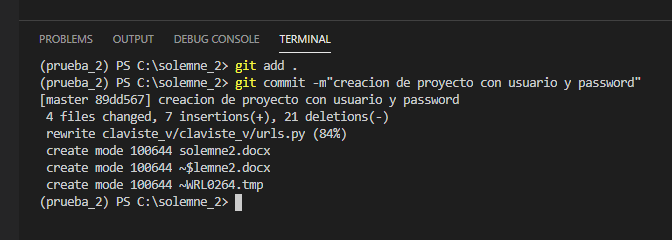
*(prueba\_2) PS C:\solemne\_2\claviste\_v> git add .*



Paso 9: Para finalizar se crea un commit para control de versiones

*(prueba\_2) PS C:\solemne\_2\claviste\_v> git commit –m "creacion de proyecto con usuario y password"*

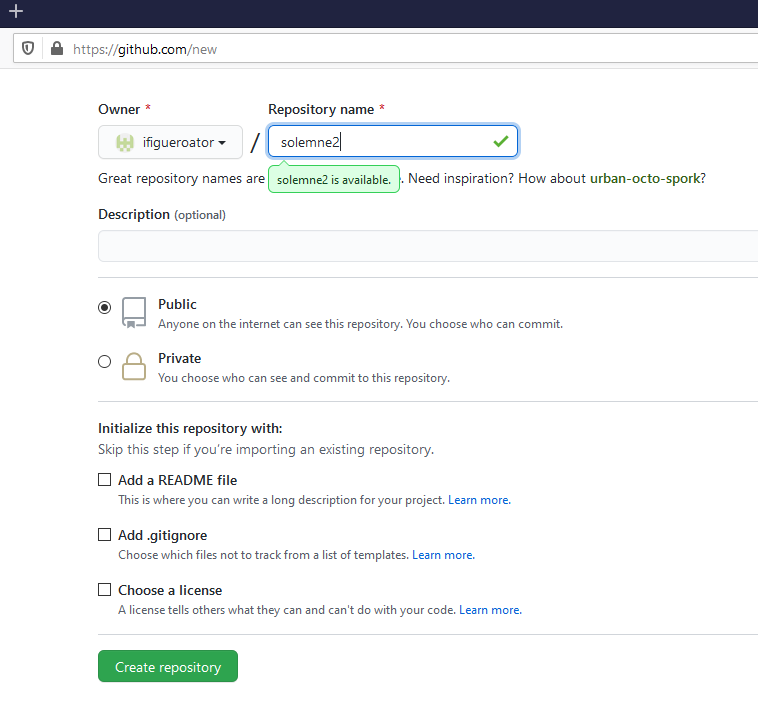
*Se vería de esta manera, luego de hacer un commit*

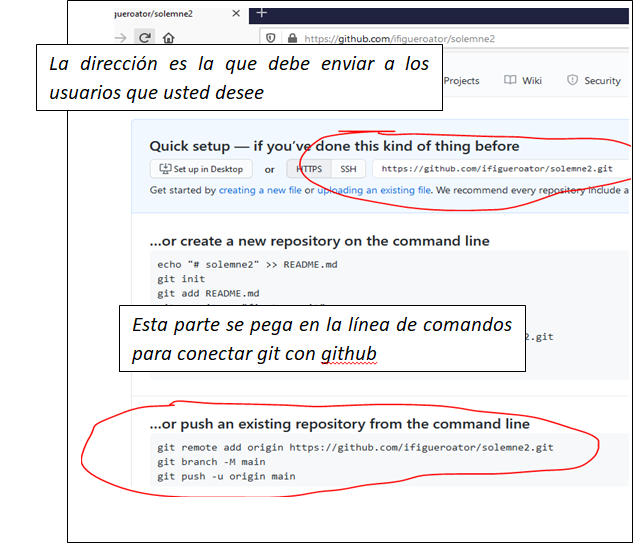


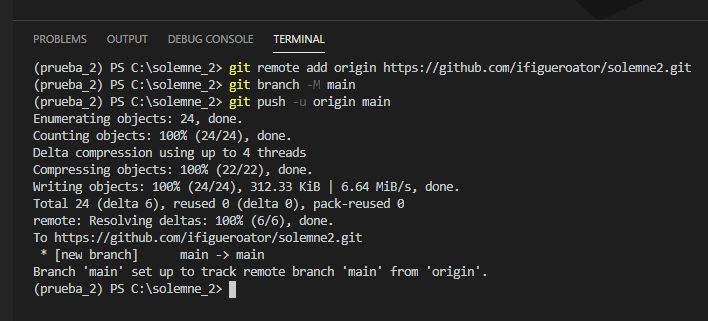
# GUARDAR VERSIONES DEL PROYECTO EN GITHUB

*Nota: se entiende que ya tiene creada una cuenta en github*

*Paso 1: ingresar a GITHUB y crear un nuevo proyecto*





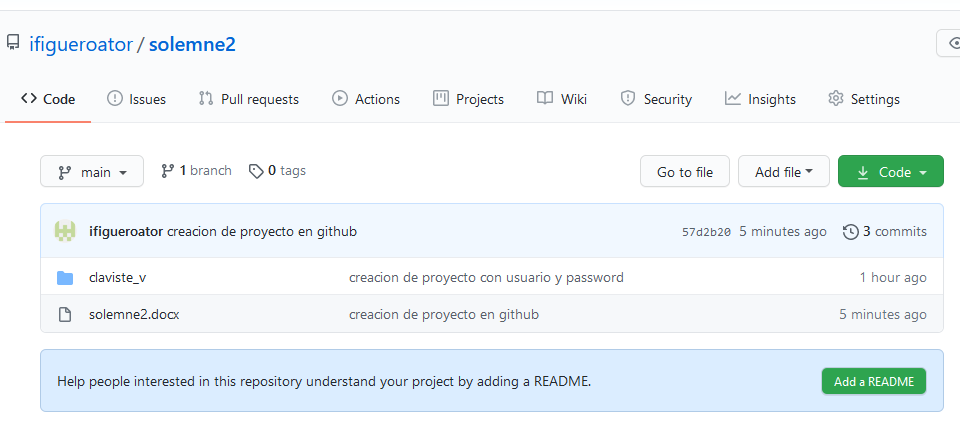


*Para actualizar nuevamente, utilizar el siguiente comando.*

*Nota: Recuerde tener todo guardado*

*(prueba\_2) PS C:\solemne\_2> git push -u origin main*

*En el GITHUB, se vera de la siguiente manera*

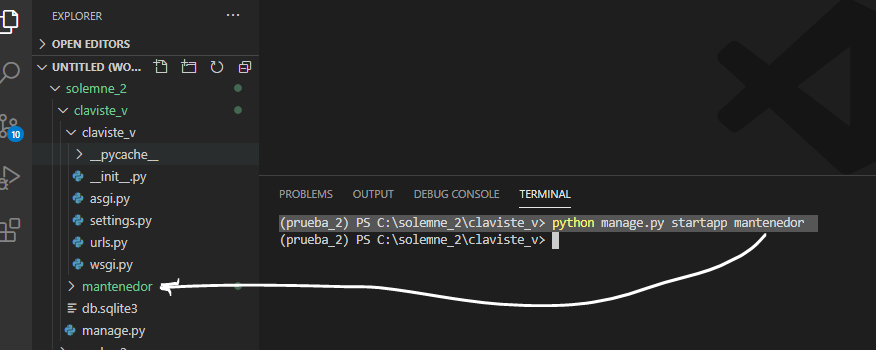


*CREACION DE APLICACIÓN PARA PROYECTO CLAVISTE\_V*

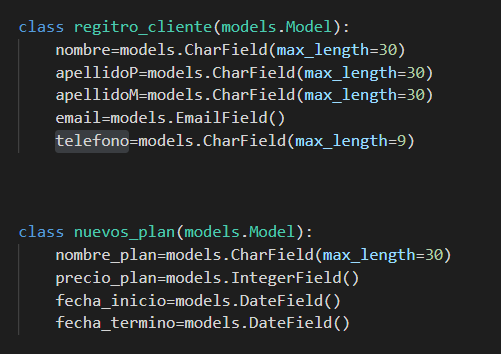
*Para este proyecto se creara una aplicación para tipo de planes e ingreso de clientes.*

Paso 1: creación de la APP

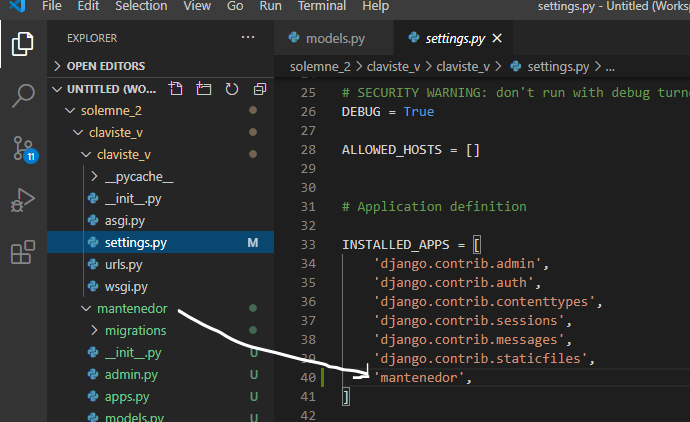
*(prueba\_2) PS C:\solemne\_2\claviste\_v> python manage.py startapp mantenedor*



Paso 2: creación de los modelos, los cuales migraran a la base de datos. Click en models.py

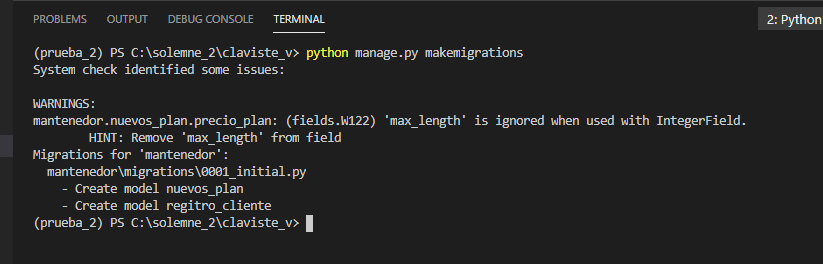


paso 3: Ir a setting.py y definir la aplicación mantenedor



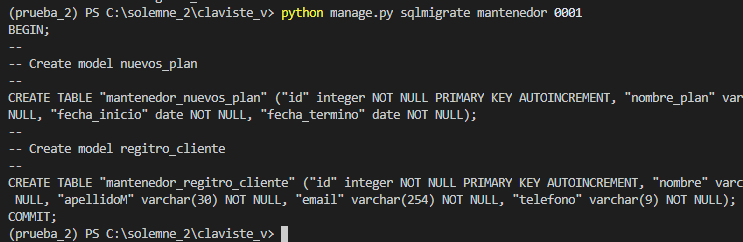
Paso 4: crear las migraciones

(prueba\_2) PS C:\solemne\_2\claviste\_v> python manage.py makemigrations



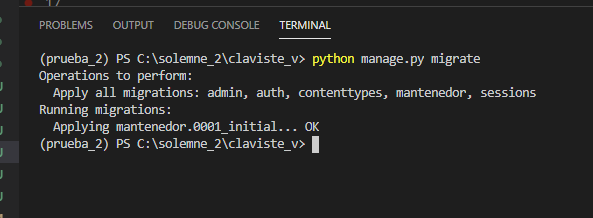
Paso 5: creación del código sql, para la creación de la base de datos

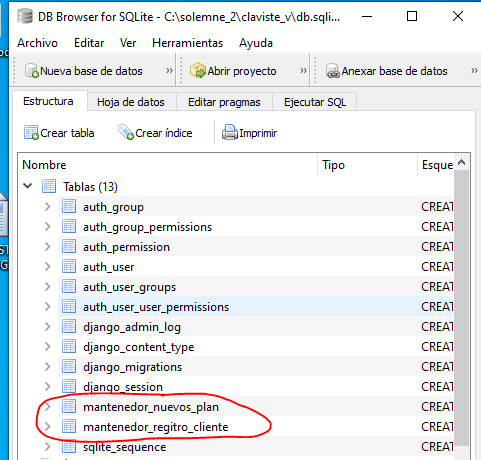
(prueba\_2) PS C:\solemne\_2\claviste\_v> python manage.py sqlmigrate mantenedor 0001



Paso 6: crear la migración para que se hagan las tablas en la base de datos

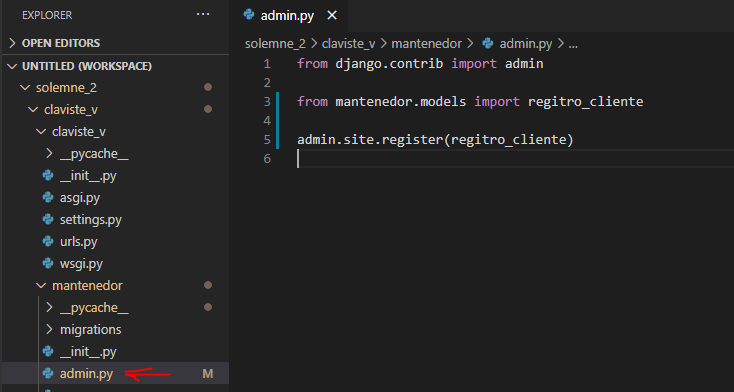
(prueba\_2) PS C:\solemne\_2\claviste\_v> python manage.py migrate





# CREACION DE CRUD PARA LAS TABLAS CREADAS

Paso 1: Ir la Carpeta de la aplicación y hacer click en el archivo admin.py, y escribir el código de la imagen.

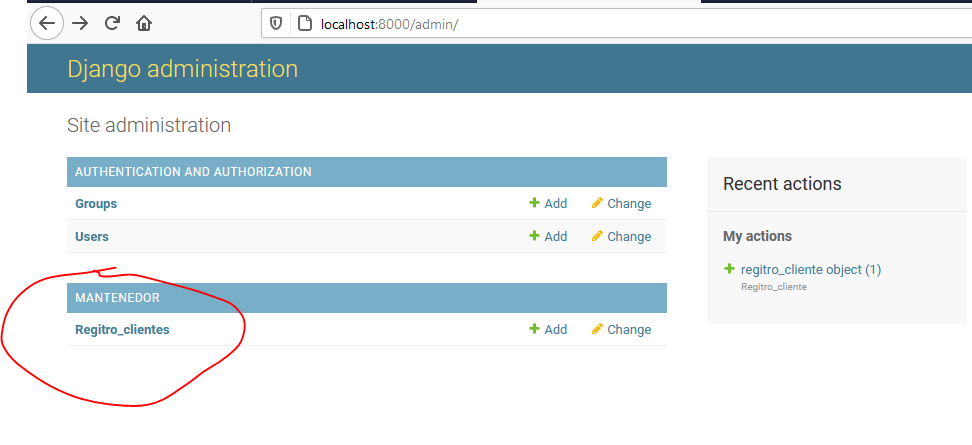


Guardar y levantar el servidor

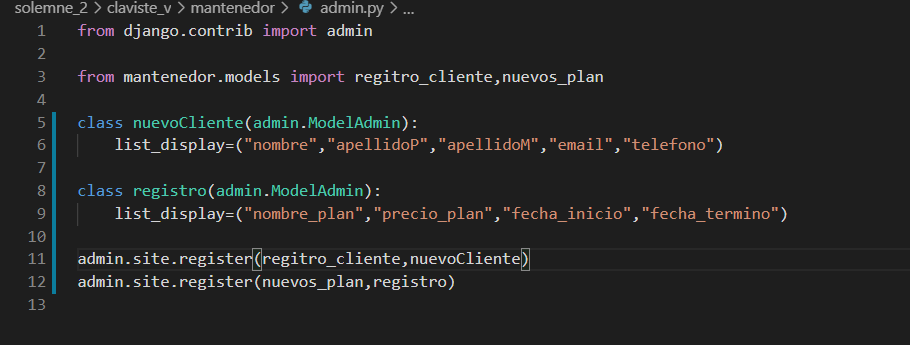
(prueba\_2) PS C:\solemne\_2\claviste\_v> python manage.py runserver

Paso 2: ingresar al panel de control. Se debe ver el titulo de la Aplicación con el nombre de la tabla creada en el paso 1.

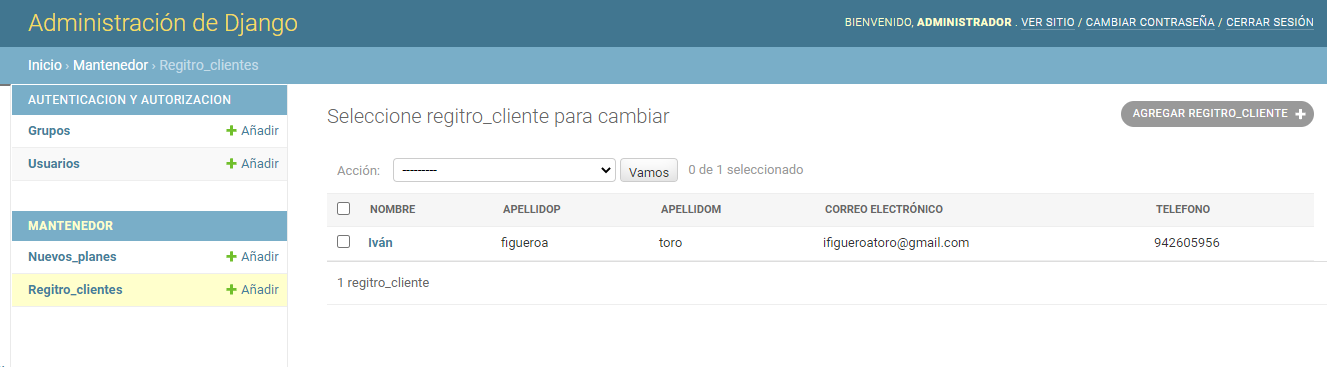
Ya puedes ingresar un nuevo cliente.



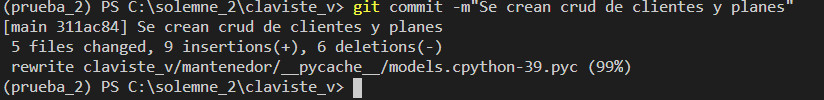
Para el ingreso de las demás tablas, se repite lo mismo que el paso 1.



Queda de la siguiente manera

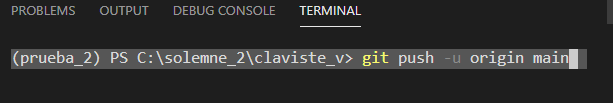


Paso 3: Guardamos los cambios en control de versiones GIT commit

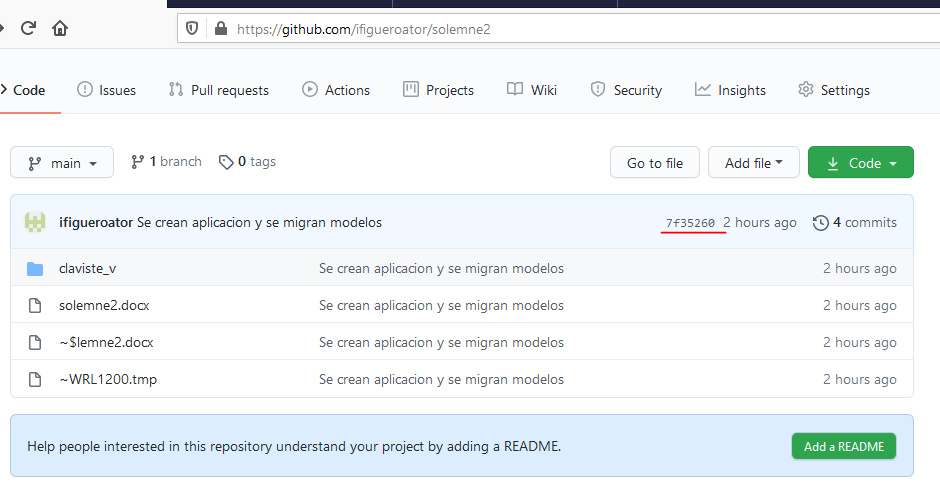
(prueba\_2) PS C:\solemne\_2\claviste\_v> git commit -m"Se crean crud de clientes y planes" 

Paso 3: actualizar GITHUB con el crud

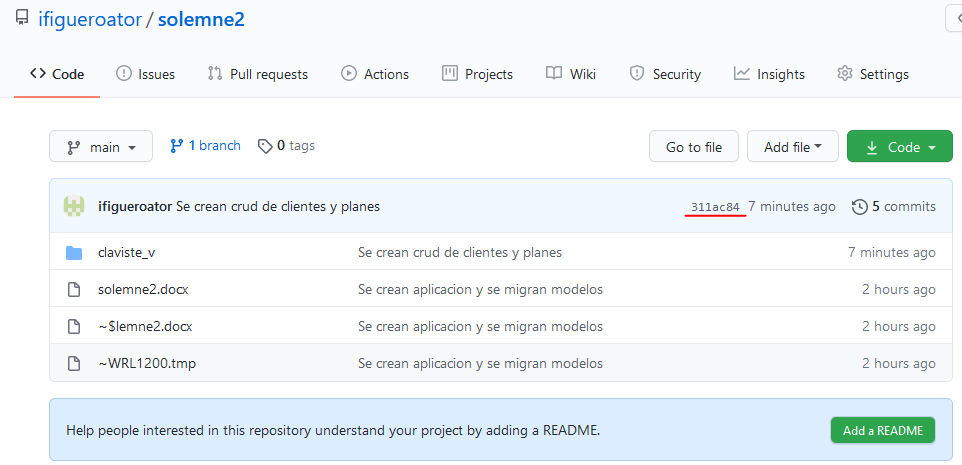
(prueba\_2) PS C:\solemne\_2\claviste\_v> git push -u origin main



Anterior

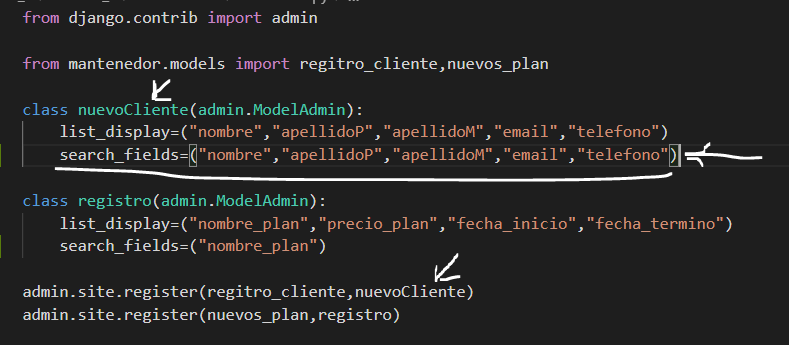


Nuevo



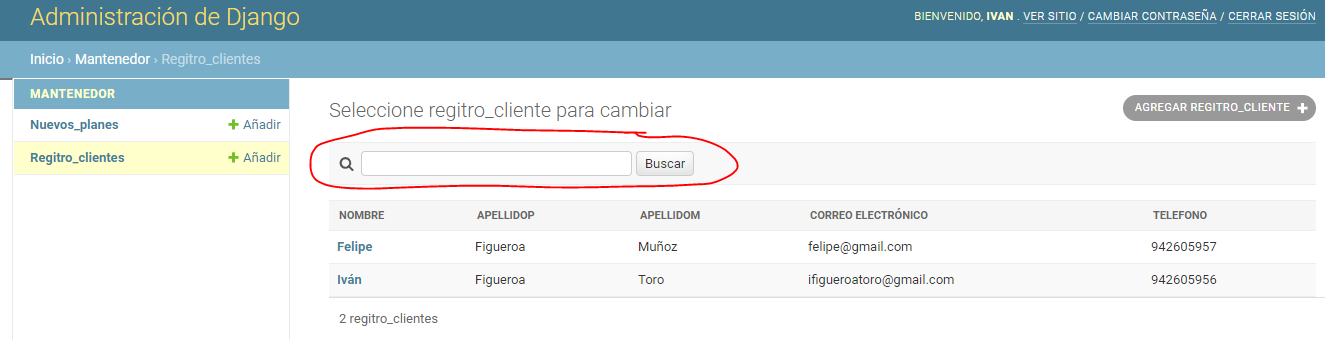
# IMPLEMENTACION DE CASILLAS DE BUSQUEDA

Paso 1: agregar comando search\_fields y agregar los nombre de las columnas



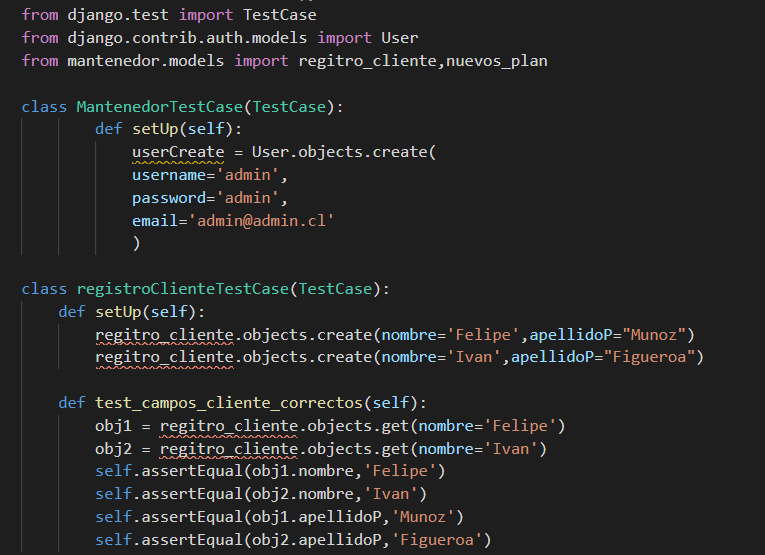
Paso 2: guardar los cambios y levantar el servidor

Quedara de esta manera



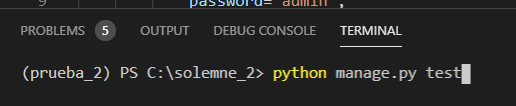
# PRUEBAS UNITARIAS

Paso 1: Creación de clase para las pruebas unitarias.

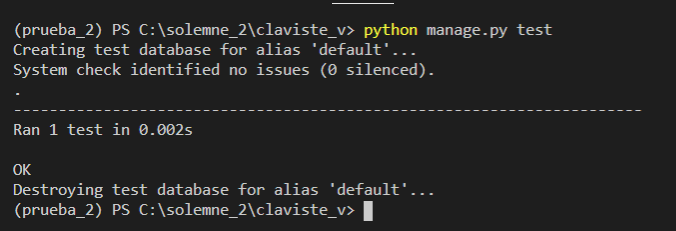


Paso2: Correr la pruebas con el siguiente comando

(prueba\_2) PS C:\solemne\_2\claviste\_v> python manage.py test



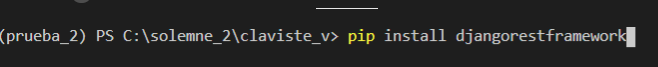
PRUEBA SIN ERROR



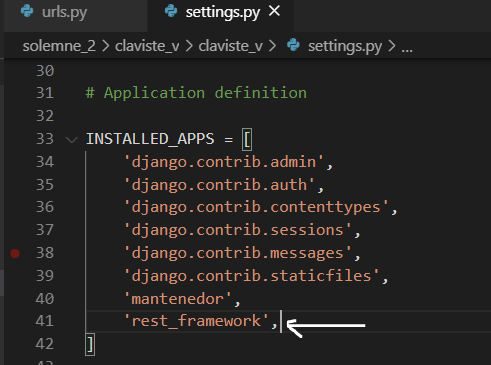
# CREACION DE API REST

Paso 1: instalación del paquete API REST para django dentro de la aplicación

(prueba\_2) PS C:\solemne\_2\claviste\_v> pip install djangorestframework

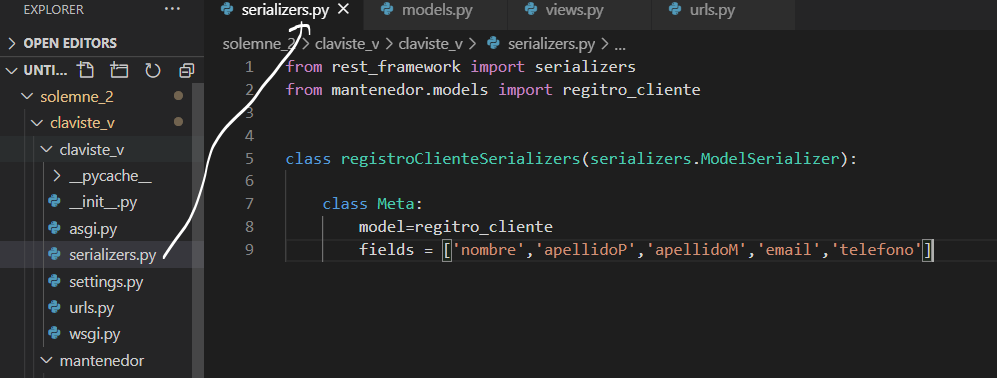


Paso 2: Definir la aplicación rest\_framenwork en el archivo settings.py

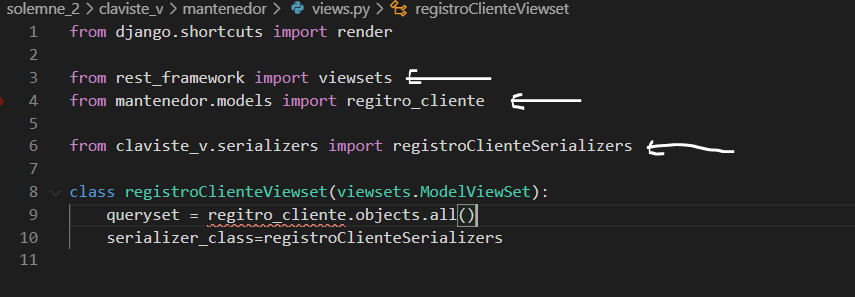


Paso 3: Generar un archivo dentro de la carpeta de cliente llamada serialaizers.py

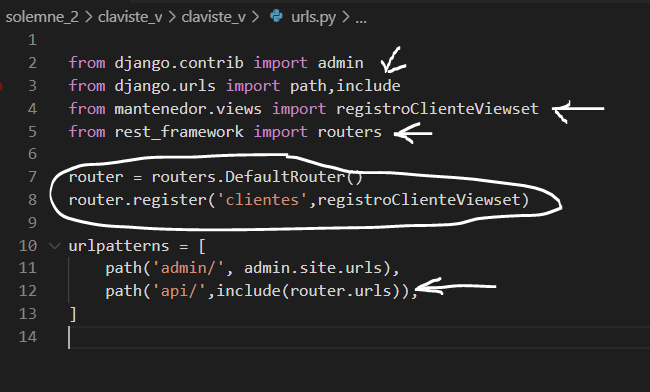
* hacer una clase nombrar según gusto o representatividad
* hacer una sub clase llamada Meta
* nombra una variable, que en este caso se llama model y igualarla a la tabla importada, que en este caso es regitro\_cliente
* usar variable fields e igualar con los nombres de columna exactos que tenga la tabla



Paso 4: importar Bibliotecas y modelos. Generar una clase en el archivo views.py



Paso 5 : importar las bibliotecas y generar las url



Paso 6: probar la API levantando el servidor.

