

Creando los modelos

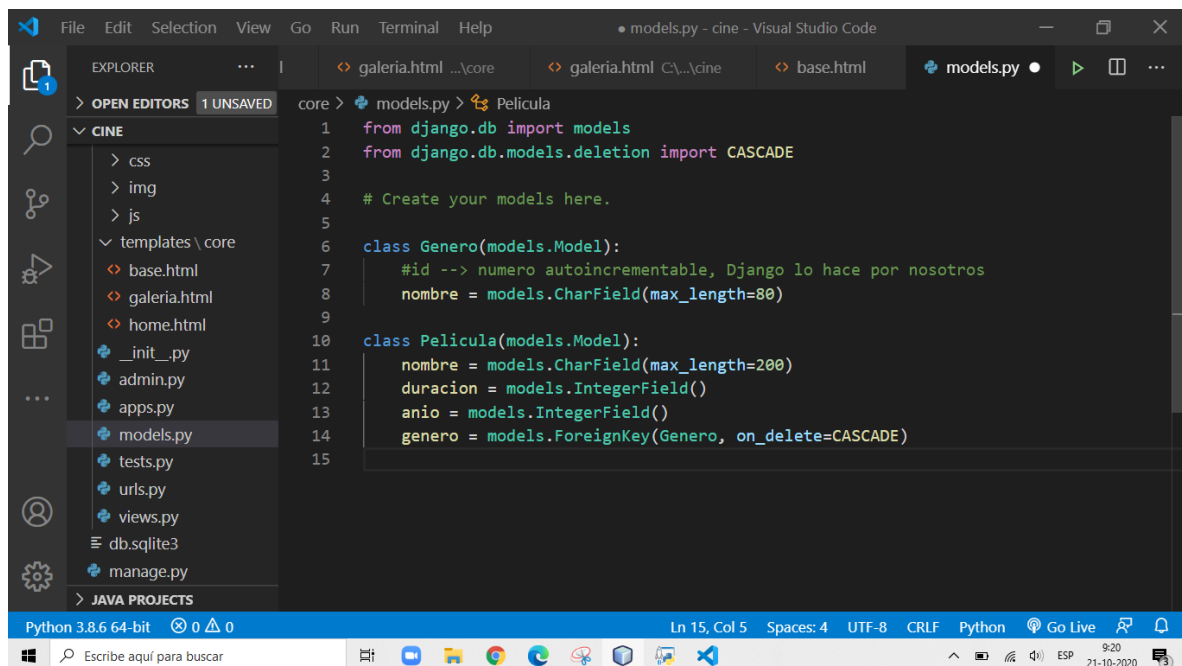
Modelos de migración

modelo me refiero ya a lo que es la estructura de los datos que vamos a manejar con la base de datos y algo tenemos varias maneras de manejar la base principal es hacerlo a partir de el código byte es decir hay tres fases para generar la tabla

escribir el modelo: `models {pelicula}` ---> estos pasan al archivo migration el cual es un conjunto de instrucciones que añade Python para pasar desde el modelo hacia la base de datos, es decir, agregar tabla, cambiar campo de una tabla, etc. Una vez realizado esto, se pasa de migración a la base de datos

Para hacer un modelo hay que identificar la tabla y sus atributos, en este caso haremos la tabla: película y sus atributos nombre, titulo, duración y que esta tenga un genero (comedia, terror, infantil) que estarán en otra tabla: genero. Una relación uno a muchos: un género, muchas películas.

Modificar archivo *core/models.py*



```
1 from django.db import models
2 from django.db.models.deletion import CASCADE
3
4 # Create your models here.
5
6 class Genero(models.Model):
7     #id --> numero autoincrementable, Django lo hace por nosotros
8     nombre = models.CharField(max_length=80)
9
10 class Pelicula(models.Model):
11     nombre = models.CharField(max_length=200)
12     duracion = models.IntegerField()
13     anio = models.IntegerField()
14     genero = models.ForeignKey(Genero, on_delete=CASCADE)
15
```

Ahora necesitamos que estos modelos pasen a la base de datos, en nuestro caso SQLite

Así que desde la consola ejecutamos el comando

Python manage.py makemigrations

Para que lea el models.py y cree el archivo con las instrucciones

The screenshot shows the Visual Studio Code interface with a Django project named 'core'. The Explorer sidebar on the left shows the project structure, including a 'migrations' directory. The main editor displays the 'models.py' file with the following code:

```
8 nombre = models.CharField(max_length=80)
9
10 class Pelicula(models.Model):
11     nombre = models.CharField(max_length=200)
12     duracion = models.IntegerField()
13     anio = models.IntegerField()
14     genero = models.ForeignKey(Genero, on_delete=CASCADE)
15
```

The Terminal at the bottom shows the command `python .\manage.py makemigrations` being executed, resulting in the creation of a new migration file: `core\migrations\0001_initial.py`. The migration file contains the following operations:

```
Migrations for 'core':
  core\migrations\0001_initial.py
    - Create model Genero
    - Create model Pelicula
PS C:\Django\cine>
```

Eso creará un nuevo archivo que estará en *migrations*

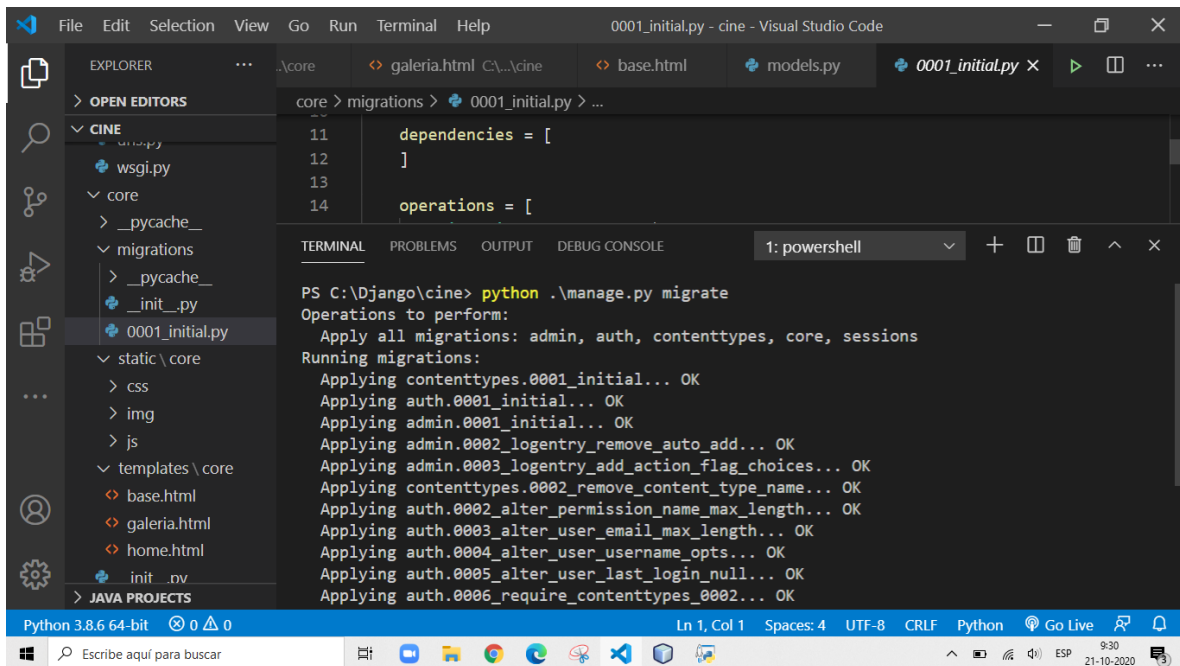
The screenshot shows the Visual Studio Code interface with the newly created migration file `0001_initial.py` open in the editor. The file contains the following code:

```
11 dependencies = [
12 ]
13
14 operations = [
15     migrations.CreateModel(
16         name='Genero',
17         fields=[
18             ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
19             ('nombre', models.CharField(max_length=80)),
20         ],
21     ),
22     migrations.CreateModel(
23         name='Pelicula',
24         fields=[
25             ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
26             ('nombre', models.CharField(max_length=200)),
27             ('duracion', models.IntegerField()),
28             ('anio', models.IntegerField()),
29             ('genero', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='core.genero')),
30         ],
31     ),
32 ]
```

Ahora necesitamos volcar estas instrucciones creadas a la base de datos

Ahora ejecutamos desde la consola el comando

Python manage.py migrate



The screenshot shows the Visual Studio Code interface with a Django project. The Explorer pane on the left shows the project structure, including the 'migrations' directory. The main editor shows the '0001_initial.py' migration file with the following code:

```
dependencies = [
]

operations = [
```

The Terminal pane at the bottom shows the command `python .\manage.py migrate` being executed, resulting in the following output:

```
PS C:\Django\cine> python .\manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, core, 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
```

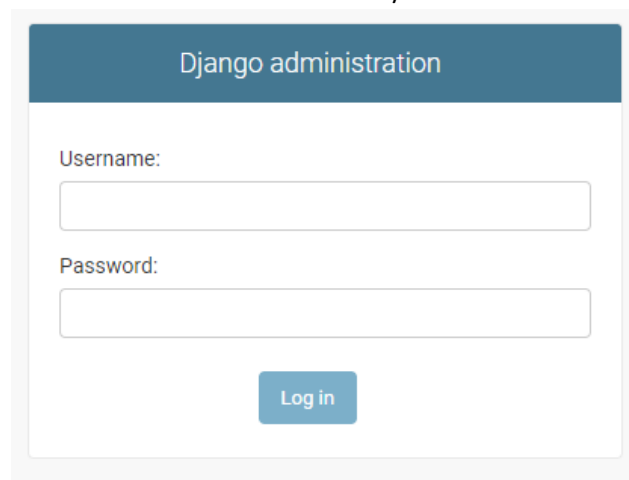
Esto crea en la base datos no solo las tablas Película y Genero, sino que además tablas para los usuarios, grupos y permisos de la base de datos, que sirven para la autenticación

Si quieres ver tu base de datos, puedes buscar en línea: sqliteonline.com

Pero ahora veamos desde nuestro Django

Nos aseguramos de tener el servidor de nuestra aplicación arriba: *Python manage.py runserver*

Y desde el navegador colocamos la ruta `127.0.0.1:8000/admin`



The screenshot shows the Django administration login page. It has a blue header with the text 'Django administration'. Below the header, there are two input fields: 'Username:' and 'Password:'. At the bottom, there is a blue button labeled 'Log in'.

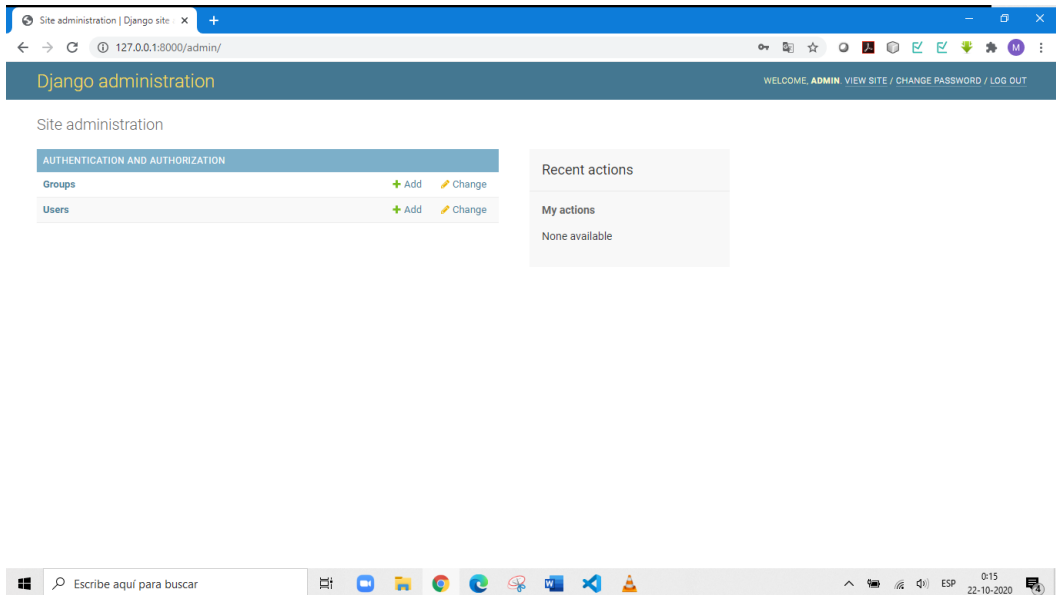
Para poder ingresar a este login necesitamos tener un usuario admin, el que se crea mediante terminal

`python .\manage.py createsuperuser`

Te pedirá los datos de user, correo electrónico para recuperación y password, en mi caso admin admin

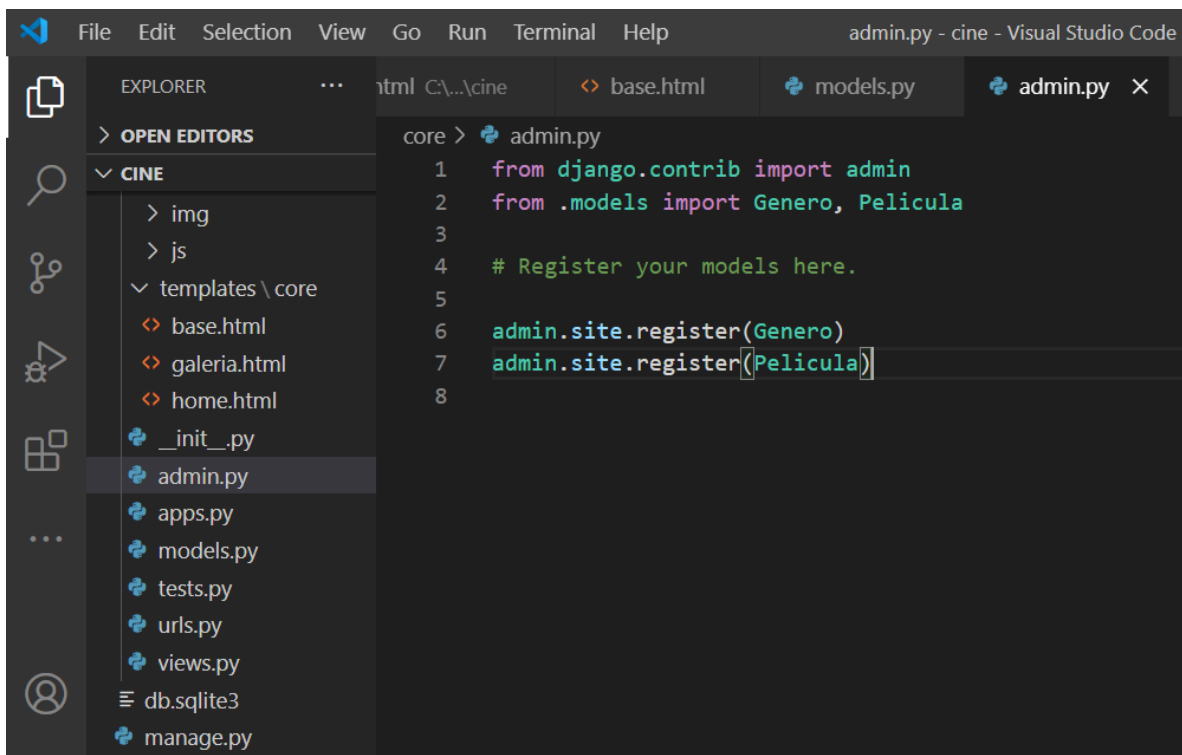
Ahora probemos si resulta

Vamos a la ventana de admin con los datos recién creados



Para que se puedan ver nuestros modelos en este módulo de administrador, vamos a editar el archivo *core/admin.py*

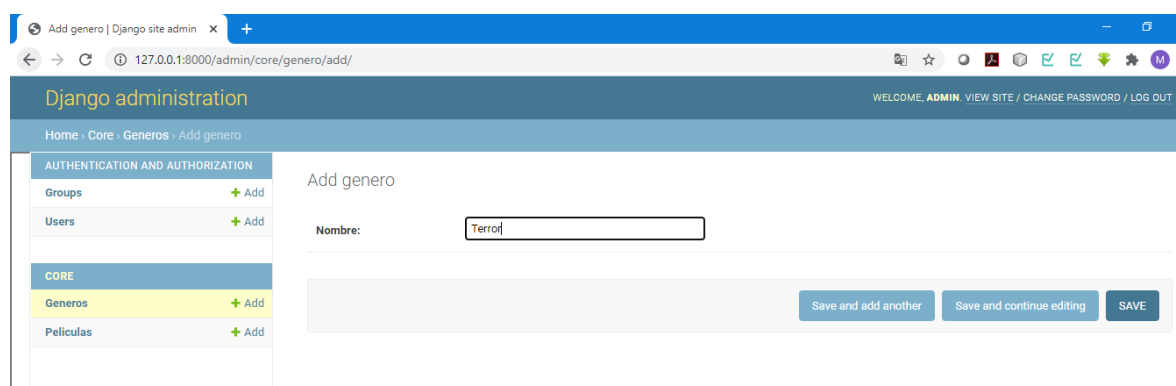
Para importar y registrar los modelos de Genero y Película



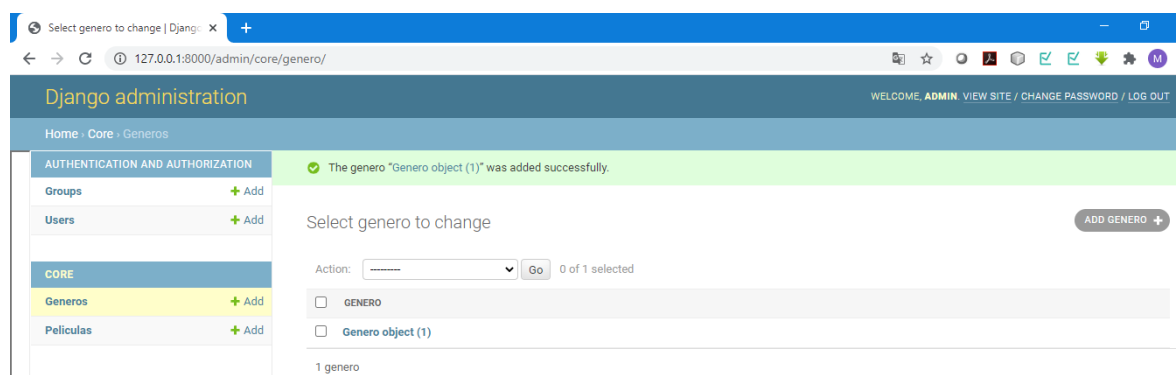
Al grabar, observa tu módulo de administración



Comienza a aparecer los modelos que tenemos en core, donde ya se puede agregar y gestionar.

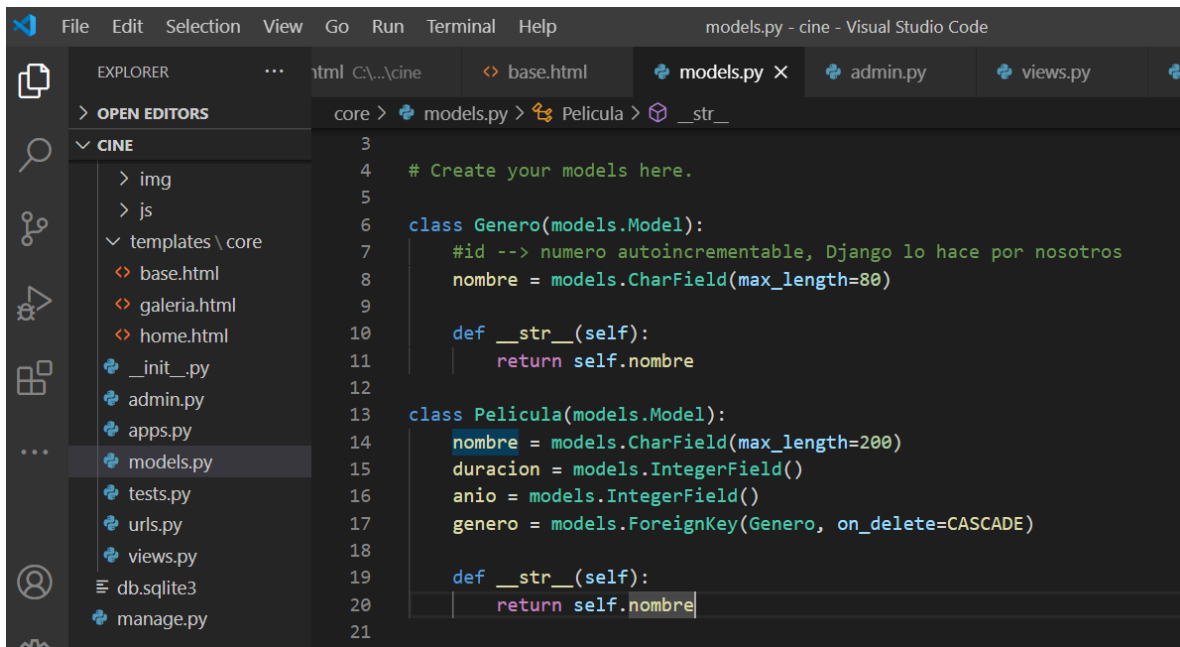


Pero al guardar se ve extraño



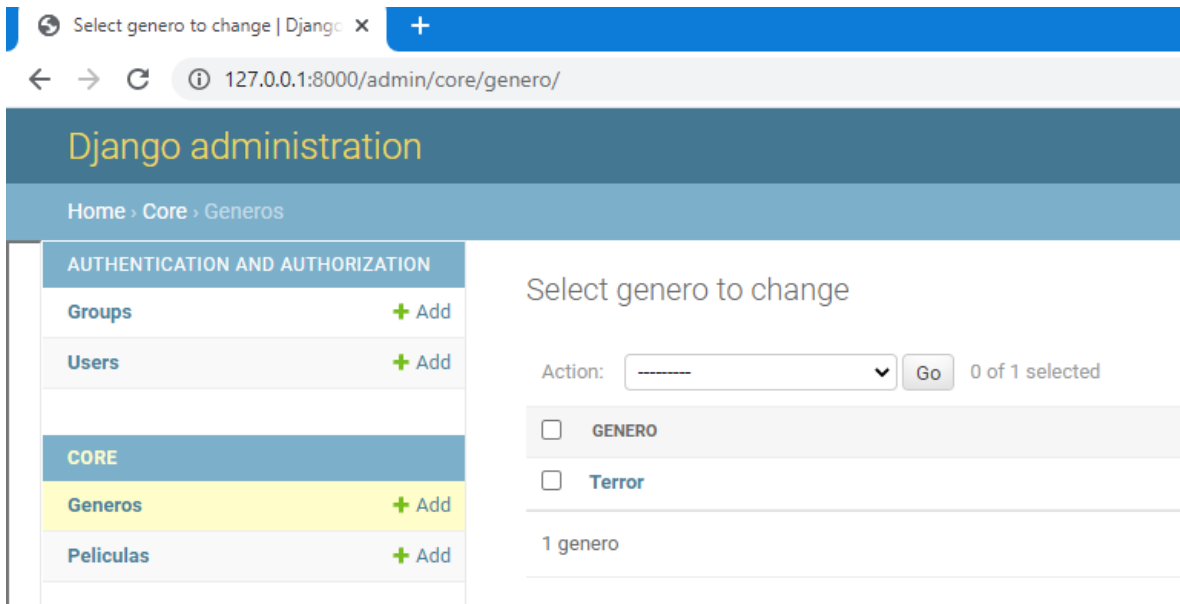
Dice en el género object (1), es mejor cambiarlo

Para eso editamos el archivo `core/models.py` y le indicamos como se verá cuando lo llamen



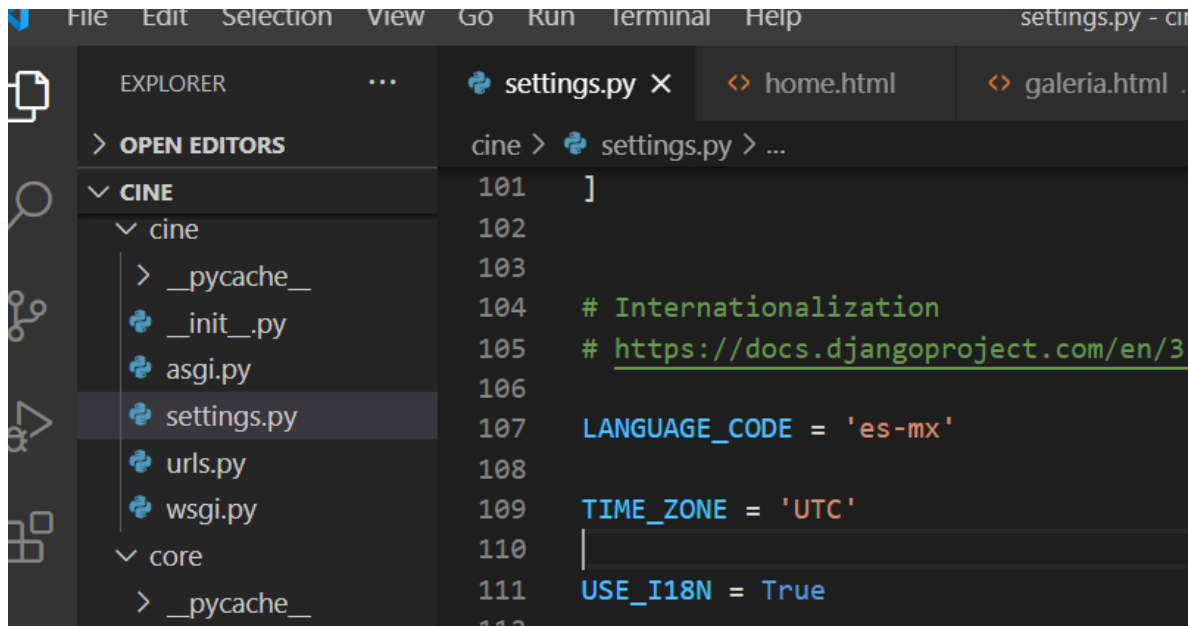
```
3
4 # Create your models here.
5
6 class Genero(models.Model):
7     #id --> numero autoincrementable, Django lo hace por nosotros
8     nombre = models.CharField(max_length=80)
9
10    def __str__(self):
11        return self.nombre
12
13    class Pelicula(models.Model):
14        nombre = models.CharField(max_length=200)
15        duracion = models.IntegerField()
16        anio = models.IntegerField()
17        genero = models.ForeignKey(Genero, on_delete=CASCADE)
18
19    def __str__(self):
20        return self.nombre
21
```

Ya con eso se ve distinto en el administrador



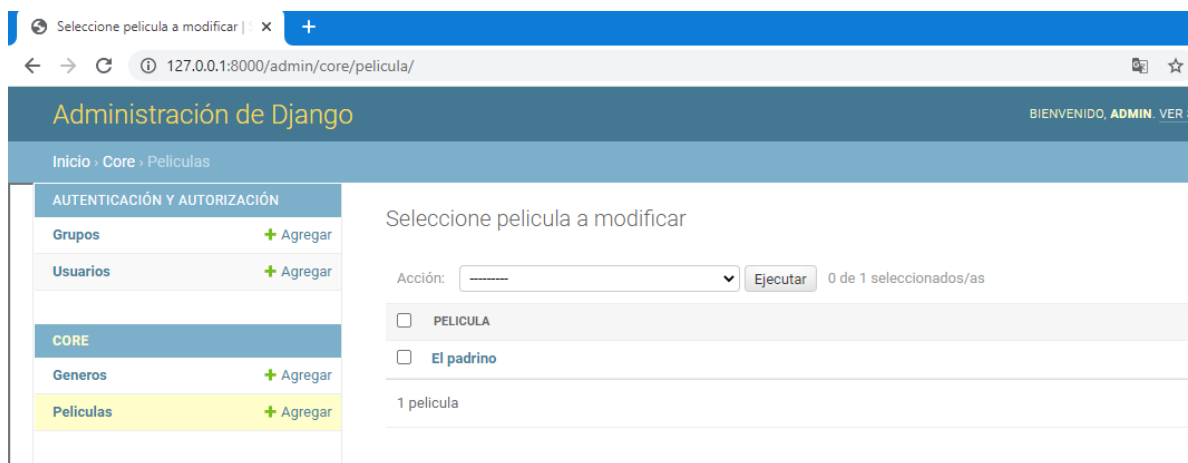
Ahora ya se empiezan a ver los nombres

Si necesitamos o queremos que se vea en español, le cambiamos el lenguaje en *cine/settings.py*

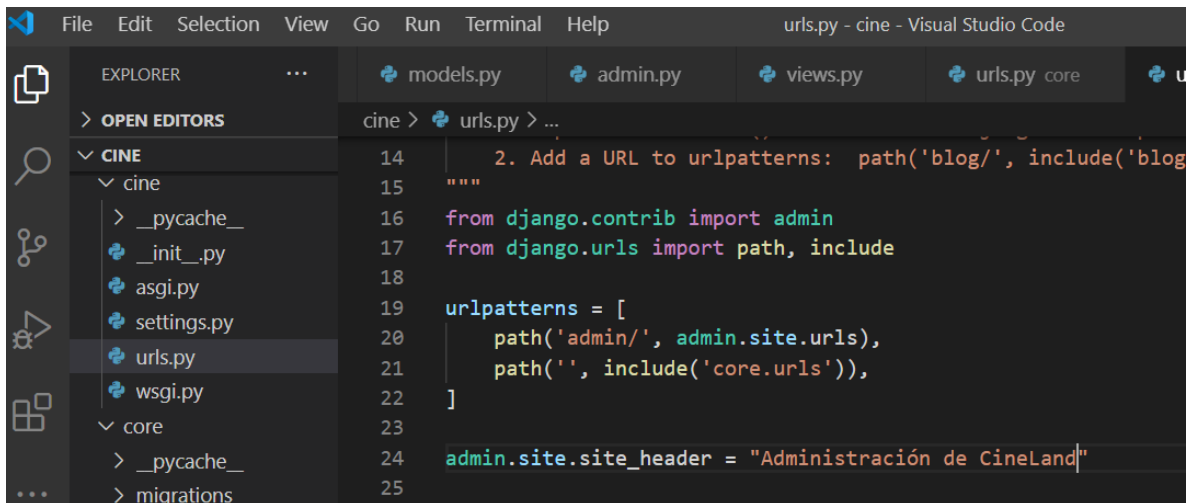


The image shows a Visual Studio Code editor window with the Explorer sidebar on the left and the Editor pane on the right. The Explorer sidebar shows a project structure with a folder named 'CINE' containing subfolders 'core' and 'cine'. The 'cine' folder contains files: '__pycache__', '__init__.py', 'asgi.py', 'settings.py', 'urls.py', and 'wsgi.py'. The 'settings.py' file is selected and open in the Editor pane. The code in the Editor pane shows the following lines:

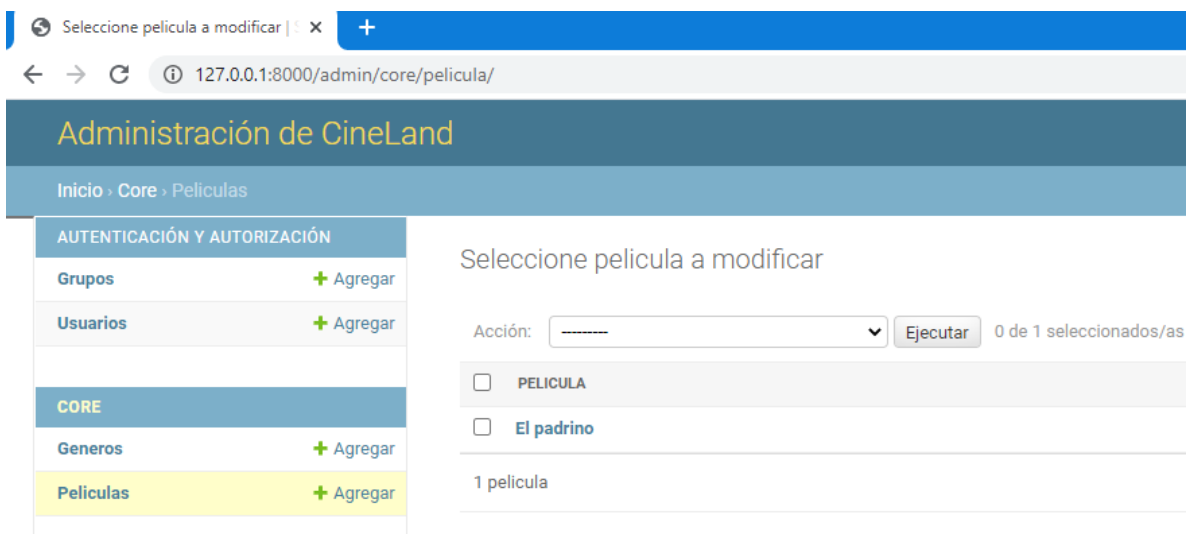
```
101 ]
102
103
104 # Internationalization
105 # https://docs.djangoproject.com/en/3
106
107 LANGUAGE_CODE = 'es-mx'
108
109 TIME_ZONE = 'UTC'
110
111 USE_I18N = True
```



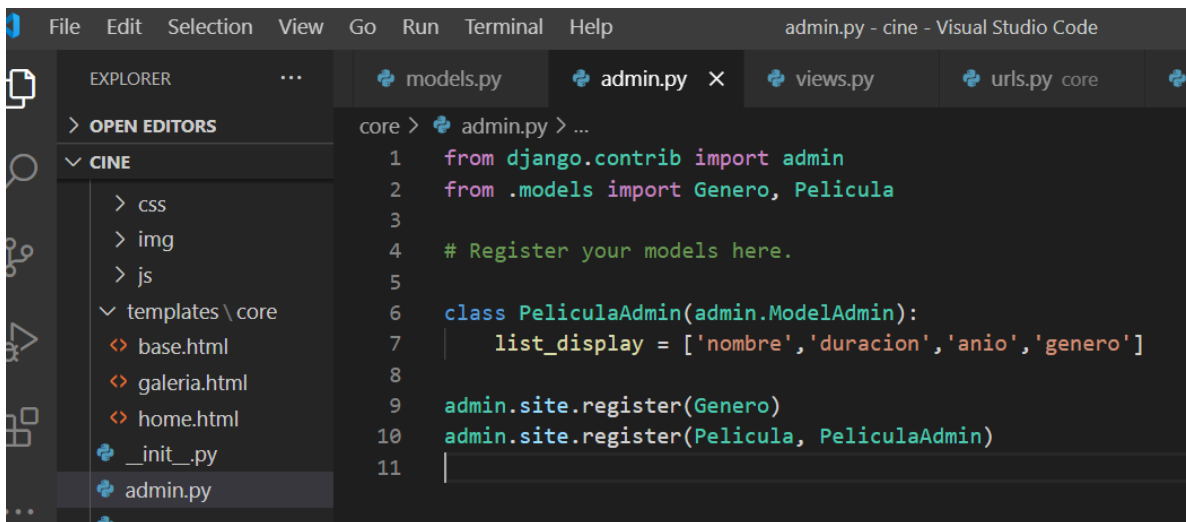
Para cambiar el nombre de Administración de Django por el de nuestro sitio: modificamos el archivo `cine/urls.py` y agregamos la línea `admin.site.site_header = "Administración de CineLand"`



```
14      2. Add a URL to urlpatterns: path('blog/', include('blog
15      """
16      from django.contrib import admin
17      from django.urls import path, include
18
19      urlpatterns = [
20          path('admin/', admin.site.urls),
21          path('', include('core.urls')),
22      ]
23
24      admin.site.site_header = "Administración de CineLand"
```

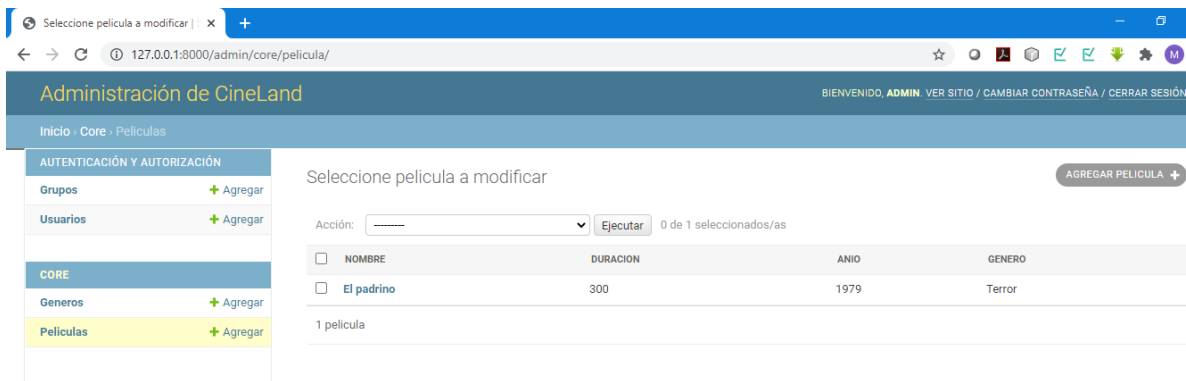


Para mostrar de forma más detallada los campos a mostrar en este módulo, modificamos el archivo `core/admin.py`

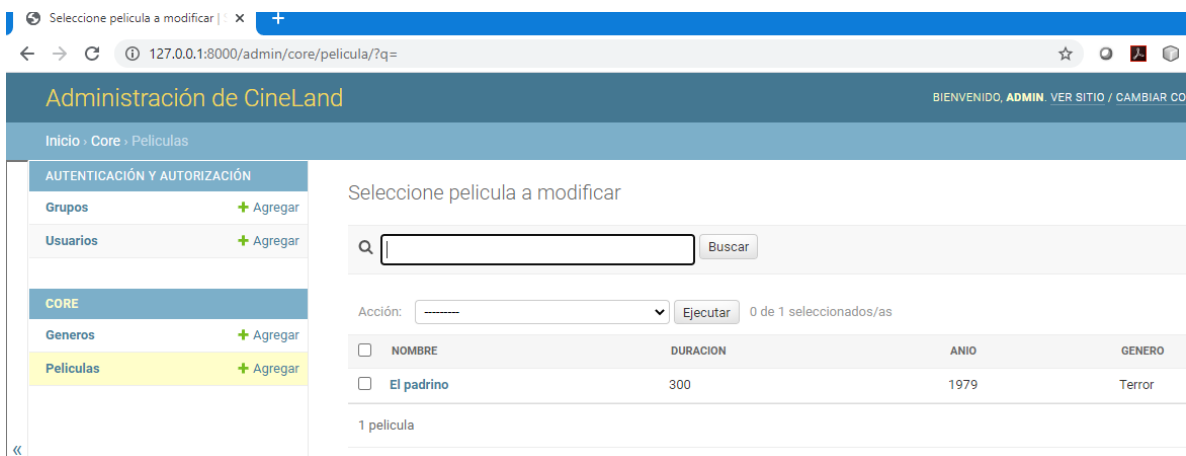
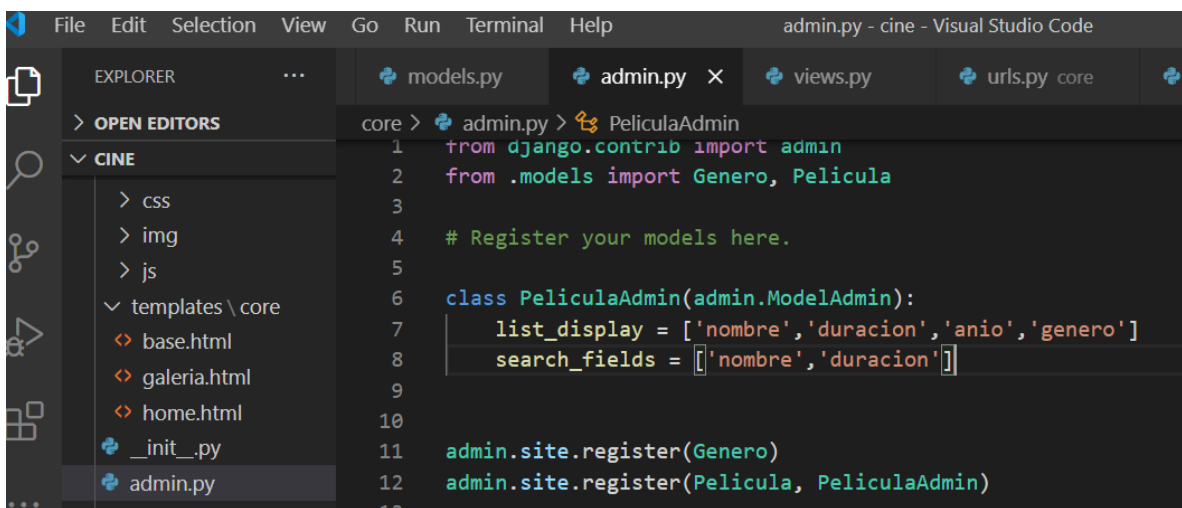


```
1 from django.contrib import admin
2 from .models import Genero, Pelicula
3
4 # Register your models here.
5
6 class PeliculaAdmin(admin.ModelAdmin):
7     list_display = ['nombre', 'duracion', 'anio', 'genero']
8
9     admin.site.register(Genero)
10    admin.site.register(Pelicula, PeliculaAdmin)
11
```


Si recargamos el navegador, ahora se tendrán las columnas desplegadas

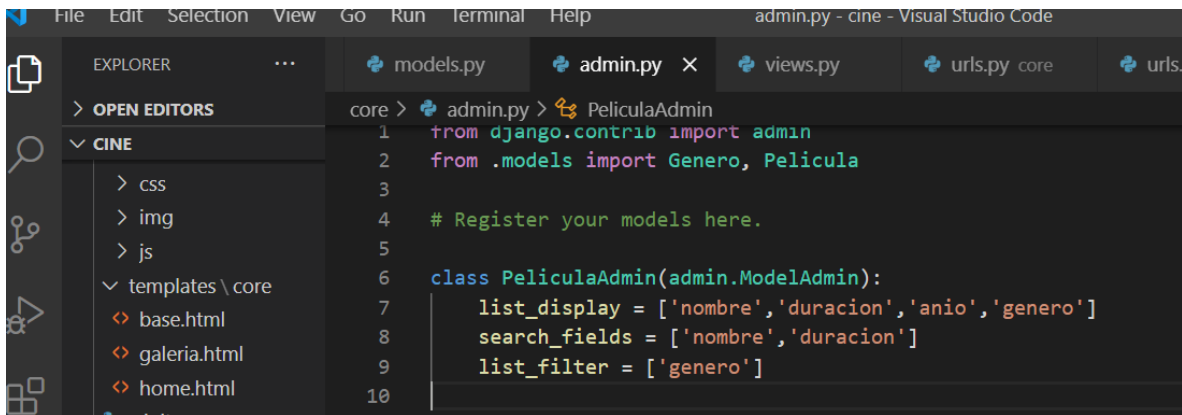


Agreguemos un buscador de películas:



Donde ya puedes buscar por nombre o por duración

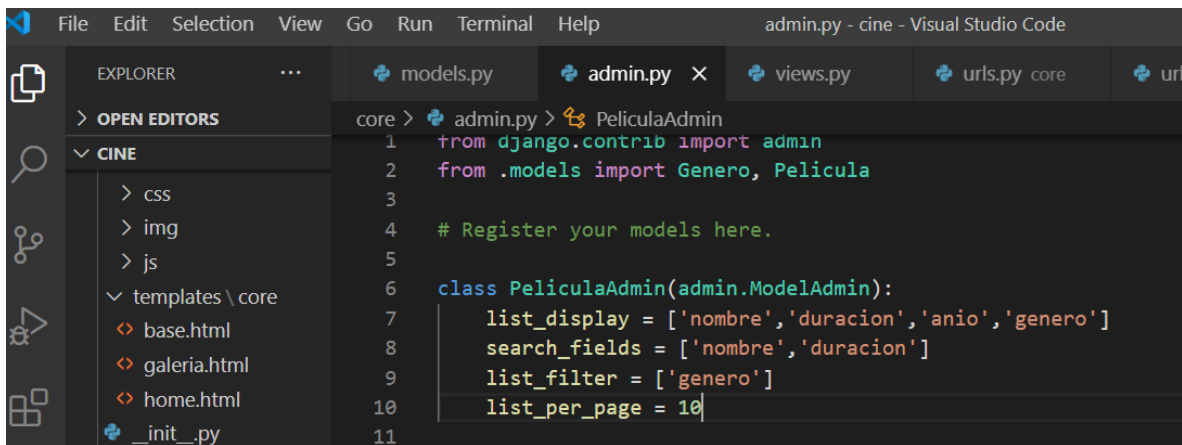
Agreguemos un filtro, agreguemos más géneros y películas para poder filtrar



```
core > admin.py > PeliculaAdmin
1 from django.contrib import admin
2 from .models import Genero, Pelicula
3
4 # Register your models here.
5
6 class PeliculaAdmin(admin.ModelAdmin):
7     list_display = ['nombre', 'duracion', 'anio', 'genero']
8     search_fields = ['nombre', 'duracion']
9     list_filter = ['genero']
10
```



Si quieres paginar, le indicas cuantos registros necesitas por cada página



```
core > admin.py > PeliculaAdmin
1 from django.contrib import admin
2 from .models import Genero, Pelicula
3
4 # Register your models here.
5
6 class PeliculaAdmin(admin.ModelAdmin):
7     list_display = ['nombre', 'duracion', 'anio', 'genero']
8     search_fields = ['nombre', 'duracion']
9     list_filter = ['genero']
10     list_per_page = 10
11
```

Todo lo anterior es para un super usuario, un administrador.

¿Podemos hacer un CRUD para un usuario no administrador ?

