ISPC
INSTITUTO SUPERIOR
POLITÉCNICO CÓRDOBA

TECNICATURA SUPERIOR EN

Desarrollo Web y
Aplicaciones Digitales

NOMBRE DEL ESPACIO CURRICULAR

Módulo

Programador Web

Tema

Primer proyecto en Django

Dirección General de EDUCACIÓN TÉCNICA Y FORMACIÓN PROFESIONAL
Ministerio de EDUCACIÓN
GOBIERNO DE LA PROVINCIA DE CÓRDOBA
entre todos
Hacemos

ÍNDICE

ÍNDICE **1**
CRUD EN Django **2**
Opción 2: VirtualEnv 2
Creando nuestro primer proyecto 3
Migración inicial 6
Usuario del Panel de Administrador 7
Creando la aplicación 7

CRUD EN Django

Requisito:

- Tener instalado Visual Studio Code
- Tener instalada la extensión de Python para Visual Studio Code
- Tener instalado Python3
- Tener instalado Django

Opcionales:

- Usar una máquina virtual o contenedores.
- Usar virtualenv en caso de no utilizar maquina virtual, esto es para evitar que Django se instale en el entorno “global” de Python.

Opción 2: VirtualEnv

Como primer paso instalamos VirtualEnv con el siguiente comando:

pip install virtualenv

Creamos el entorno

venv mientornovirtual

Luego, activamos el entorno virtual

source mientornovirtual/Scripts/activate

Listo, ya podemos seguir con Django API REST.

Creando nuestro primer proyecto

Crearemos nuestro proyecto con el siguiente comando

django-admin startproject abm_ispc

Una vez creado, nos generó el directorio con los archivos esenciales para comenzar.

/abm_ispc

├── */abm_ispc*

├── db.sqlite3

├── manage.py

Nosotros trabajaremos con MySQL como motor de base de datos, lo cual no significa que no puedan trabajar con otro, pero tengan en cuenta que se brindará soporte a lo trabajado con MySQL. Asimismo, utilizaremos en todo momento Visual Studio Code.

En el directorio del proyecto, nos encontraremos con la siguiente estructura:

/abm_ispc

├── */abm_ispc*

├── /_pycache_

├── _init_.py

├── asgi.py

├── settings.py

```
    ├── urls.py
    ├── wsgi.py
├── db.sqlite3
├── manage.py
```

En primera instancia, nos interesa el archivo “settings.py”

```python
# Importamos el módulo 'os'
import os

# Configuramos la Base de datos
DATABASES = {
    'sqlite': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    },
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'api_rest_ispc',
        'USER': 'root',
        'PASSWORD': 'contraseña',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
            'sql_mode': 'traditional',
        }
    }
}
```

Para copiar y pegar:

```
DATABASES = {
    'sqlite': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    },
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'abm_ispc',
        'USER': 'root',
        'PASSWORD': 'contraseña',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
            'sql_mode': 'traditional',
        }
    }
}
```

Migración inicial

Django nos provee “migraciones” que nos permiten crear varias tablas que van a ser parte del sistema de autenticación de nuestra aplicación.

Antes de seguir, hay que asegurarse que el servidor MySQL esté funcionando, y crear la base de datos 'abm_ispc'.

Ejecutaremos el comando:

python manage.py makemigrations

Y luego:

python manage.py migrate

Si entramos a ver nuestra base de datos, utilizando PhpMyAdmin, veremos lo siguiente:

auth_group	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	0	InnoDB	utf8mb4_general_ci	32.0 KB	-
auth_group_permissions	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
auth_permission	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	24	InnoDB	utf8mb4_general_ci	32.0 KB	-
auth_user	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	0	InnoDB	utf8mb4_general_ci	32.0 KB	-
auth_user_groups	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
auth_user_user_permissions	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
django_admin_log	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
django_content_type	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	6	InnoDB	utf8mb4_general_ci	48.0 KB	-
django_migrations	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	18	InnoDB	utf8mb4_general_ci	16.0 KB	-
django_session	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	0	InnoDB	utf8mb4_general_ci	32.0 KB	-
10 tablas	**Número de filas**						**48**	**InnoDB**	**utf8mb4_general_ci**	**384.0 KB**	**0 B**

Usuario del Panel de Administrador

Como siguiente paso, tengo que crear un usuario para administrar Django, esto lo realizaremos con el comando:

python manage.py createsuperuser

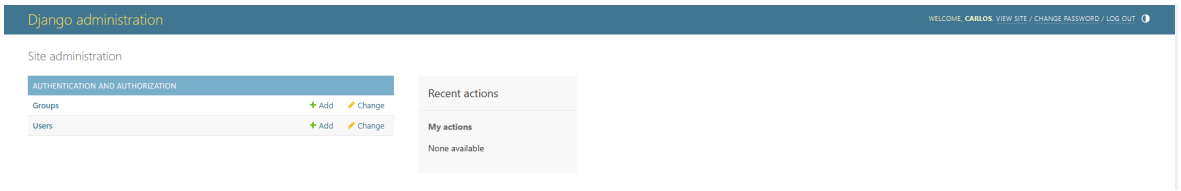
Luego de introducir los datos, procederemos a poner en funcionamiento el servidor para poder probar el panel. Esto lo haremos con el comando:

python manage.py runserver

Ingresamos en nuestro navegador web e iremos a:

http://127.0.0.1:8000/admin

Luego de ingresar con nuestro usuario y contraseña creados, veremos lo siguiente:

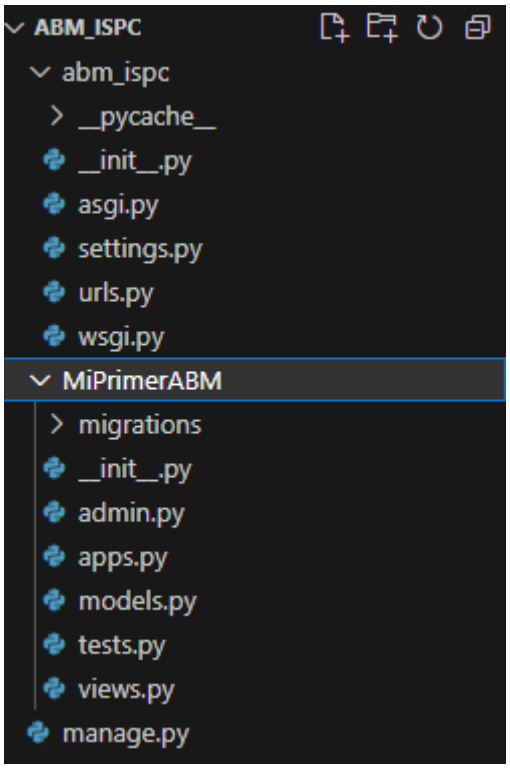


Creando la aplicación

Para crear nuestra aplicación, utilizaremos el comando:

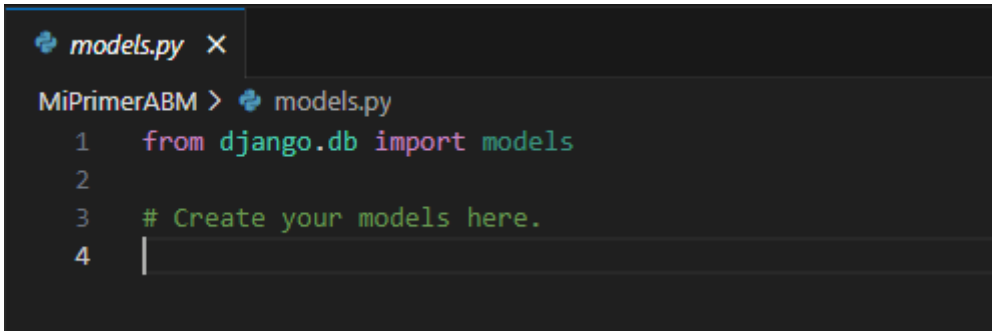
python manage.py startapp MiPrimerABM

Este comando nos creará una carpeta con archivos dentro, que se verá asi en Visual Studio Code.



El siguiente paso será crear nuestro primer modelo y eso lo haremos en el archivo *models.py* de nuestra aplicacion “MiPrimerABM”

El archivo se verá inicialmente así:



Aquí entra en juego el modelo de base de datos definido, definiremos las tablas necesarias para operar, en mi caso pienso en una tienda que vende productos artesanales de consumo masivo, por lo que estaremos utilizando 2 tablas para simplificar nuestro ejemplo:

1. Producto
2. Categoría

```python
# Create your models here.
class Categoria(models.Model):

    id_Categoria = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=100, blank=False)
    descripcion = models.TextField( max_length=1000, blank=False)
    class Meta:
        db_table = 'categoria'
        verbose_name = 'Categoria'
        verbose_name_plural = 'Categorias'
    def __unicode__(self):
        return self.nombre
    def __str__(self):
        return self.nombre

class Producto(models.Model):

    id_Producto = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=100, blank=False)
    descripcion = models.TextField( max_length=1000, blank=False)
    peso = models.DecimalField( max_length=100, blank=False, decimal_places = 2, max_digits=10)
    precio = models.DecimalField(blank=False, default=2000, decimal_places = 2, max_digits=10)
    cantidad = models.IntegerField(blank=False, default=2000)
    id_Categoria = models.ForeignKey(Categoria, to_field='id_Categoria', on_delete=models.CASCADE)
    class Meta:
        db_table = 'producto'
        verbose_name = 'Producto'
        verbose_name_plural = 'Productos'

    def __unicode__(self):
        return self.nombre
    def __str__(self):
        return self.nombre
```

Una vez creadas nuestras clases, que representan a nuestras tablas en la base de datos junto con sus relaciones, deberemos registrar los modelos para que los veamos en el panel de administrador de Django.

Para ello iremos a *MiPrimerABM/admin.py* que se verá asi:

MiPrimerABM > admin.py

```python
from django.contrib import admin

# Register your models here.

```

Introduciremos:

```
MiPrimerABM > admin.py > ...
1   from django.contrib import admin
2
3   from .models import Categoria
4   from .models import Producto
5
6   # Register your models here.
7   class CategoriaAdmin(admin.ModelAdmin):
8       list_display = ( 'nombre', 'descripcion')
9
10  class ProductoAdmin(admin.ModelAdmin):
11      list_display = ( 'nombre', 'descripcion', 'peso', 'precio', 'cantidad', 'id_Categoria')
12
13
14  admin.site.register(Categoria, CategoriaAdmin)
15  admin.site.register(Producto, ProductoAdmin)
16
```

Ahora toca registrar la aplicación en Django, para ello iremos a *abm_ispc/settings.py* e introduciremos en “INSTALLED_APPS” el nombre de nuestra aplicación: “MiPrimerABM”

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'MiPrimerABM',
]
```

Para reflejar los cambios ejecutaremos los siguientes comandos:

python manage.py makemigrations

y

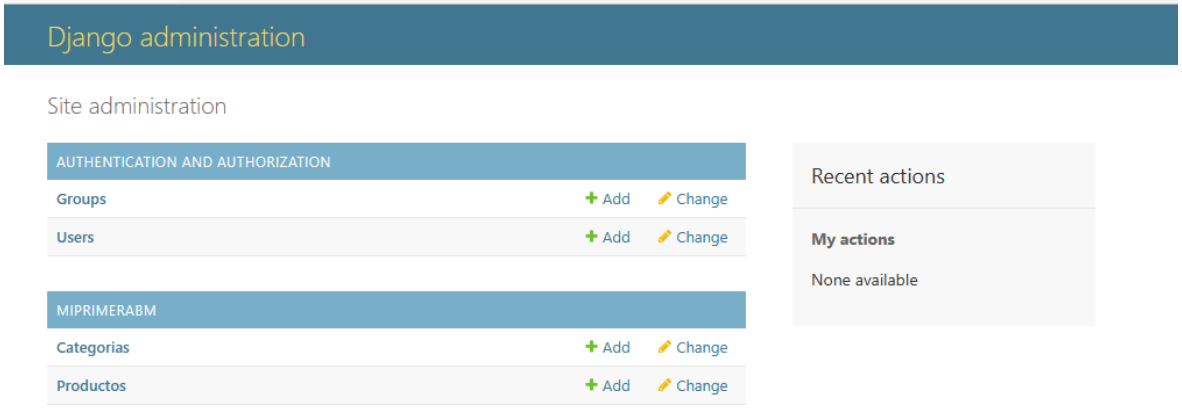
python manage.py migrate

Veremos lo siguiente:

```
C:\Users\PruebasServerWeb\Documents\ISPC\Python Django\abm_ispc>python manage.py makemigrations
Migrations for 'MiPrimerABM':
  MiPrimerABM\migrations\0001_initial.py
    - Create model Categoria
    - Create model Producto

C:\Users\PruebasServerWeb\Documents\ISPC\Python Django\abm_ispc>python manage.py migrate
System check identified some issues:

WARNINGS:
?: (mysql.W002) MariaDB Strict Mode is not set for database connection 'default'
        HINT: MariaDB's Strict Mode fixes many data integrity problems in MariaDB, such as data tr
n, by escalating warnings into errors. It is strongly recommended you activate it. See: https://do
n/4.2/ref/databases/#mysql-sql-mode
Operations to perform:
  Apply all migrations: MiPrimerABM, admin, auth, contenttypes, sessions
Running migrations:
  Applying MiPrimerABM.0001_initial... OK

C:\Users\PruebasServerWeb\Documents\ISPC\Python Django\abm_ispc>
```

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, 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 sessions.0001_initial... OK

C:\Users\PruebasServerWeb\Documents\ISPC\Python Django\abm_ispc>
```

Ejecutaremos el servidor y veremos como quedó:



Intentemos crear una categoría: