

Universidad de Oriente
Núcleo Anzoátegui
Escuela de Ingeniería y Ciencias Aplicadas
Departamento de Ingeniería de Computación y Sistema
Taller de Desarrollo de Software



**Conexión a desde Python a una Base de Datos
con Ejemplo práctico usando interfaz gráfica,
para 3 niveles administrador, usuario y cliente.**

Bachilleres:

Agreda, Luis C.I 26.886.935

Prado, Fadel C.I 26.346.932

Yick, Carlos C.I 25.879.703

Barcelona, 19 de mayo de 2023.

Python y Django

A pesar que actualmente en el mundo del desarrollo backend existen muchos frameworks muy producidos en distintos lenguajes de programación y cada año tenemos nuevas alternativas, de las cuales DJANGO sigue siendo un lenguaje de backend muy utilizado.

Django es un framework desarrollado web backend creado en python, es decir, si ya conoces las bases de python, solo tendrías que estudiar un poco de html, css y Javascript y con lo básico puedes pasar a ser aplicaciones web con este framework. De hecho, Django tiene un enorme número de características muy útiles que ayudan a simplificar el desarrollo de aplicaciones web.

Concepto básicos:

Model View Template(MVT):

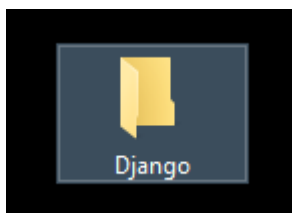
Este es un patrón que hace referencia en la forma en que creamos una URL. Esta llamará una función que es el view y esta consultará a la base de datos que es el model y este enviará un HTML al frontend que sería el template. Concepto casi igual al modelo vista controlador.

Object Relational Mapping(ORM):

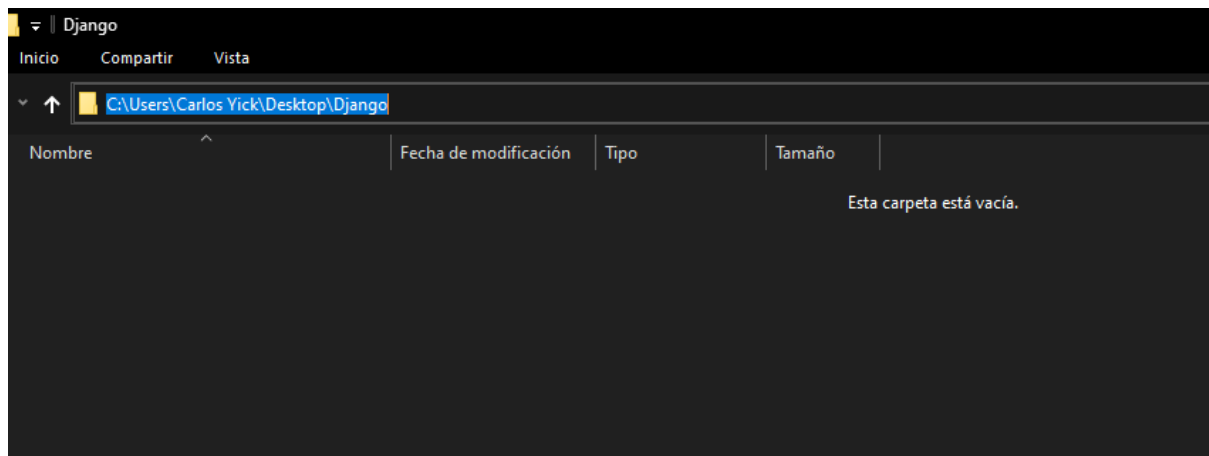
Un ORM es un módulo o paquete que maneja por ti las consultas que van a tu sistema gestor de base de datos, es decir, que en lugar de escribir SQL que requiere trabajo y conocimiento mantener, Django te da funciones que crearán por ti las consultas SQL y podrás utilizarlas con múltiples bases de datos entre ellas PostgreSQL, MariaDB etc.

Primero crearemos un proyecto en Django

Una vez ya teniendo todos los recursos instalados en nuestro computador vamos a crear nuestro primer proyecto usando el framework Django. El primer paso para esto es tener una carpeta (recomendablemente vacía) donde puedas crear dicho proyecto. En este ejemplo usaremos esta carpeta llamada **"Django"**:



Luego necesitamos conocer la ruta a dicha carpeta. Para hacerlo de forma sencilla puedes abrir dicha carpeta y copiar la dirección que te da el explorador de archivos como en el ejemplo:



Debemos ahora ir al símbolo del sistema con permisos de administrador y movernos hacia la ruta de la carpeta. Para esto debemos escribir “cd ruta”. En la imagen pueden ver como hacerlo de forma práctica.

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>cd C:\Users\Carlos Vick\Desktop\Django
```

Si lo hiciste correctamente el símbolo del sistema debería indicar que ya estas en la carpeta así:

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>cd C:\Users\Carlos Vick\Desktop\Django
C:\Users\Carlos Vick\Desktop\Django>
```

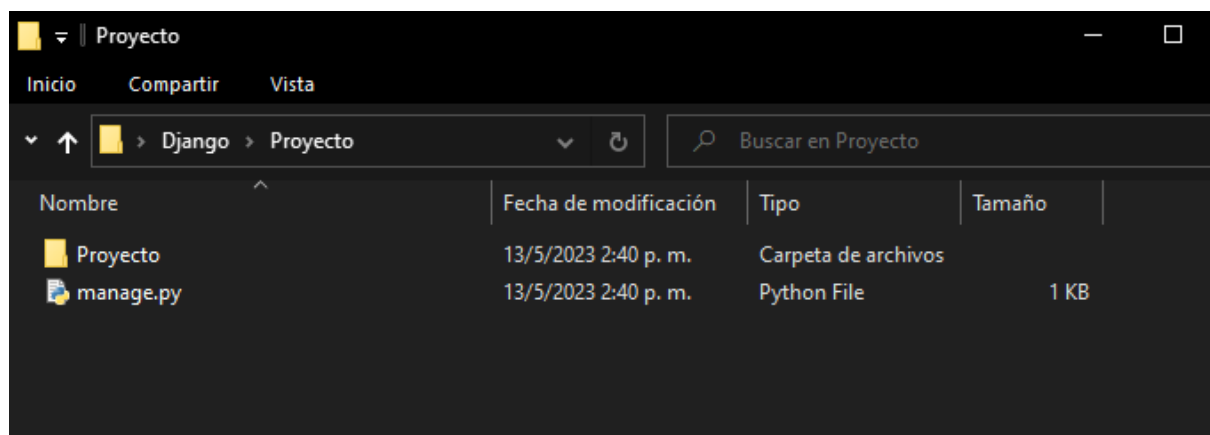
Una vez ahí debes introducir el siguiente comando sin las comillas “django-admin startproject nombre”. Debes reemplazar “nombre” con el nombre que quieras ponerle a tu proyecto. En este ejemplo lo llamaremos “**Proyecto**”:

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>cd C:\Users\Carlos Vick\Desktop\Django

C:\Users\Carlos Vick\Desktop\Django>django-admin startproject Proyecto
```

Este proceso tardará unos instantes y luego podrás ver que en la carpeta de tu proyecto se ha creado una carpeta con el nombre que le pusiste a tu proyecto y dentro de dicha carpeta habrán algunos archivos y carpetas necesarios que explicaremos luego. En nuestro ejemplo tenemos lo siguiente:



Y listo. Con esto ya tendríamos nuestro proyecto de Django creado.

Crear aplicaciones dentro de un proyecto

Puedes pensar que un proyecto y una aplicación son términos equivalentes pero por lo menos en Django no es así, ya que este framework hace una distinción entre estos conceptos. Una aplicación es algo que siempre estará dentro de un proyecto, y un proyecto puede tener una o muchas aplicaciones. En concreto una aplicación hará una tarea o un conjunto de tareas en específico dentro de un proyecto. Ejemplo, nuestro proyecto puede ser una “Tienda Online” y las aplicaciones dentro de este proyecto pueden ser “Ventas”, “Almacén”, “Clientes”, “carrito de compras”, “blog de novedades” etc .

Lo primero que debemos hacer para crear una aplicación en el proyecto que ya tenemos es entrar en la carpeta de dicho proyecto. Si aún tienes abierto el símbolo del sistema solo debes escribir “cd nombre”, donde “nombre” es como llamaste al proyecto. En mi este caso sería así:

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>cd C:\Users\Carlos Vick\Desktop\Django

C:\Users\Carlos Vick\Desktop\Django>django-admin startproject Proyecto

C:\Users\Carlos Vick\Desktop\Django>cd Proyecto_
```

Nota: En caso de que no tuvieras aún el símbolo del sistema abierto no hay problema, ábrelo con permisos de administrador y coloca la ruta completa como ya se explicó en pasos anteriores.

Luego el símbolo del sistema debe indicar que estas dentro de la carpeta del proyecto de esta manera:

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>cd C:\Users\Carlos Vick\Desktop\Django

C:\Users\Carlos Vick\Desktop\Django>django-admin startproject Proyecto

C:\Users\Carlos Vick\Desktop\Django>cd Proyecto

C:\Users\Carlos Vick\Desktop\Django\Proyecto>
```

Una vez ya dentro de la carpeta del proyecto debemos introducir el siguiente comando en el símbolo del sistema sin las comillas **“python manage.py startapp nombre”**. Debes reemplazar donde dice “nombre” y colocar como quieres que se llame tu aplicación. En este ejemplo llamaremos a nuestra aplicación “productos” ya que esto nos ayudará a ser más ilustrativos con los siguientes ejemplos.

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

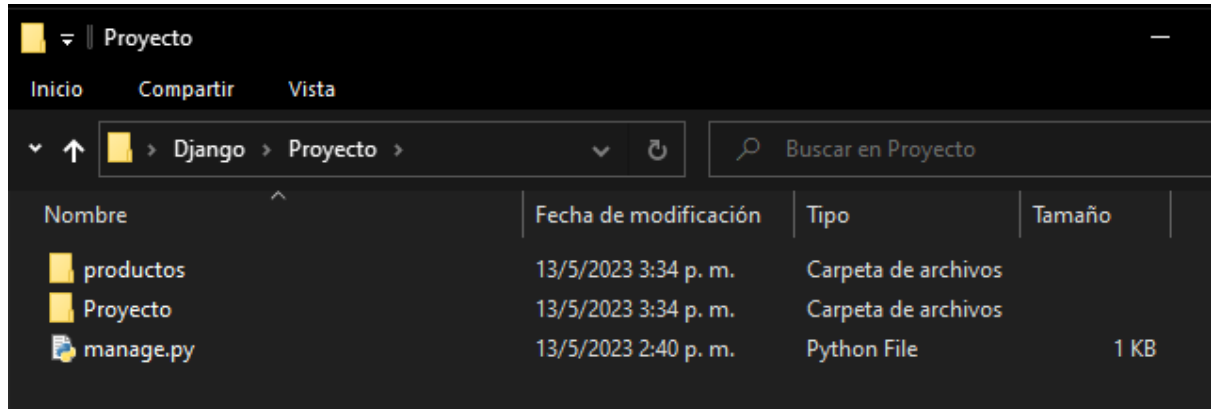
C:\Windows\system32>cd C:\Users\Carlos Vick\Desktop\Django

C:\Users\Carlos Vick\Desktop\Django>django-admin startproject Proyecto

C:\Users\Carlos Vick\Desktop\Django>cd Proyecto

C:\Users\Carlos Vick\Desktop\Django\Proyecto>python manage.py startapp productos_
```

El proceso tardará unos instantes y luego podrás visualizar dentro de la carpeta de tu proyecto que se ha creado una carpeta con el nombre que le has puesto a tu aplicación y como aquí:

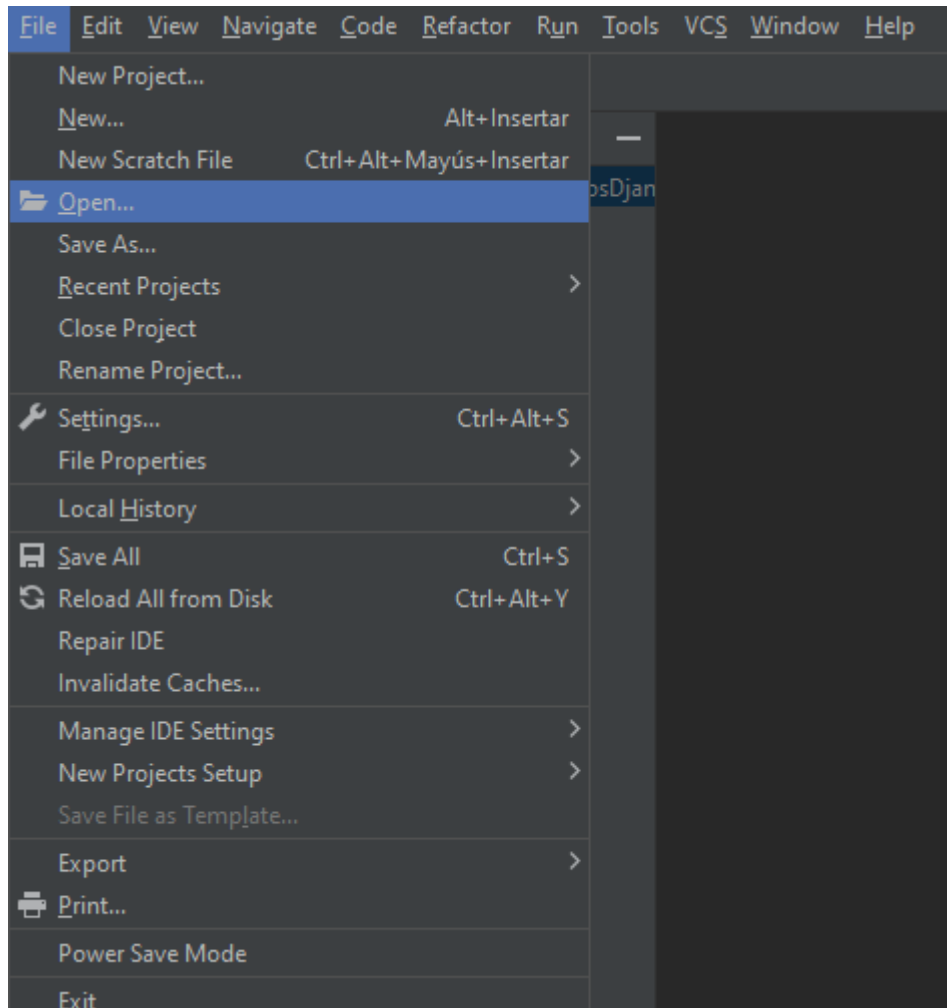


Dentro de esta carpeta con el nombre de tu aplicación hay varios archivos importantes que se explicarán más adelante.

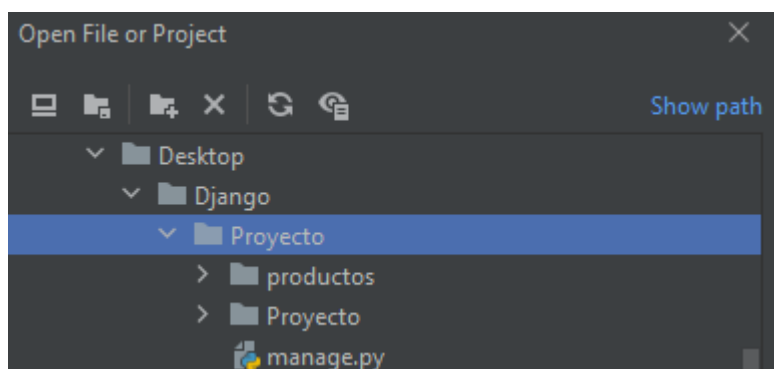
Y listo. Ya sabes como crear aplicaciones dentro de un proyecto en Django.

Bases de datos en Django

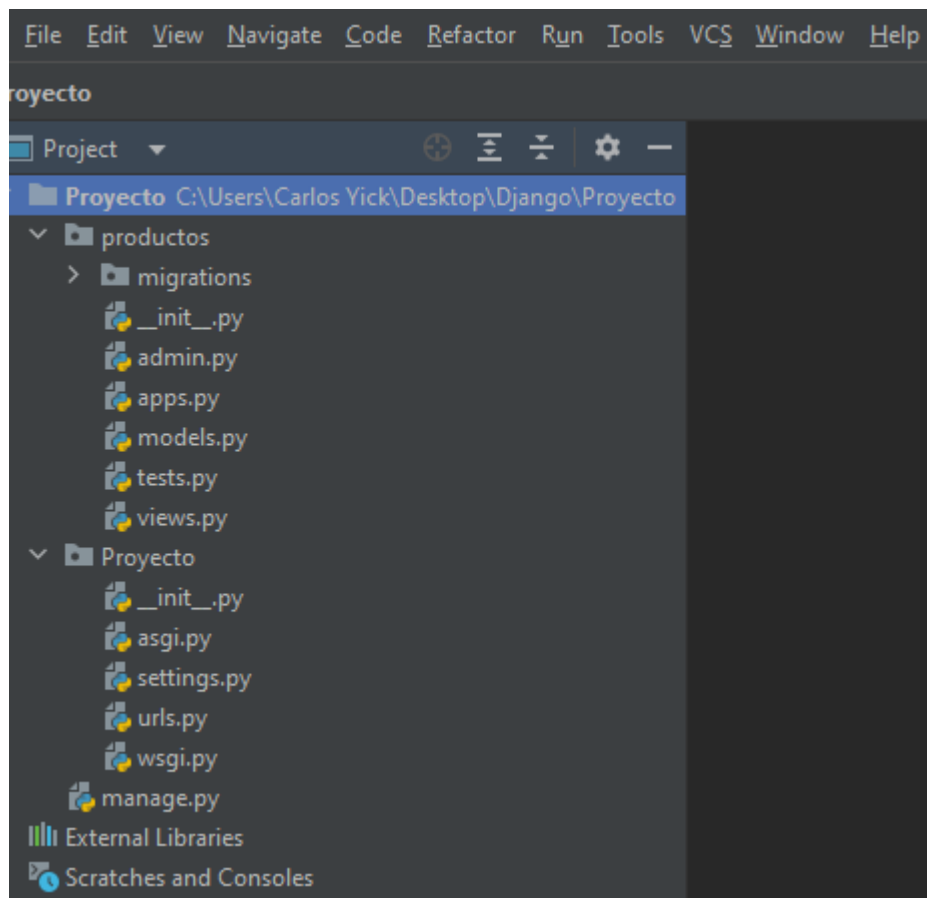
Para empezar a crear bases de datos en Django debemos abrir nuestro proyecto en nuestro compilador. En la mayoría de compiladores aparecerá la opción como “Open” así:



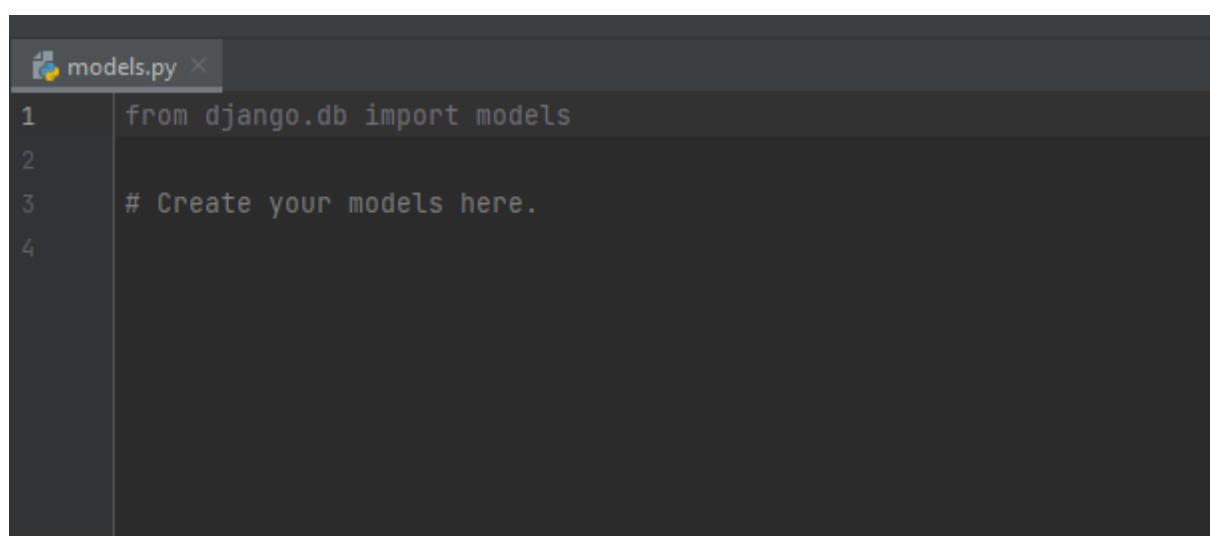
Después debemos de buscar la carpeta donde tenemos nuestro proyecto, en nuestro ejemplo sería así:



Damos Ok y nuestro proyecto ya se abrirá en nuestro compilador así:

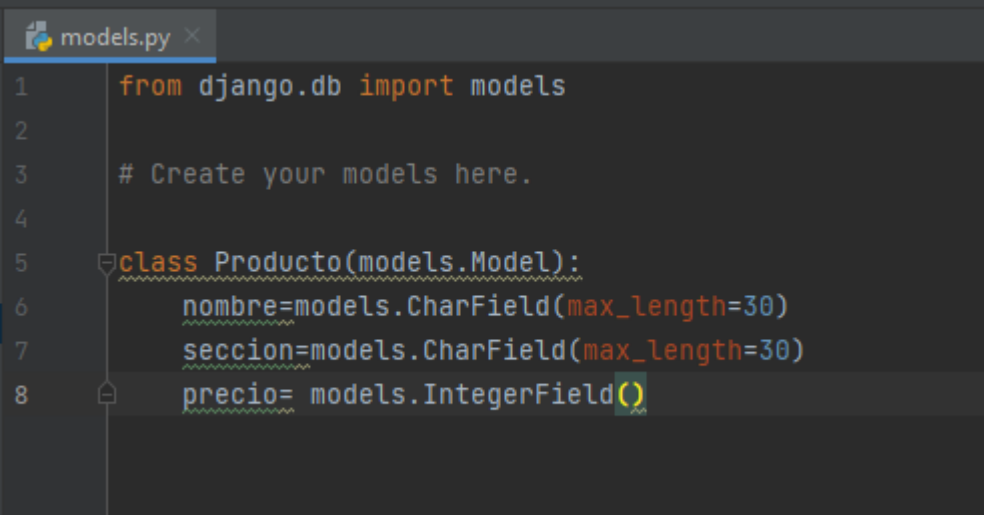


Ahora presta atención porque es fácil confundirse entre tantos archivos. Debes abrir el archivo “**models.py**” que está dentro de nuestra aplicación (en este caso recuerda que la nombramos “productos”) y te aparecerá esto en tu compilador:



Como puedes ver Django te explica para qué sirve este archivo en específico. Te dice que aquí debes crear tus modelos. Dentro de este archivo debes crear una clase por cada tabla

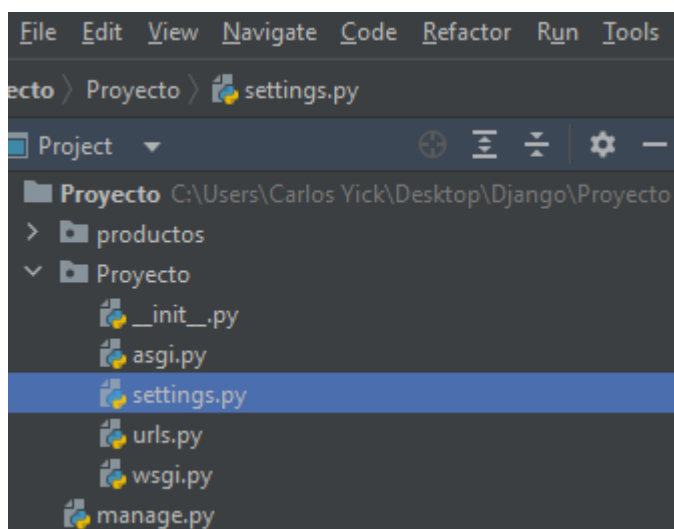
que quieras tener en tu base de datos. Este framework tiene la particularidad de que no se necesita hacer uso del lenguaje SQL para realizar operaciones en la base de datos, todo lo hará Django de forma automática. Entonces vamos a crear nuestra primera tabla a la que llamaremos “Producto” al igual que la aplicación que ya creamos.



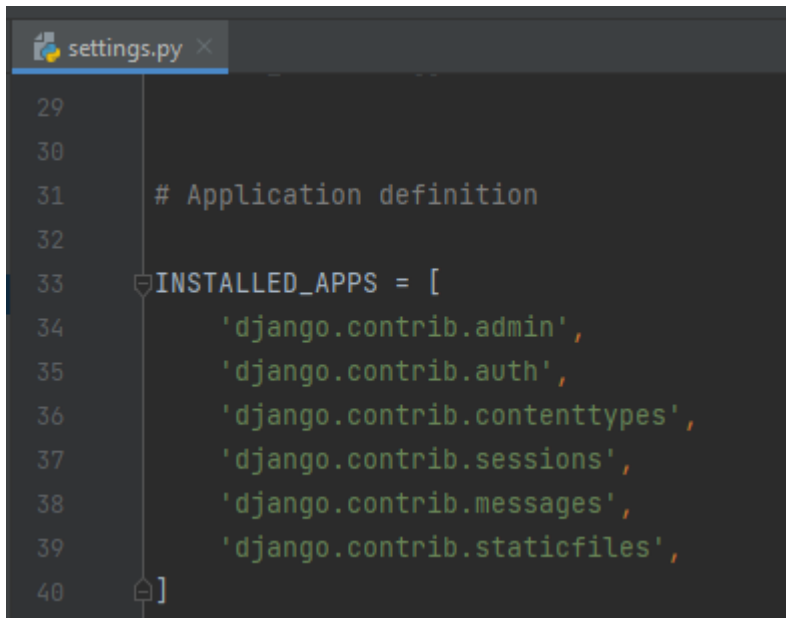
```
1 from django.db import models
2
3 # Create your models here.
4
5 class Producto(models.Model):
6     nombre=models.CharField(max_length=30)
7     seccion=models.CharField(max_length=30)
8     precio= models.IntegerField()
```

Las clases deben siempre recibir el atributo “models.Model”. Los atributos de esta clase corresponden a las columnas en nuestra tabla. En este ejemplo tenemos 3 columnas para un producto que son “nombre, sección y precio”. Siempre que declares un atributo debes agregar el “models.” y colocarle el tipo de dato de la columna(puede ser tipo texto, entero o cualquier tipo que necesites). También agregar que ese atributo en CharField que dice “max_length” significa que nuestro atributo de tipo texto tendrá máximo 30 caracteres.

Algo que debemos hacer es agregar esta aplicación al proyecto, porque a pesar de que ya está creada Django aún no lo sabe, para esto debemos irnos al archivo “settings.py” dentro de la subcarpeta con el nombre de nuestro proyecto. En nuestro ejemplo sería así:



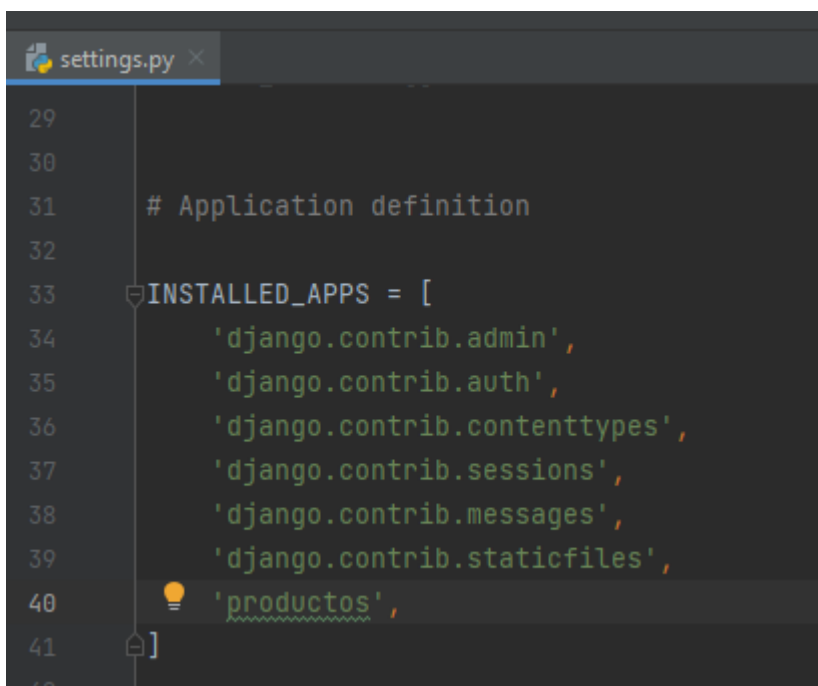
Dentro de este archivo hay bastante código, pero la parte que nos interesa es un apartado que dice “INSTALLED_APPS”, lo encontrarás así:



```
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
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 ]
```

De forma breve hay que decir que en esta lista están todas las aplicaciones que por defecto tiene un proyecto de Django. Siempre que creamos una aplicación debemos agregarla a esta lista.

Ahora debemos agregar nuestra aplicación a esta lista, para eso debemos colocar el nombre entre comillas simples y que no se nos olvide colocar la coma del final. Fíjese en el ejemplo con nuestra aplicación productos:



```
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
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     'productos',
41 ]
```

Listo ya agregamos nuestra aplicación y definimos las tablas de nuestra base de datos. Ahora el siguiente paso es crear dicha base de datos ya que esta aún no está creada.

Crear Base de Datos

Para crear nuestra base de datos debemos volver a utilizar el símbolo del sistema con permisos de administrador y dirigirnos a la carpeta de nuestro proyecto como ya hemos hecho anteriormente. Aquí te muestro la ruta de nuestro ejemplo:

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>cd C:\Users\Carlos Vick\Desktop\Django\Proyecto

C:\Users\Carlos Vick\Desktop\Django\Proyecto>_
```

Ahora dentro de la carpeta de nuestro proyecto debemos introducir el siguiente comando sin las comillas **“python manage.py makemigrations”** y eso nos creará la base de datos. Así debería verse una vez ejecutado el comando:

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>cd C:\Users\Carlos Vick\Desktop\Django\Proyecto

C:\Users\Carlos Vick\Desktop\Django\Proyecto>python manage.py makemigrations
Migrations for 'productos':
  productos\migrations\0001_initial.py
    - Create model Producto

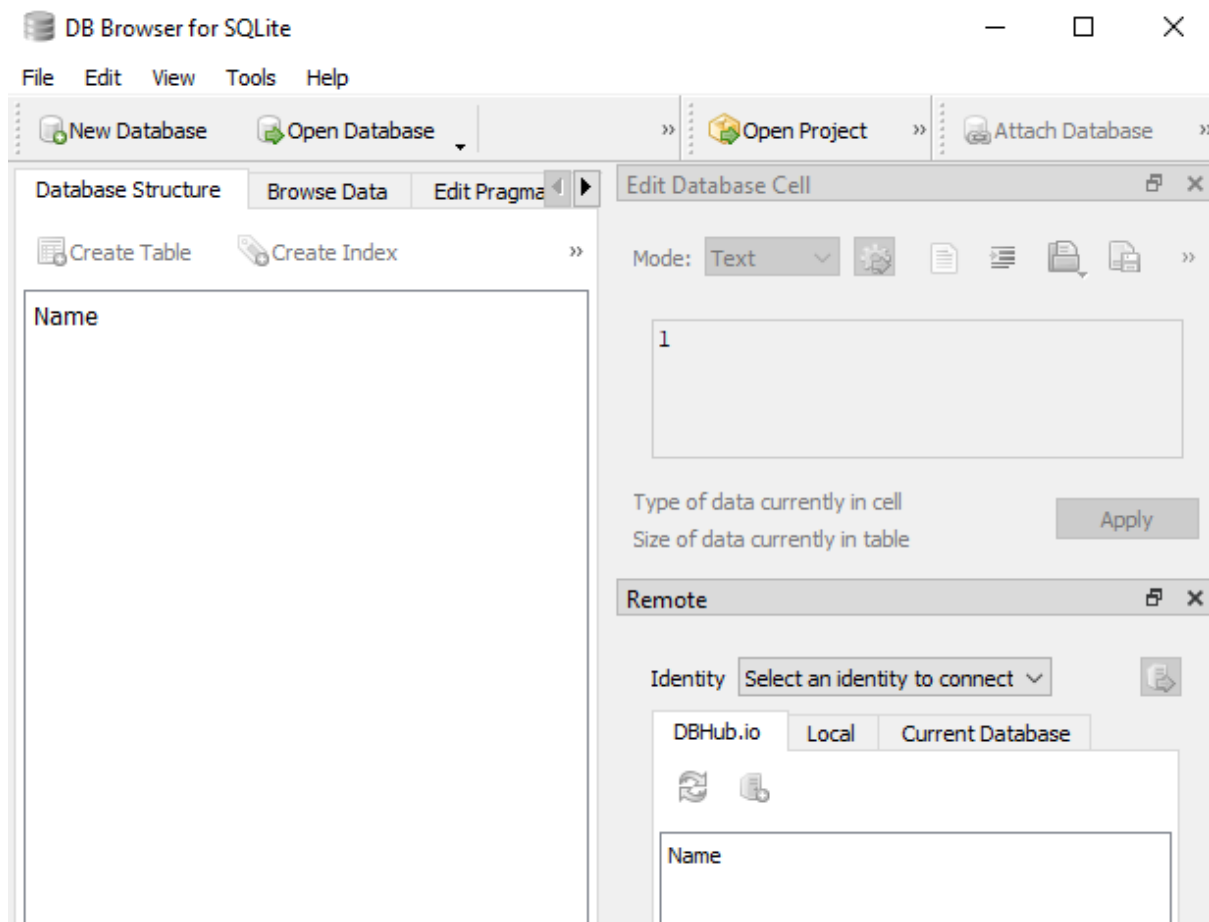
C:\Users\Carlos Vick\Desktop\Django\Proyecto>
```

Ahora si entras en la carpeta de tu proyecto podrás ver que se ha creado la base de datos así está en nuestro ejemplo:

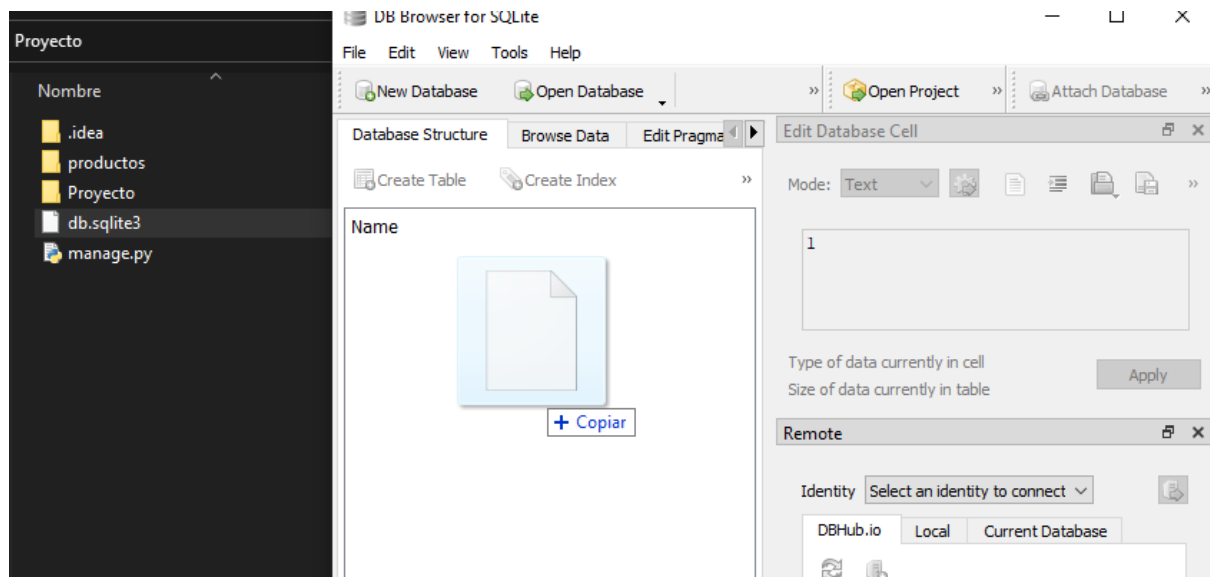
Proyecto >				
Nombre	Fecha de modificación	Tipo	Tamaño	
.idea	15/5/2023 1:46 p. m.	Carpeta de archivos		
productos	15/5/2023 2:18 p. m.	Carpeta de archivos		
Proyecto	15/5/2023 2:07 p. m.	Carpeta de archivos		
db.sqlite3	15/5/2023 2:18 p. m.	Archivo SQLITE3	0 KB	←
manage.py	13/5/2023 2:40 p. m.	Python File	1 KB	

Para este ejemplo usaremos el gestor SQLite3 para las base de datos ya que es el gestor predeterminado de Django.

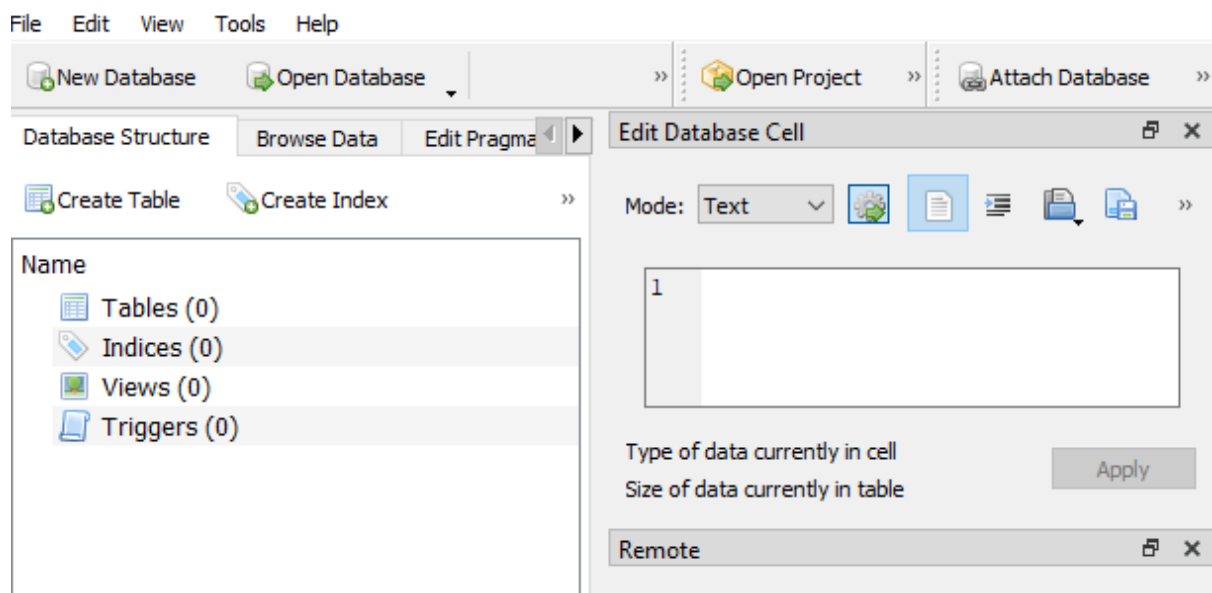
Si te das cuenta la base de datos está vacía, y eso es porque fue creada pero no aún no hemos insertado ninguna tabla o información dentro de esta. Podemos ver que no hay nada dentro de esta con un programa muy útil que recomendamos llamado “DB Browser for SQLite” cuyo link de descarga es: <https://sqlitebrowser.org/> y que es muy fácil de instalar y que nos permitirá ver de forma visual a la base de datos aunque no es necesario. Una vez abierto el programa aparecerá algo así:



Y para ver la base de datos basta con arrastrarla hasta el programa de esta manera:



Una vez hecho esto podrás ver la base de datos muy fácilmente:



Como puedes ver la base de datos sí que está vacía, pero ahora solucionaremos eso. Si aún tienes el símbolo del sistema abierto con la ruta a la carpeta del proyecto introduce este comando **"python manage.py migrate"**. Si ya lo habías cerrado simplemente vuelve a escribir la ruta a la carpeta del proyecto como ya se te ha enseñado. Una vez ejecuta el comando debe aparecerte algo como esto en el símbolo del sistema:

```
Administrador: Símbolo del sistema

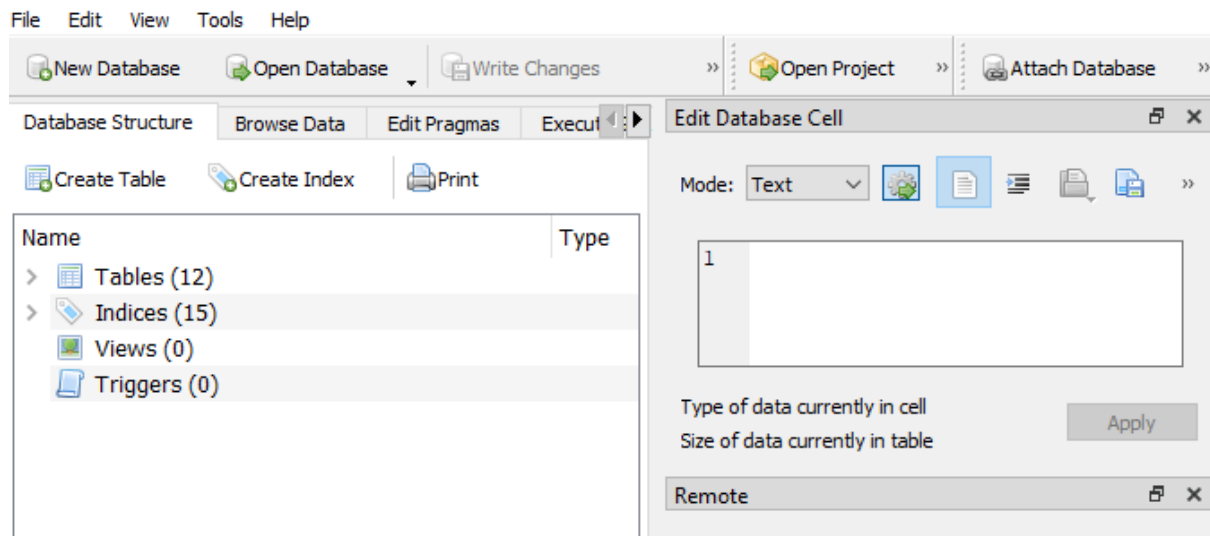
C:\Users\Carlos Vick\Desktop\Django\Proyecto>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, productos, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... 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 productos.0001_initial... OK
  Applying sessions.0001_initial... OK

C:\Users\Carlos Vick\Desktop\Django\Proyecto>
```

Ahora si ves la base de datos te darás cuenta que ya tiene algo de información:

Proyecto			
Nombre	Fecha de modificación	Tipo	Tamaño
.idea	15/5/2023 1:46 p. m.	Carpeta de archivos	
productos	15/5/2023 2:18 p. m.	Carpeta de archivos	
Proyecto	15/5/2023 2:07 p. m.	Carpeta de archivos	
db.sqlite3	15/5/2023 2:35 p. m.	Archivo SQLITE3	132 KB
manage.py	13/5/2023 2:40 p. m.	Python File	1 KB

Si vuelves a abrir el DB Browser for SQLite y arrastrar nuevamente la base de datos podrás ver que ya tiene las tablas que creaste:



Como puedes ver ahora si hay tablas dentro de nuestra base de datos. Pero como podrás darte cuenta hay 12 tablas y nosotros creamos tan solo 1 tabla. Esto es porque Django crea unas tablas de forma predeterminada que son fundamentales para el framework. Pero dentro de esas tablas que ves ahí está la que hemos creado en pasos anteriores.

El Panel de Administración

Django nos ofrece un panel de administración integrado, que nos permite hacer operaciones con la base de datos de forma sencilla. Pero para acceder a dicho panel de administración debemos crear lo que se conoce como un “SuperUsuario”. Esta se hace de forma muy sencilla, debes nuevamente abrir el símbolo del sistema y dirigirte a la carpeta del proyecto, como ya hemos hecho un varias veces en esta guía y ejecutar el siguiente comando “**python manage.py createsuperuser**” así:

```
Administrador: Símbolo del sistema - python manage.py createsuperuser

C:\Users\Carlos Yick\Desktop\Django\Proyecto>python manage.py createsuperuser
Username (leave blank to use 'carlosyick'): _
```

Empezará a pedirte varios datos como un username, correo electrónico y una contraseña. Por cuestiones de seguridad no mostraremos esos pasos. Pero son muy intuitivos y no tienen ninguna complicación.

Una vez tengas ya tu SuperUsuario creado ya puedes entrar al panel de administración que proporciona Django. Para eso primero debes correr el servidor, debes estar como de costumbre en la carpeta de tu proyecto y ejecutar este comando “**python manage.py runserver**” y si hiciste todo bien te saldrá esto:

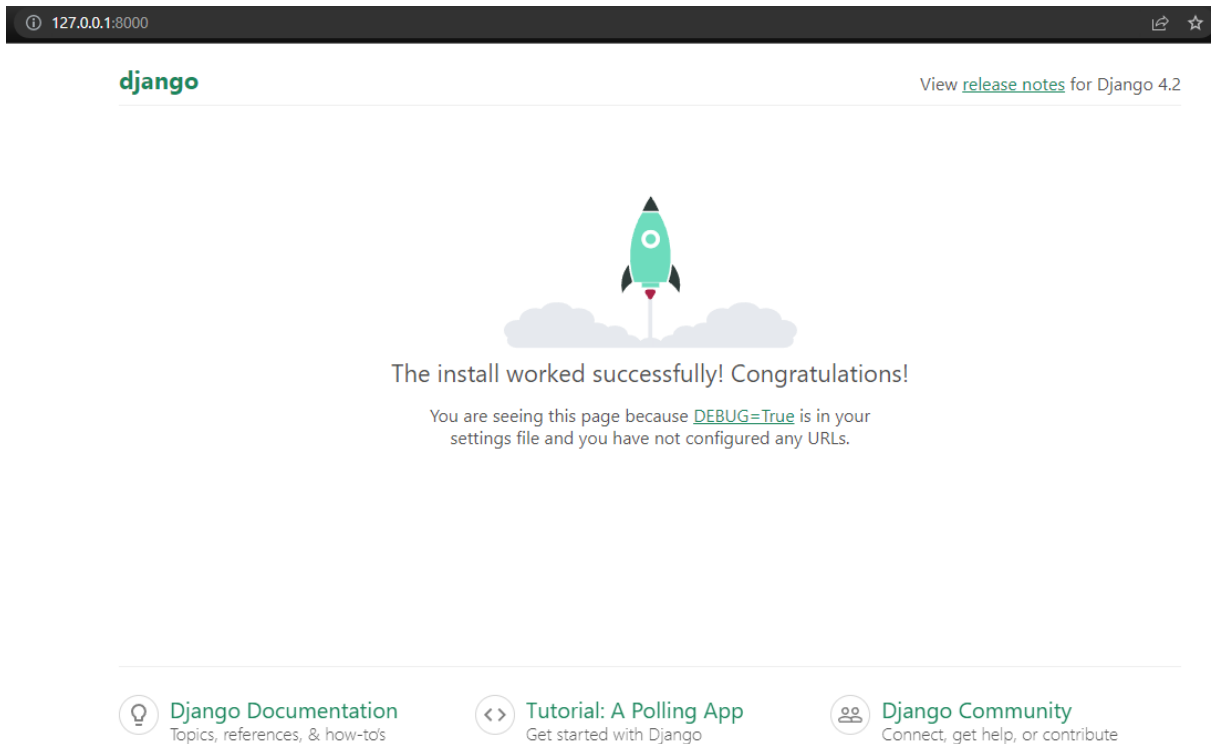
```
Administrador: Símbolo del sistema - python manage.py runserver

C:\Users\Carlos Yick\Desktop\Django\Proyecto>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

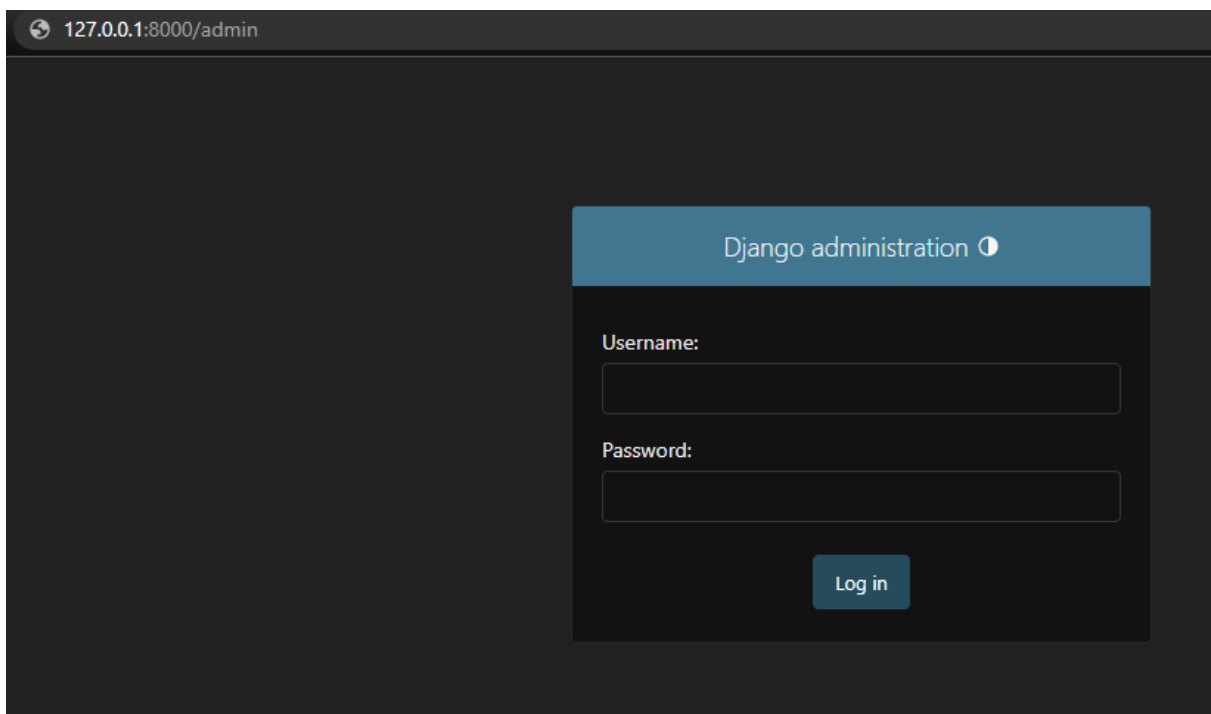
System check identified no issues (0 silenced).
May 15, 2023 - 15:05:26
Django version 4.2.1, using settings 'Proyecto.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Nota: no puedes cerrar el símbolo del sistema ya que ahora está ejecutando el servidor.

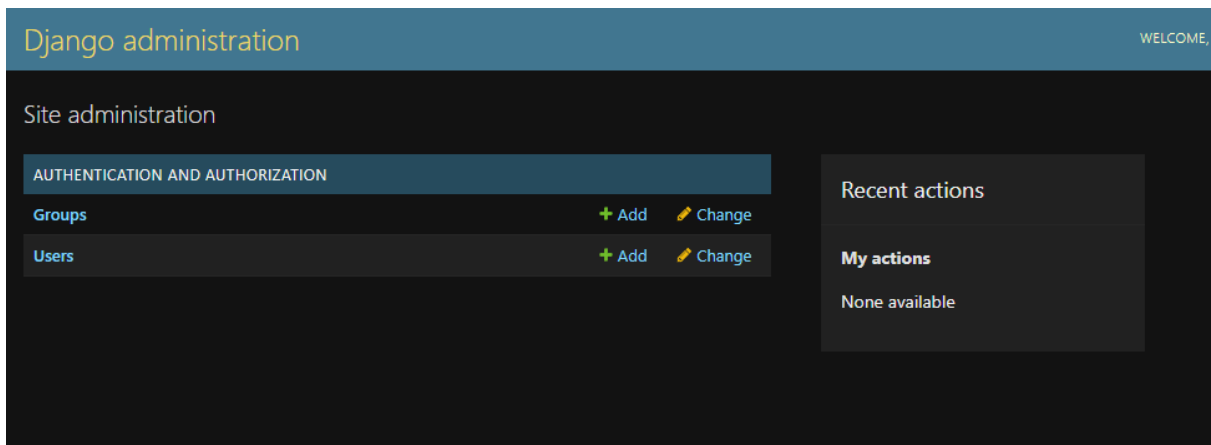
Puedes ver la línea que dice “**Starting development server...**”? En esa línea está la dirección que debes introducir en el navegador para ver cómo funciona el proyecto. Si has hecho todo bien hasta el momento debería salirte algo parecido a esto al meterla en tu navegador:



Felicidades, pero este no es el panel de administración. Para entrar al panel de administración debes agregar a esa dirección esto **“/admin”** y al introducir eso te debería ya salir algo así:



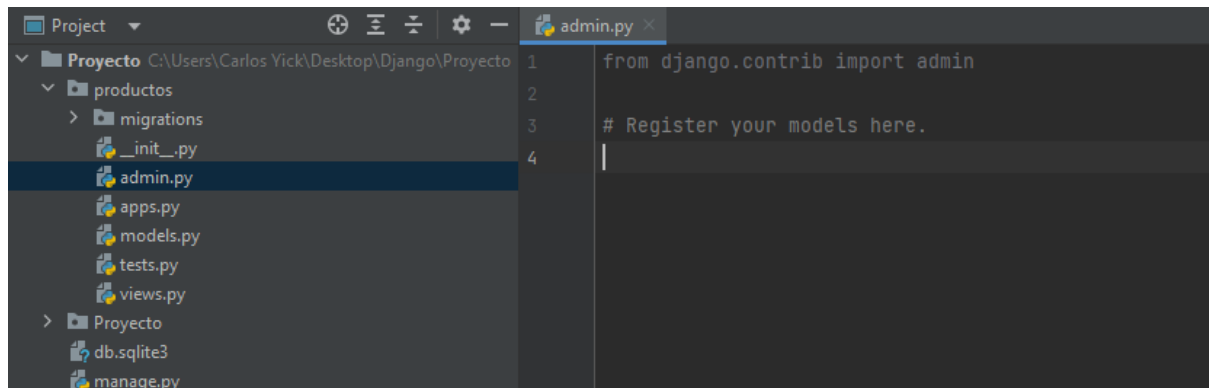
Aquí todo ya es fácil, solo introduce el super usuario que creaste y podrás al fin entrar al panel de administración. Te debería salir algo como esto:



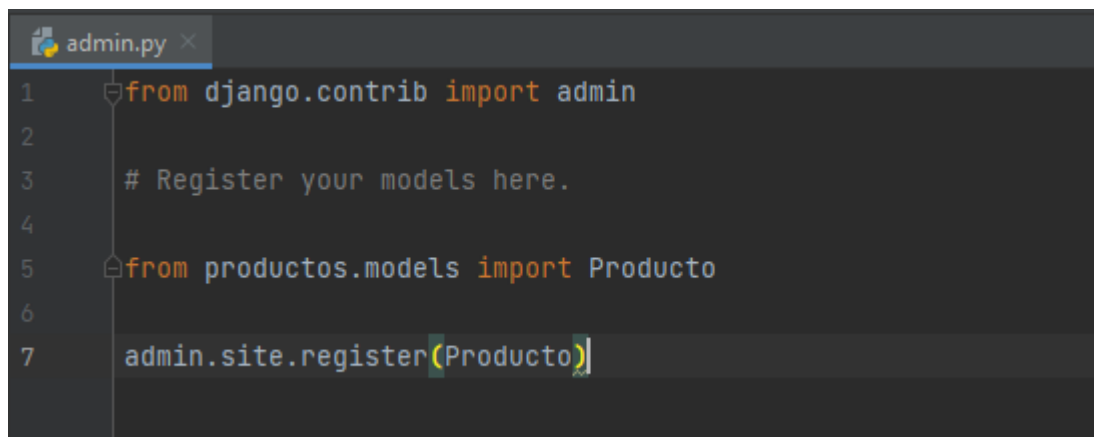
Felicidades, lograste entrar al panel de administración. Como puedes ver ya trae algunas funciones por defecto, como agregar más usuarios o grupos. En el siguiente apartado agregaremos la función para al fin poder realizar operaciones con la base de datos desde aquí.

Agregar Base de Datos al Panel de Administración

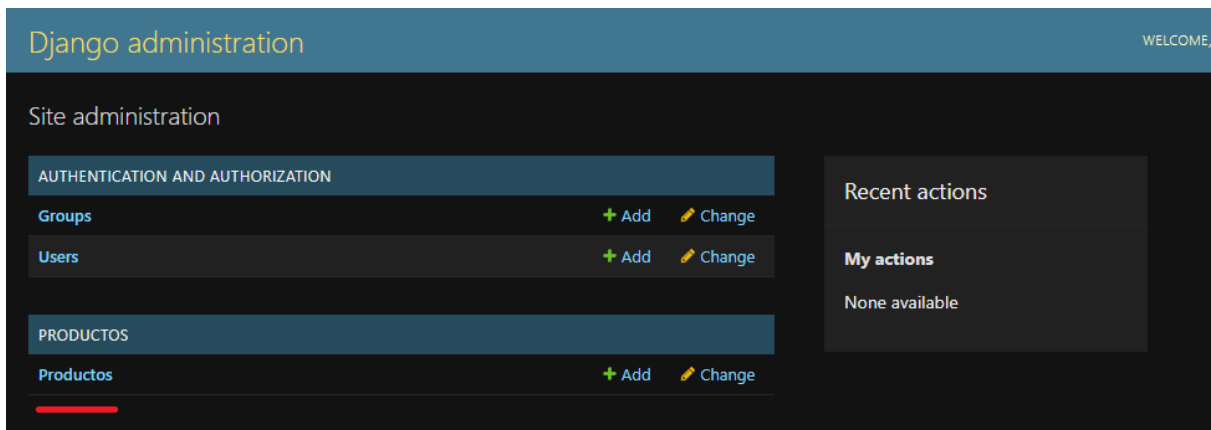
Para poder agregar nuestras tablas y así poder operar con ellas desde el panel de administración debemos dirigirnos al archivo “**admin.py**” que está dentro de la carpeta de nuestra aplicación cuyo nombre en este ejemplo es “**productos**”. Debemos hacerlo desde nuestro compilador así:



En el comentario nos dice que en este archivo debemos importar nuestros modelos. Entonces vamos a hacerlo de la siguiente manera:



Para que los cambios surtan efecto hay que cerrar y volver a arrancar el servidor. Para eso cerramos el símbolo del sistema si aún lo teníamos abierto y volvemos a introducir el comando “**python manage.py runserver**”. Luego volvemos a entrar en nuestro panel de administración de debería ya por fin aparecer nuestra tabla así:

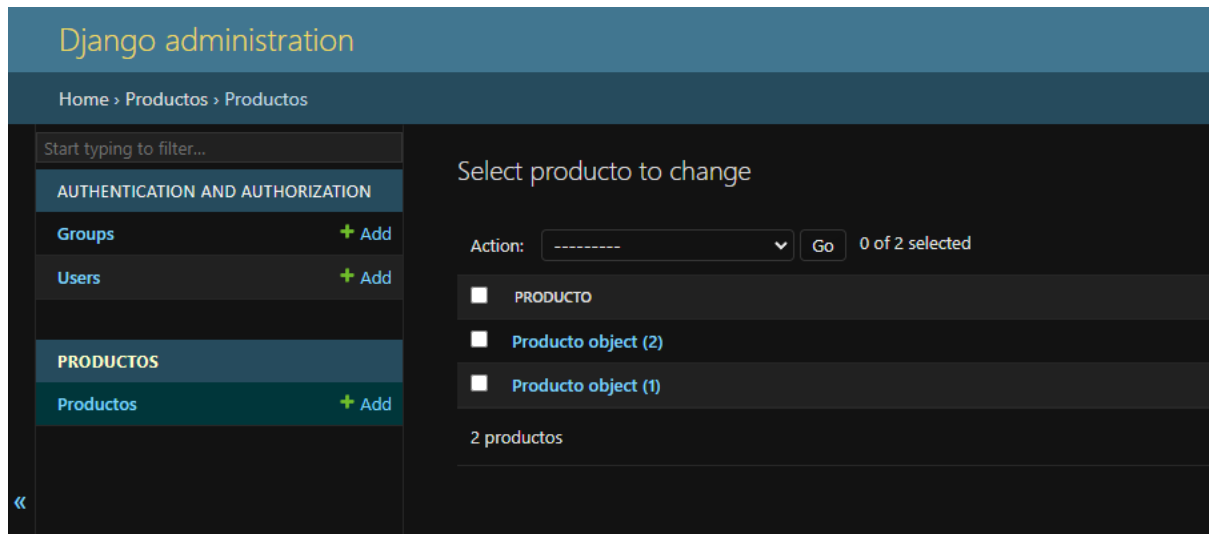


Fijate que nuestra tabla se llama “Producto” pero en el panel de administración nos aparece “Productos” esto es porque Django de forma predeterminada en el panel de administración agrega al final una “s” a cada tabla.

Bueno, vamos a agregar un par de productos. Primero hay que darle donde dice “+Add” y nos aparecerá esto para rellenar los campos:

Ejemplo de cómo se llenan los campos desde el panel de administración en Django

Ya que tenemos agregados un par de registros en la tabla nos aparecen así:

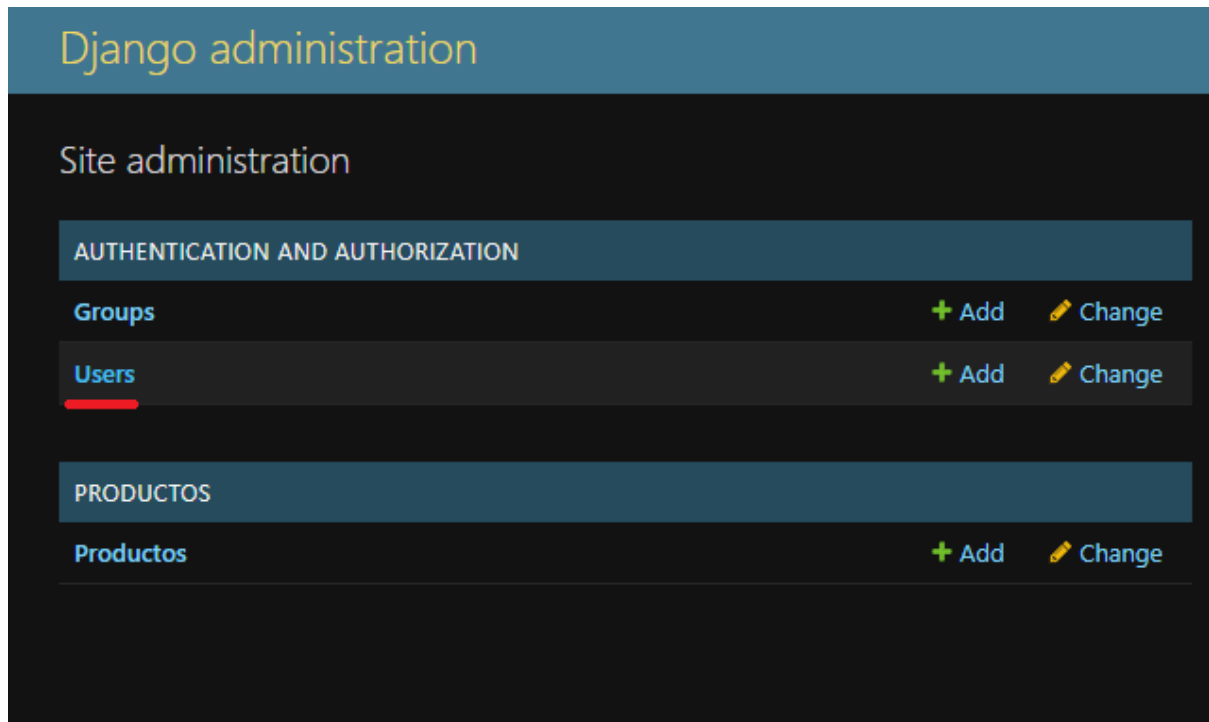


Si damos click sobre estos nos aparecerá toda la información que introducimos de cada uno

Y listo, ya podemos agregar, eliminar y modificar registros de una tabla desde el panel de administración.

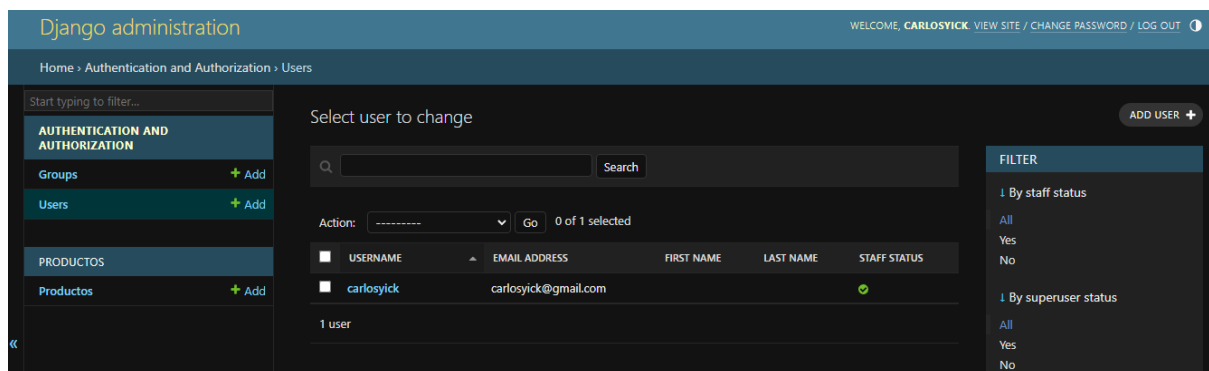
Usuarios

Como ya vimos hemos creado un “SuperUsuario” que es un tipo de usuario que tiene todos los permisos y privilegios. Ahora vamos a crear otros tipos de usuario que tengan menos permisos. Para hacer esto desde el panel de administración que nos ofrece Django tenemos que iniciar sesión con nuestro super usuario y entrar en la opción “Users”:



Una vez dentro nos damos cuenta que solo hay un usuario, que es el superusuario que creamos en pasos anteriores, también podemos ver que esta tabla tiene otros atributos que que Django configura por defecto como el atributo “**FIRST NAME**” y “**LAST NAME**”. Este superusuario tiene como ya dijimos todos los privilegios y una serie de atributos que vamos a describir a continuación:

“**STAFF STATUS**”: Este atributo indica que un usuario puede acceder al panel de administración. Si tiene un check como en la imagen es que puede entrar, si tiene una equis significa que no puede.



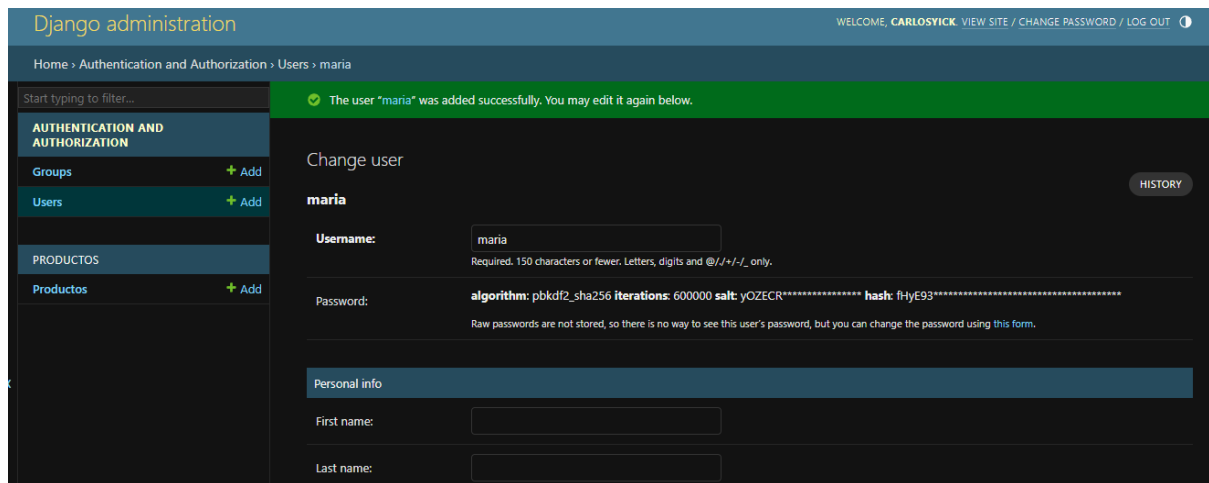
Ahora agreguemos un usuario que tenga menos privilegios, tal vez que no pueda añadir productos. Debemos darle en el botón **“ADD USER +”**

The screenshot shows the 'Select user to change' interface in Django Admin. At the top right is a button labeled 'ADD USER +'. Below it is a search bar with a magnifying glass icon and a 'Search' button. Under the search bar is an 'Action:' dropdown menu set to '-----' with a 'Go' button and a status '0 of 1 selected'. Below this is a table with columns: USERNAME, EMAIL ADDRESS, FIRST NAME, LAST NAME, and STAFF STATUS. One user is listed: 'carlosyick' with email 'carlosyick@gmail.com' and a green checkmark in the staff status column. At the bottom left of the table area, it says '1 user'. On the right side, there is a 'FILTER' sidebar with two sections: 'By staff status' and 'By superuser status', each with 'All', 'Yes', and 'No' options.

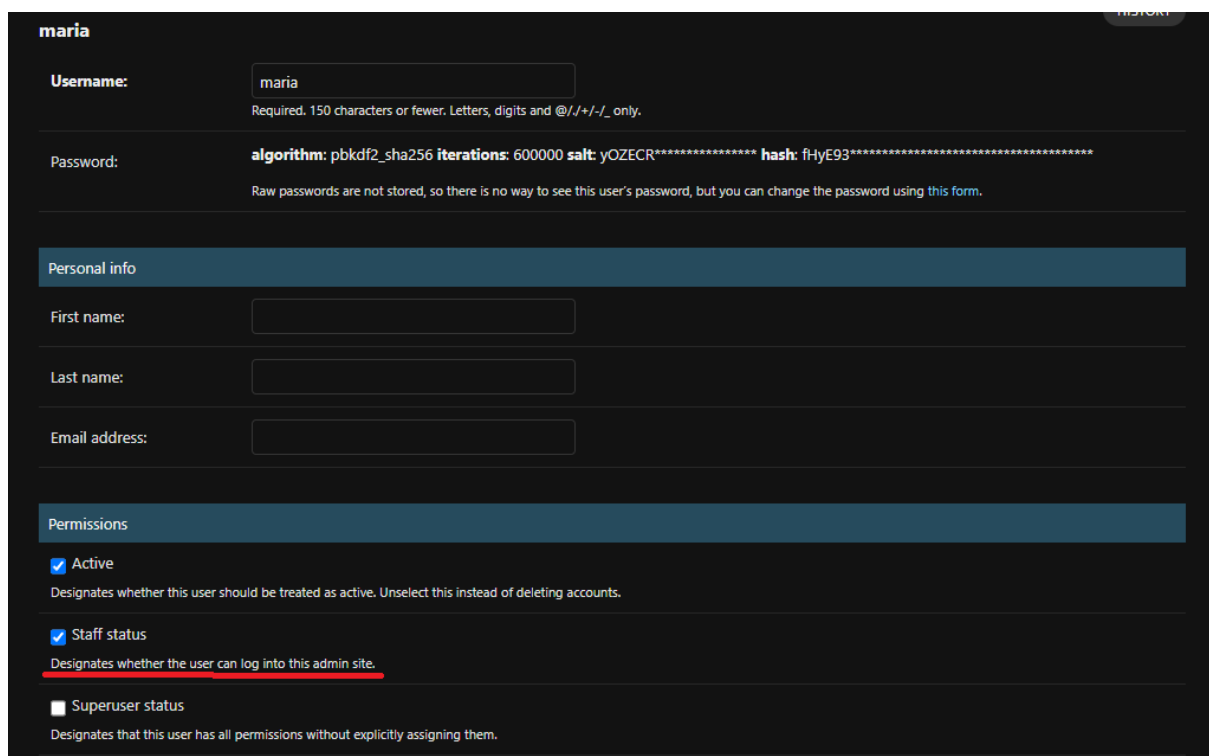
Después de pulsar dicho botón se abrirá una ventana de toda la vida para registrar un usuario. Para este ejemplo crearemos un usuario cuyo username es **“maria”** y le damos en **“SAVE”**:

The screenshot shows the 'Add user' form in Django Admin. The breadcrumb trail at the top reads 'Home > Authentication and Authorization > Users > Add user'. On the left is a sidebar with a search bar and two main sections: 'AUTHENTICATION AND AUTHORIZATION' containing 'Groups' and 'Users' (both with '+ Add' links), and 'PRODUCTOS' containing 'Productos' (with a '+ Add' link). The main content area is titled 'Add user' and contains the text: 'First, enter a username and password. Then, you'll be able to edit more user options.' There are three input fields: 'Username:' with the value 'maria' and a note 'Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.'; 'Password:' with masked characters and four notes about password requirements; and 'Password confirmation:' with masked characters and a note 'Enter the same password as before, for verification.' At the bottom are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

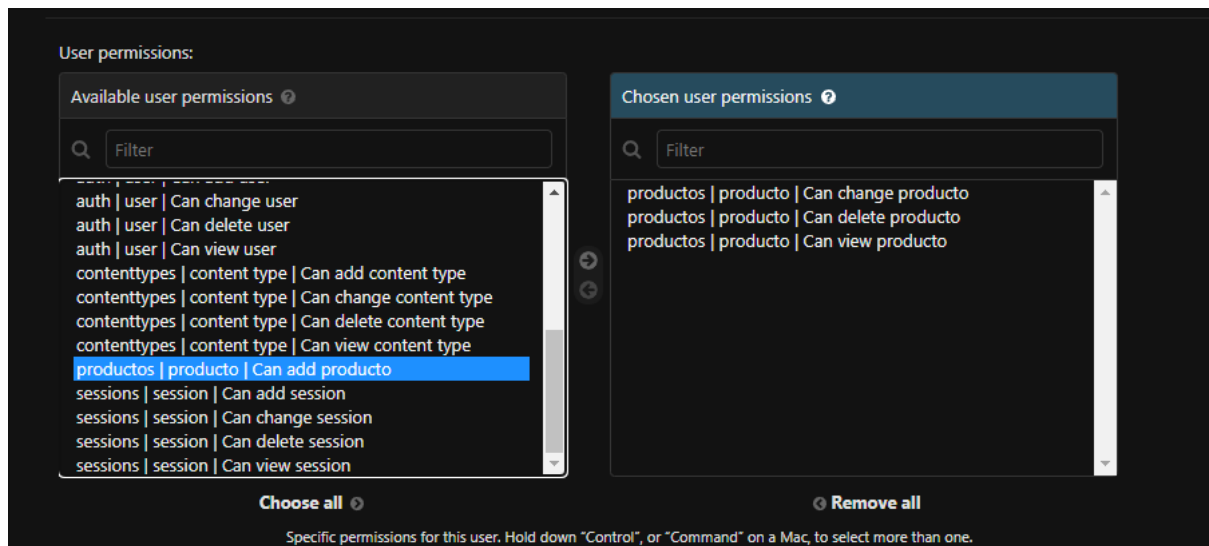
Y listo ya creamos el usuario “**maria**”, como puedes ver tiene otros campos pero no los llenaremos porque no son obligatorios:



Lo que sí haremos es ir más abajo y pulsar el checkbox “**Staff status**” para poder entrar al panel de administración así:

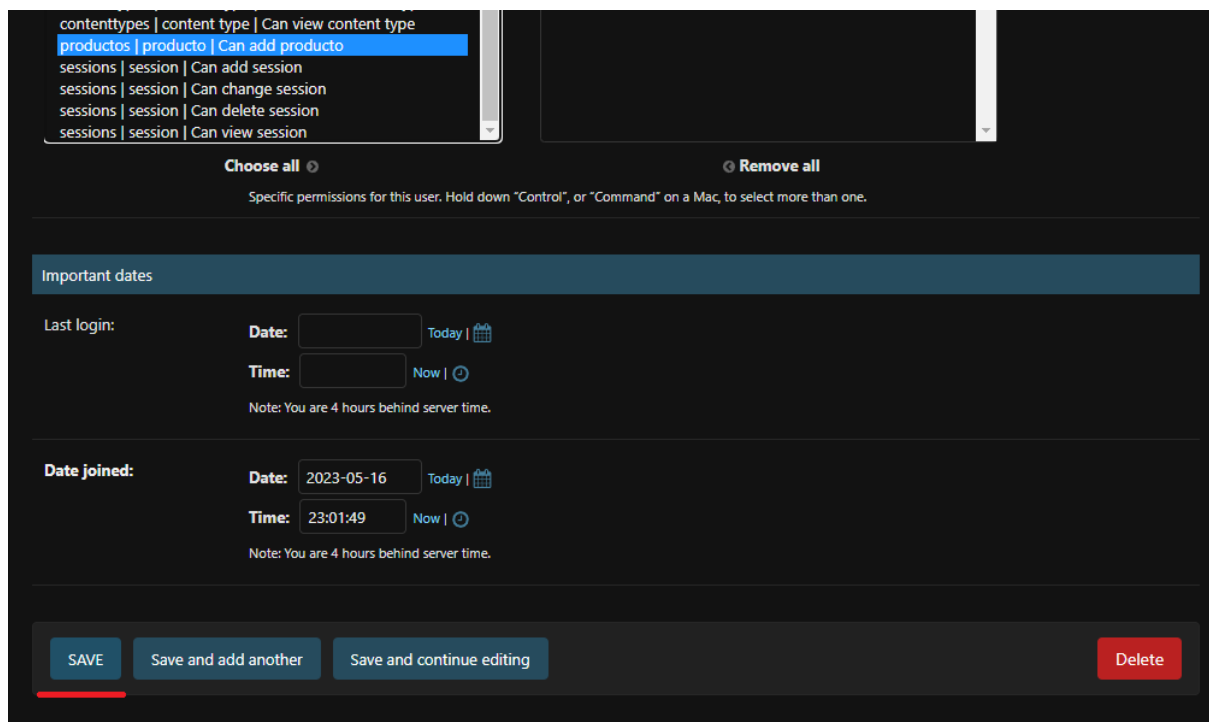


Y también iremos más abajo a la sección de “**Permissions**”, específicamente al apartado “User permissions” y buscamos los permisos que queremos otorgarle al usuario. Vamos a hacer que este usuario no tenga la opción de agregar productos a la tabla. En este panel hay muchos permisos pero es fácil identificar cuales son los que nos interesan. Ya que tienen el nombre de la aplicación y el nombre de la tabla. Una vez los encontremos agregamos los permisos que nos interesan de esta manera:

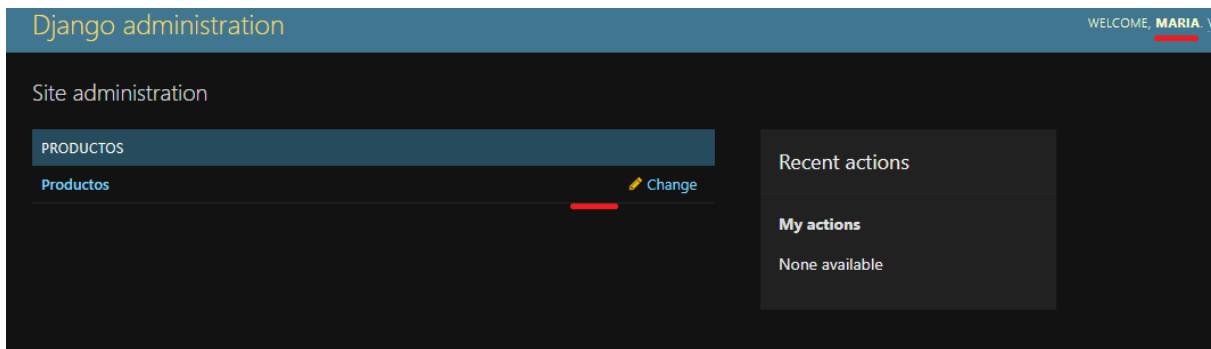


Como puedes ver, a este usuario se le están otorgando los permisos de ver, cambiar y eliminar elementos de la tabla. Pero no se le permite añadir.

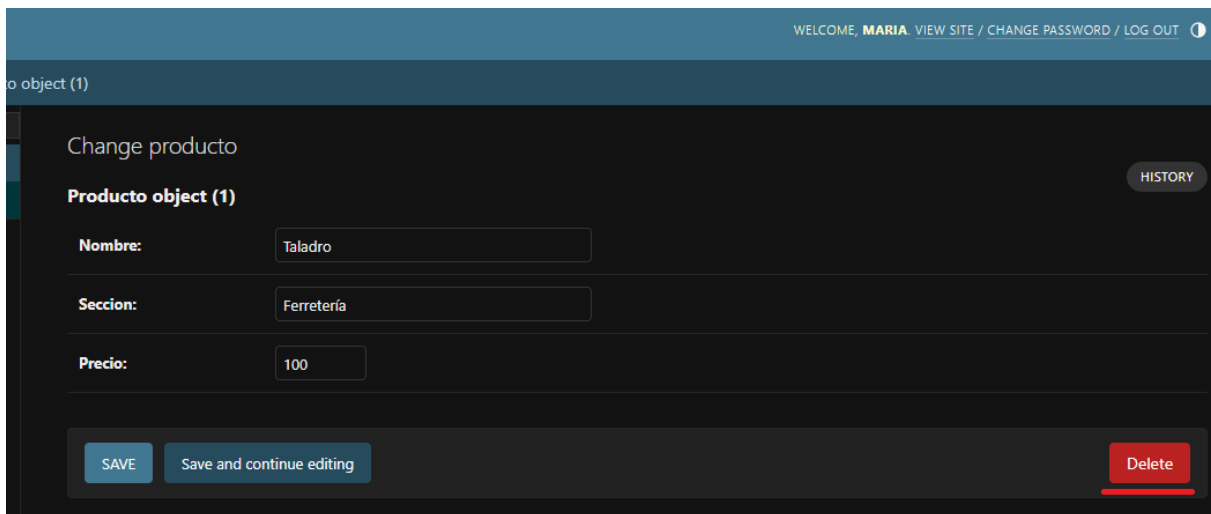
Luego vamos al botón **“SAVE”** para guardar todo y listo ya nuestro usuario con permisos restringidos está creado:



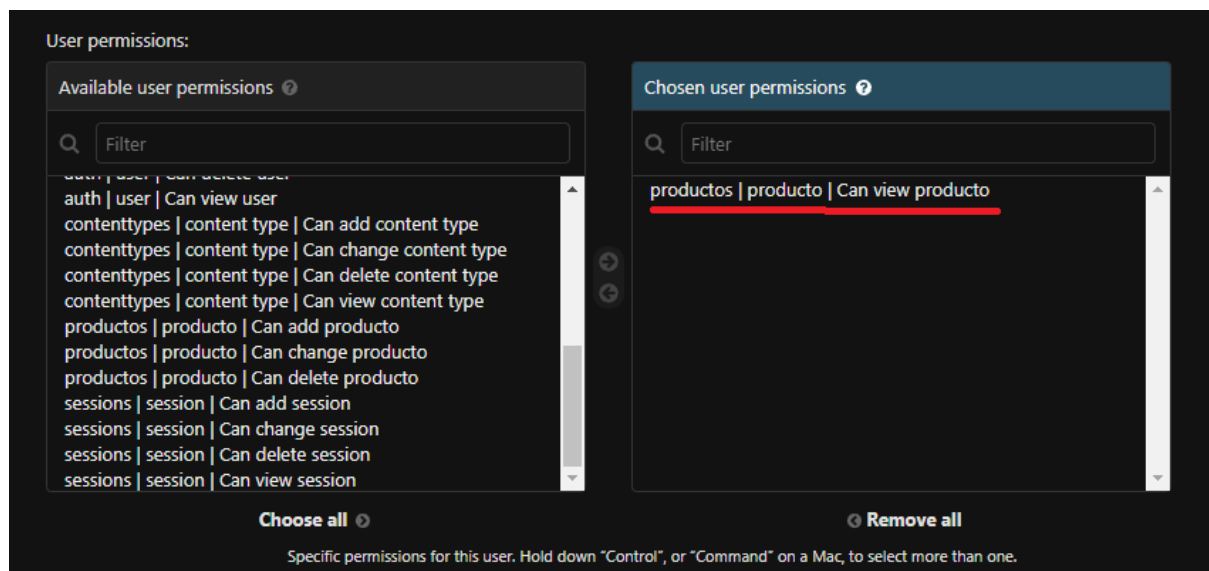
Listo, ya tenemos un nuevo usuario agregado con permisos limitados. Vamos ahora a cerrar sesión con nuestro superusuario y a iniciar sesión con el nuevo usuario que creamos. Como pueden ver entramos perfectamente en el panel de administración, pero ahora miren el detalle, no tenemos ni acceso a los usuarios ni a los grupos ni tampoco tenemos la opción “add” en productos y fíjense que si estamos desde el usuario **“maria”**:



Entremos a la tabla y veremos que si vemos un producto también tendremos la opción de eliminarlos:

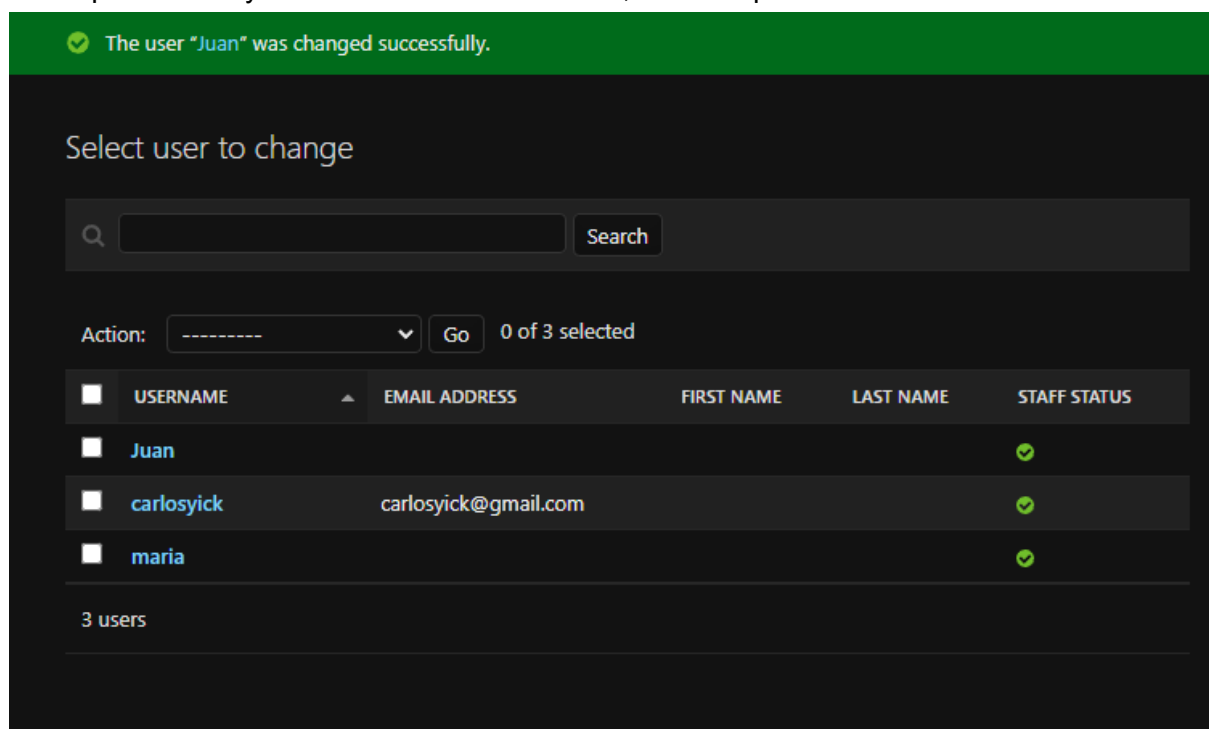


Y listo! ya viste cómo puedes crear un usuario con permisos limitados! Ahora juguemos un poco más, vamos a crear otro usuario pero que esta vez solo tenga un permiso. El proceso de crear este usuario lo omitiremos ya que se explicó anteriormente. Recordando que primero debes cerrar sesión con este usuario cuyos permisos son limitados e iniciar sesión con tu superusuario, también debes darle al check **“Staff Status”** para poder entrar al panel de administración. En este ejemplo llamaremos al usuario “Juan” y daremos **“SAVE”**:

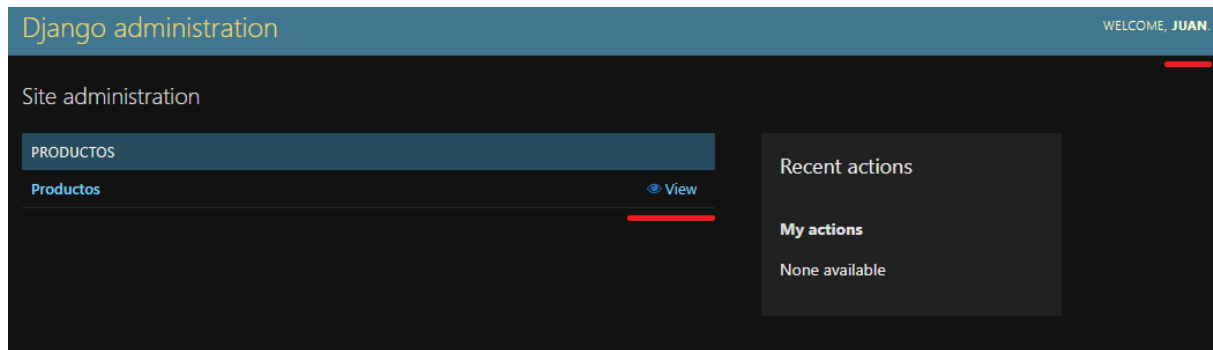


Como puedes ver solo le estamos permitiendo al usuario Juan ver la tabla productos. Mas no podrá cambiarla de ninguna forma

Como puedes ver ya creamos el usuario “Juan”, miralo aqui:

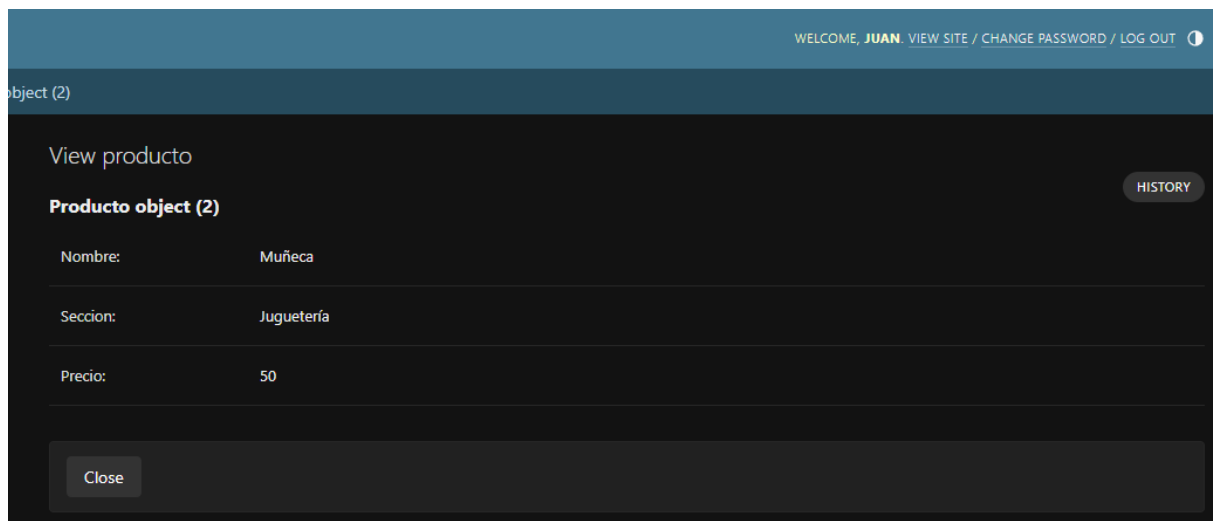


Ahora debemos cerrar sesión con nuestro superusuario e iniciar sesión con el nuevo usuario, vean lo que tenemos una vez iniciada la sesión:



Como puedes ver, cuando inicias sesión con “Juan” solo te permite ver la tabla de productos.

Si intentáramos entrar a la tabla no nos dejará realizar cambios en ella, ni eliminar, ni agregar, ni actualizar:



Fíjate que ya no aparece ninguna opción para modificar este registro.

Y listo, ya has aprendido a crear tu proyecto en Django, crear una aplicación, una base de datos y agregar diferentes privilegios de usuarios!

Feliz Codeo.

Material usado

Webgrafia

<https://www.youtube.com/watch?v=B8OIRdYwNIA&t=474s>

<https://docs.djangoproject.com/en/4.2/intro/>