# ***Administrar Universidad y Envío de Correos Electrónicos***

***Oscar García (YouTube UskoKruM2010)***

***Jesús David Guevara Munar (Aprendiz ADSI 2397491)***

***Campoalegre – Huila (2022)***



***Contenido***

[***Administrar Universidad y Envío de Correos Electrónicos*** 1](#_Toc112148498)

[***1.*** ***Preparación de entorno de trabajo*** 2](#_Toc112148499)

[***2.*** ***Creación y configuración de proyecto MIUniversidad*** 3](#_Toc112148500)

[***3.*** ***Creación de apps o módulos del proyecto*** 4](#_Toc112148501)

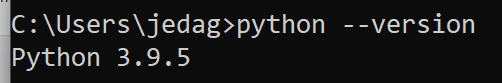
[***4.*** ***Creación de modelos y base de datos, panel de administración*** 6](#_Toc112148502)

[***5.*** ***Envío de correos electrónicos*** 14](#_Toc112148503)

# ***Preparación de entorno de trabajo***

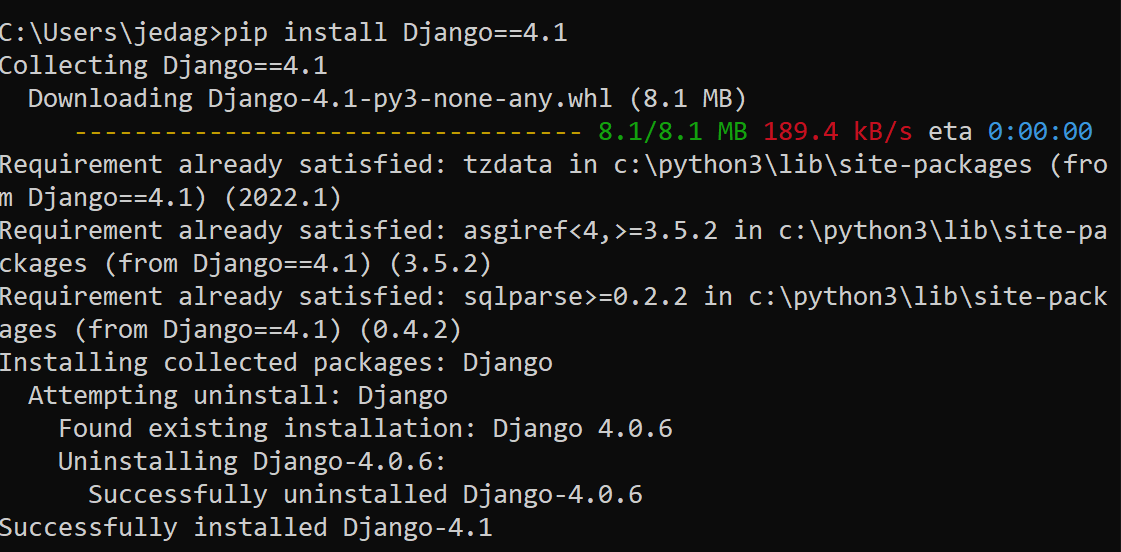
Descargar el intérprete de python: <https://www.python.org/downloads/>

Después de instalado comprobar en consola su versión con el siguiente comando:

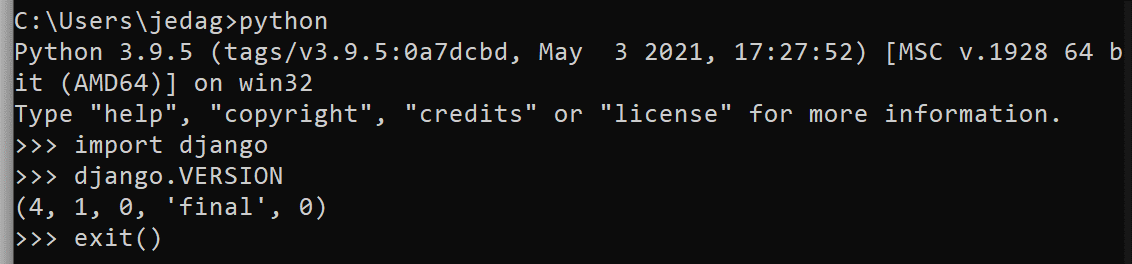


**PIP:** Package Installer for Python

Instalar Django (<https://www.djangoproject.com/download/>) ejecutando los siguientes comandos:



Comprobar si Django está instalado:



Descargar un editor de texto (VSCode <https://code.visualstudio.com/>).

# ***Creación y configuración de proyecto MIUniversidad***

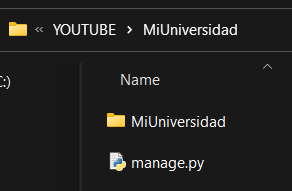
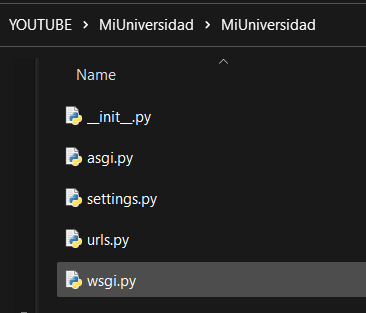
Nos dirigimos desde la consola de comando al directorio donde se pretende crear el proyecto:



Creamos el proyecto:



Verificamos el contenido del directorio que creo Django:

**manage.py:** Crear la base de datos, crear un usuario para la base de datos, realizar las migraciones hacia las BD y permite iniciar un servidor para desplegar la aplicación.

**\_\_init\_\_:** indica que este directorio va a ser tratada como un paquete de python.

**asgi.py:** (Asynchronous Server Gateway Interface) Interfaz de Puerta de enlace de servidor asíncrono.

**wsgi.ph:** (Web Server Gateway Interface) Interfaz de Puerta de enlace del servidor Web.

Tanto **asgi** como **wsgi** permite configura nuestro proyecto y trabajar en un servidor de producción de manera correcta.

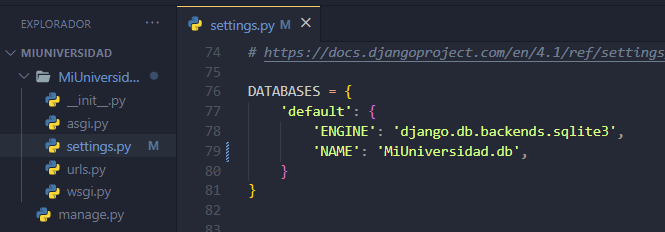
**setting.py:** configura parámetros importantes del proyecto (lenguaje, zona horario y motor de BD).

**urls.py:** permite indicar todo el índice del proyecto, las rutas que contiene y permite identificar que ruta nos devuelve determinada respuesta.

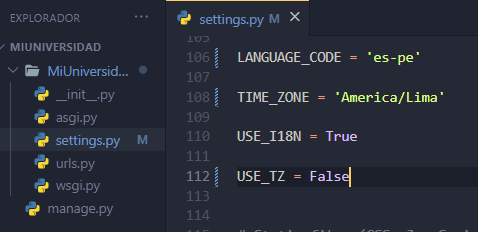
Abrimos el proyecto Django en el editor de texto y observamos la siguiente estructura:



Configuramos los datos de la base de datos que vamos a utilizar en el proyecto dentro del archivo **settings.py.**

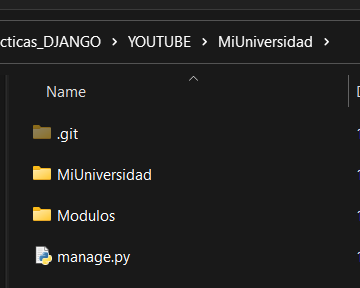


Configuramos los siguientes parámetros (idioma y zona horaria) en el archivo **settings.py**

****

# ***Creación de apps o módulos del proyecto***

Creamos una carpeta llamada **Modulos** dentro del proyecto.



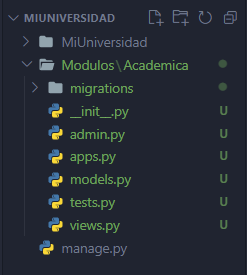
Desde la consola ingresamos a la carpeta **Modulos**



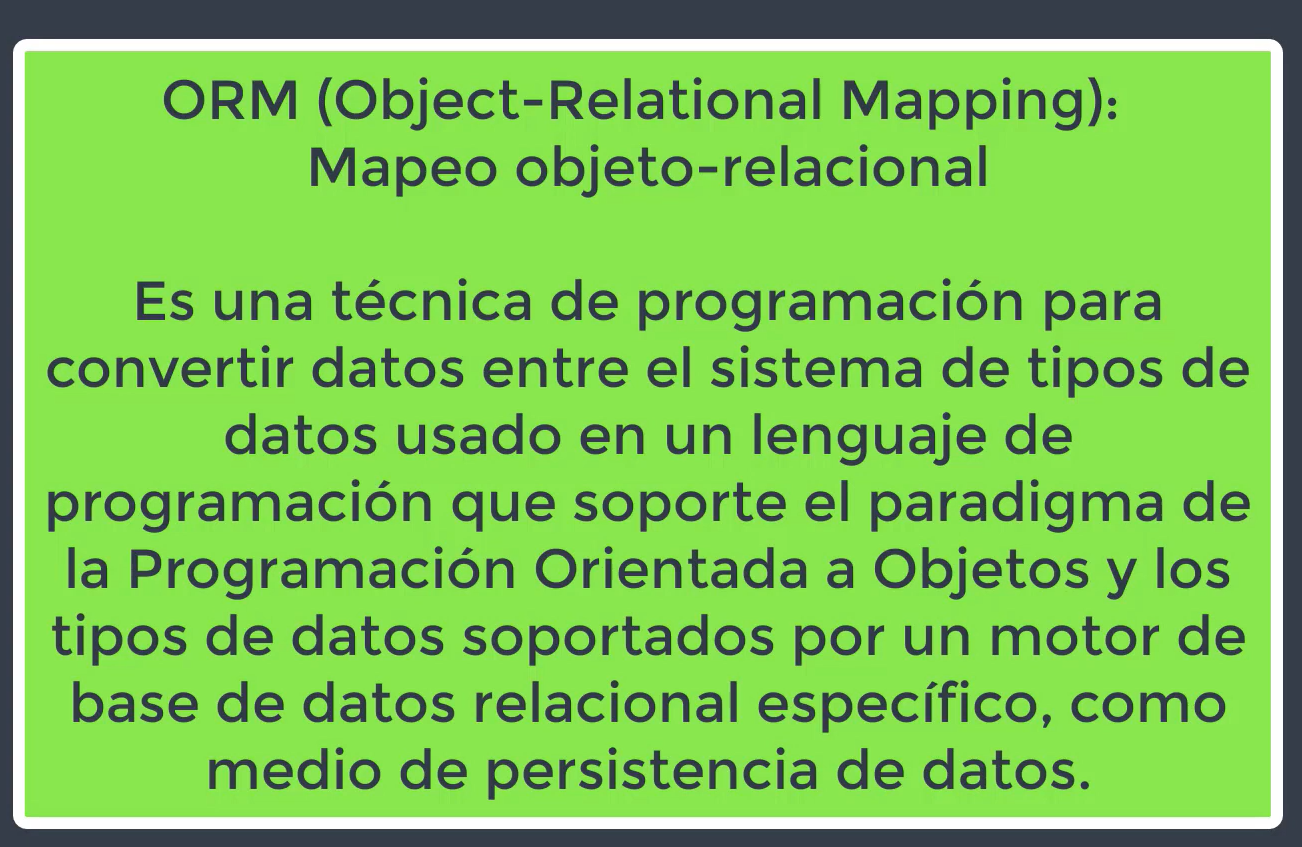
Creamos una app llamada **Academica** con el siguiente comando



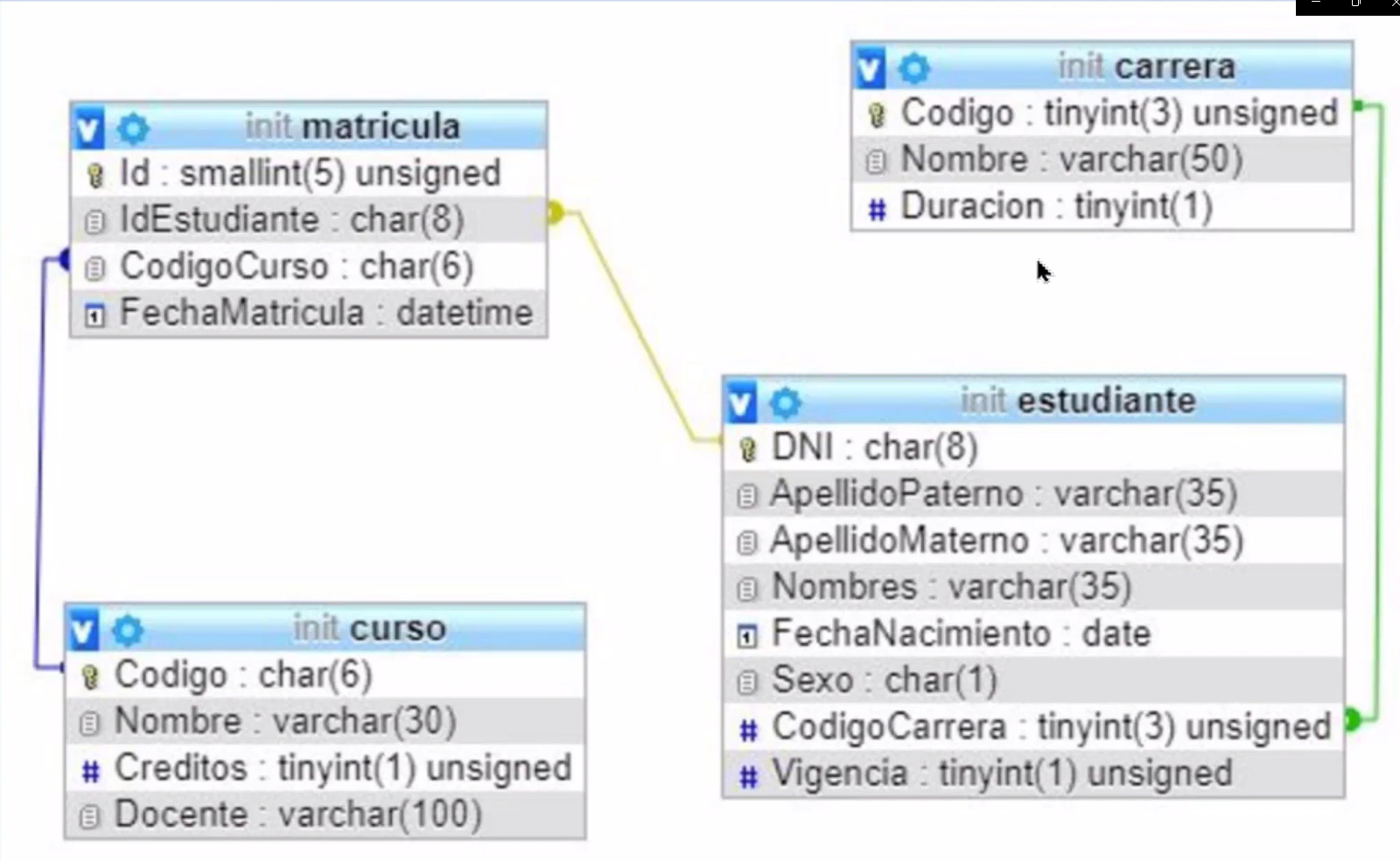
Verificamos la estructura de la aplicación creada:



* **admin.py** nos permite en el panel de administración registrar las entidades que podemos modificar.
* **migrations** refleja lo que hacemos en código en la base de datos correspondiente.

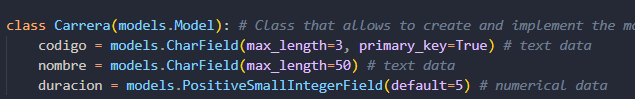


# ***Creación de modelos y base de datos, panel de administración***

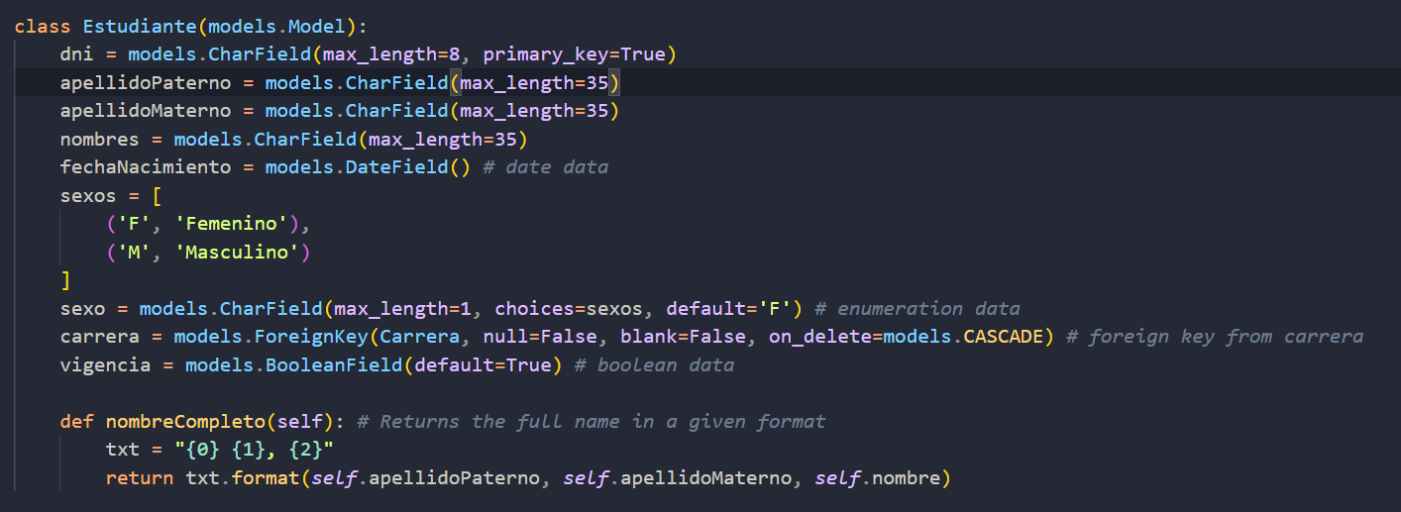


Creamos los modelos dependiendo si las entidades son fuertes o débiles (1.Carrera, 2.Estudiante, 3.Curso, 4.Matricula).

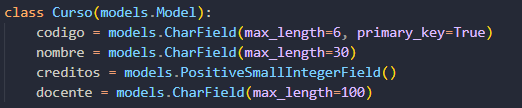
Creación del modelo **carrera** en el archivo **models.py**

****

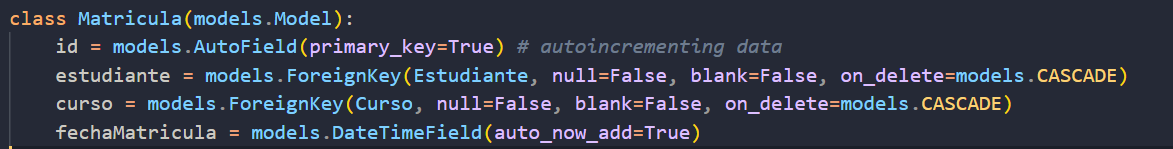
Creación del modelo **estudiante** en el archivo **models.py**

****

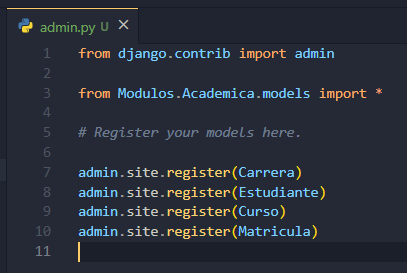
Creación del modelo **curso** en el archivo **models.py**

****

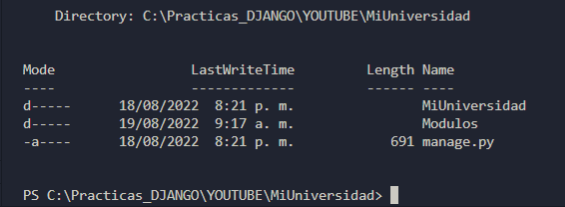
Creación del modelo **matricula** en el archivo **models.py**

****

Registrar los modelos creados para administrarlos en el panel de administración en el archivo **Academia/admin.py**:



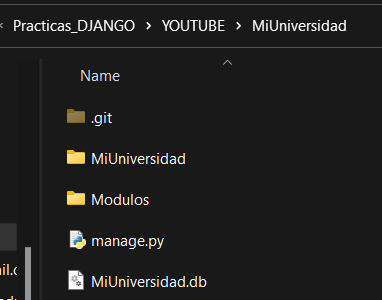
Desde la consola nos dirigimos al directorio del proyecto principal a nivel del archivo **mange.py**

****

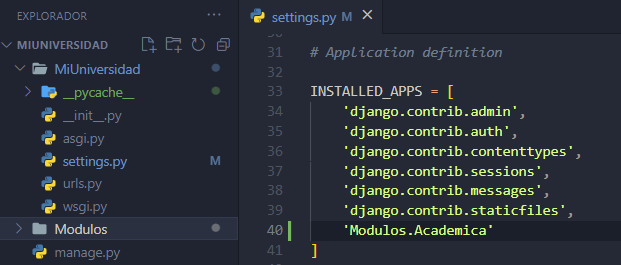
Crear la base de datos



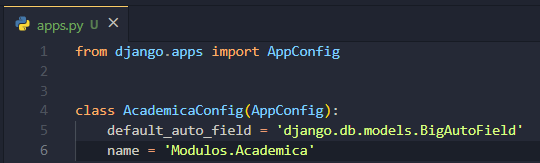
Verificamos en el proyecto que se haya creado la base de datos **SQLite (**MiUniversidad.db**):**



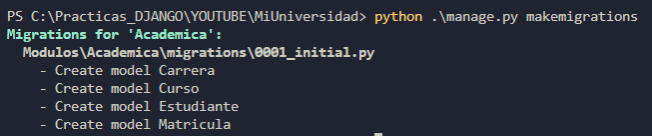
Registra la aplicación **Academia** dentro del archivo **settings.py** línea 40**:**

****

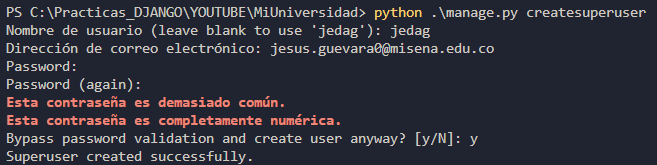
Renombramos la app **Academica** en el archivo **Modulos/Academica/apps.py** **(**línea 6**)** de la siguiente manera agregando el nombre de la carpeta que contiene las apps en este caso **Modulos:**

****

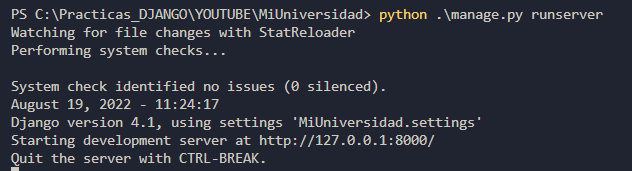
Realizamos las migraciones de los modelos que hemos creado para que se conviertan en tablas en la base de datos:



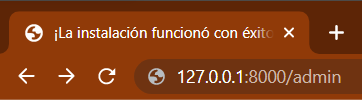
Creamos un super usuario (usuario=**jedag** y contraseña=**12345678**) para acceder a todo el sistema y administrarlo completamente.



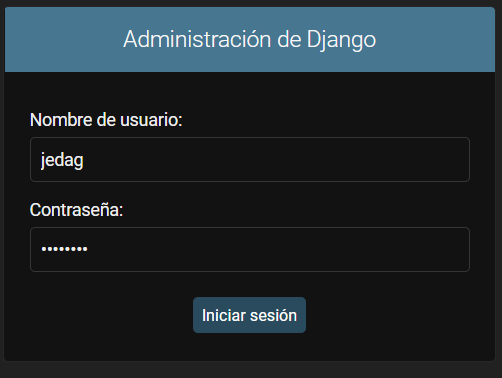
Iniciamos el servidor



En el navegador ingresamos la siguiente dirección

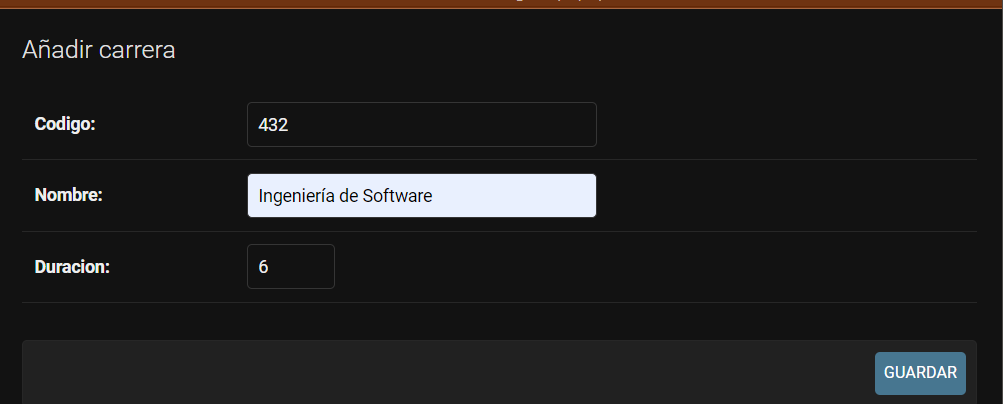
****

Ingresamos las credenciales del super usuario que creamos anteriormente.

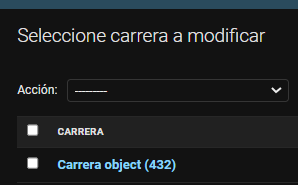


Agregamos un **estudiante** teniendo en cuenta que debemos registrar una carrera para que aparezcan en el listado de carrera y seleccionarla respectivamente.

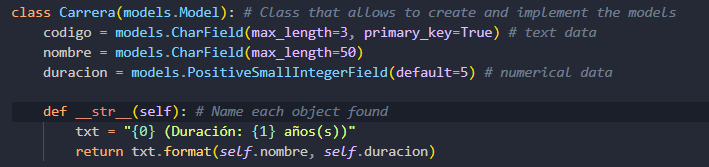




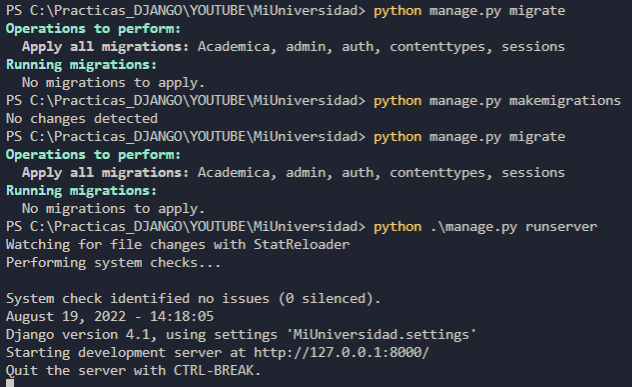
Así aparecen los registros en el panel de administrador (pero se espera que se muestren con su respectivo nombre de la carrera):



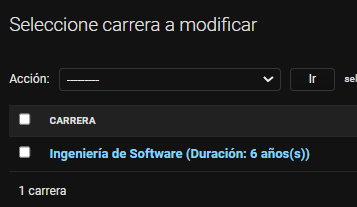
Agregamos la siguiente función **\_\_str\_\_** a la clase **carrera** para mostrar los nombres formateados de cada objeto o registro encontrado:



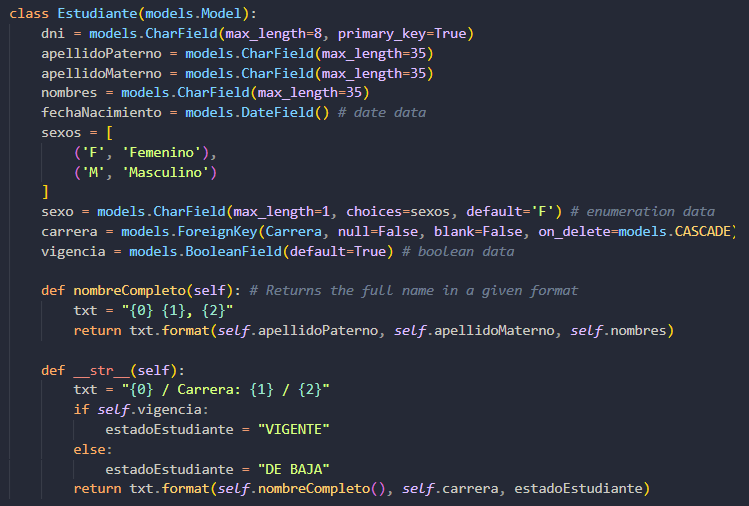
Reiniciamos todo ejecutando los siguientes comandos



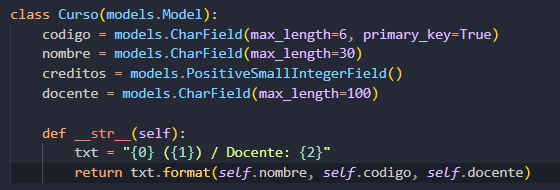
Volvemos al panel de administración en la sección carreras y visualizamos el cambio:



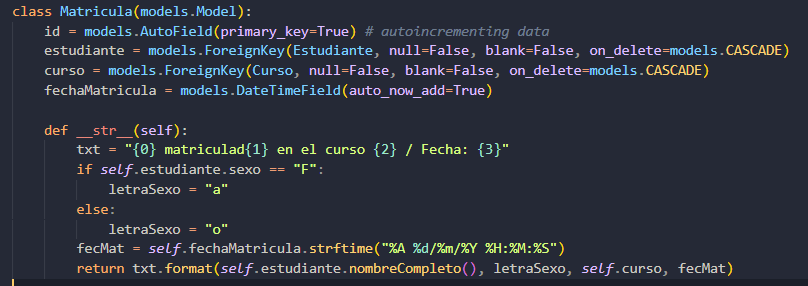
Agregamos la función **\_\_str\_\_** a la clase **Estudiante** del mismo archivo **models.py:**



Agregamos la función **\_\_str\_\_** a la clase **Curso**

****

Agregamos la función **\_\_str\_\_** a la clase **Matricula**

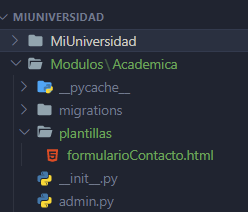
****

Para comprobar nos dirigimos nuevamente al panel de administración y agregamos nuevos registros y verificamos que cada objeto sea nombrado justo como lo hemos definido en cada una de las funciones **\_\_str\_\_**

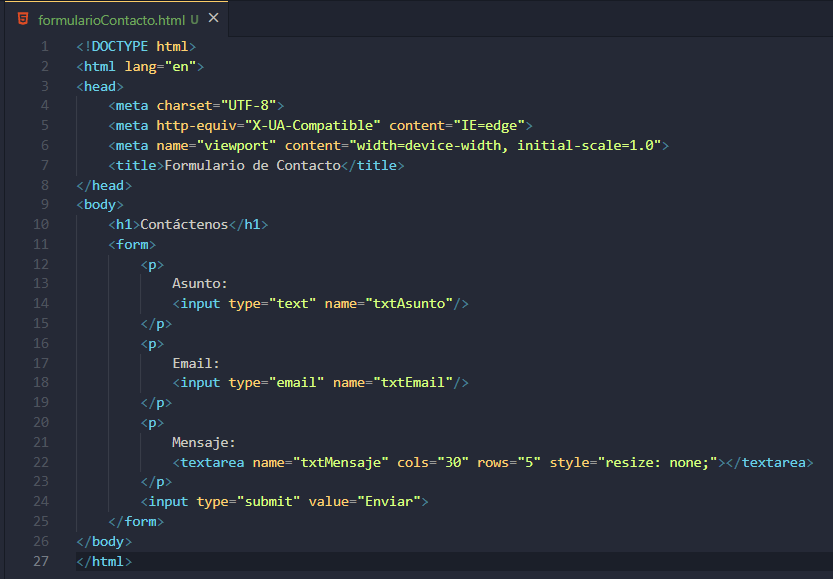
# ***Envío de correos electrónicos***

Creamos una nueva carpeta **plantillas** dentro del directorio **Modulos/Academica.**

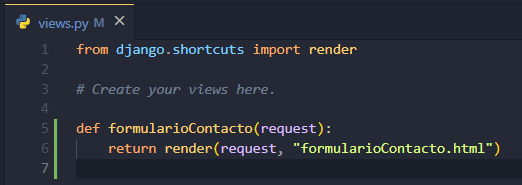
Dentro del directorio **plantillas** creamos un archivo llamado **formularioContacto,html.**

****

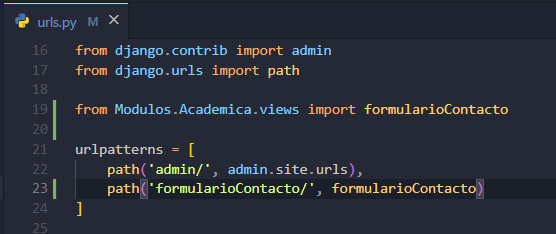
Dentro de este archivo agregamos el siguiente código HTML



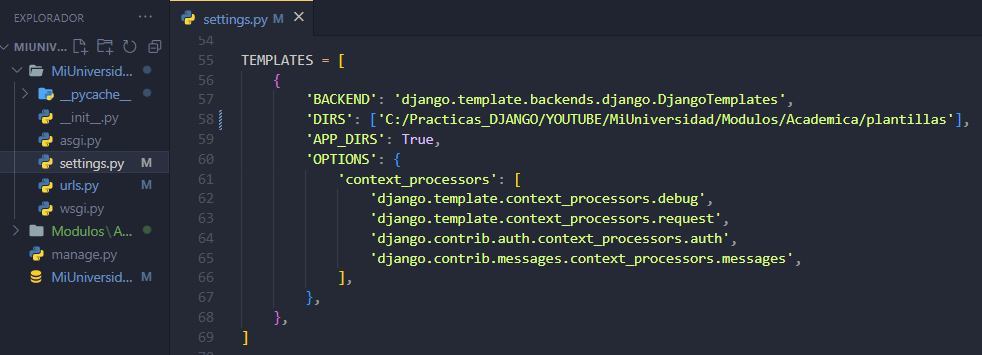
Definimos la vista en el archivo **Academica/views.py**



Definimos la **url** dentro del archivo **urls.py** del proyecto principal



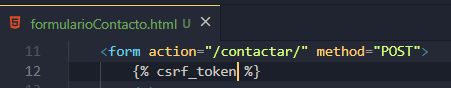
En el archivo **settings.py** del proyecto agregamos la dirección del directorio donde se encuentra el formulario en los parámetros de **TEMPLATES ‘DIRS’** línea 58.

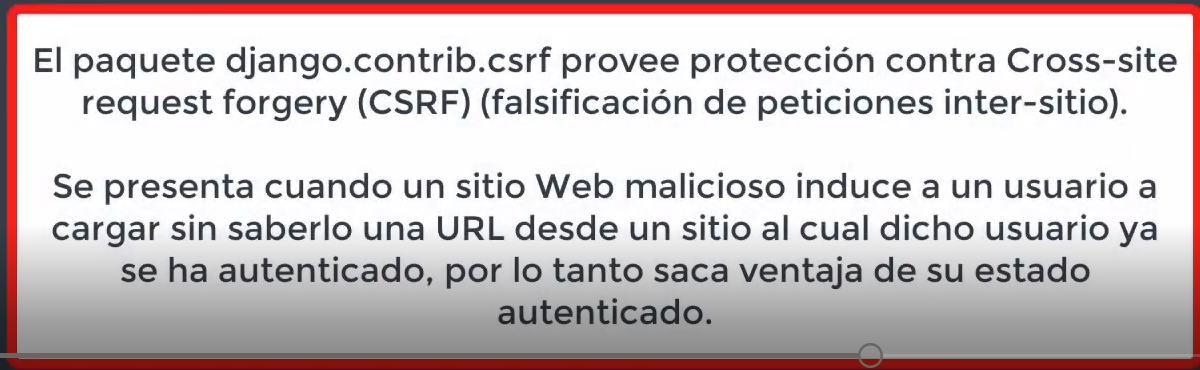


Iniciamos el servidor de nuevo escribimos en la url del navegador la ruta recién creada y verificamos si todo está correctamente relacionado.



Agregamos las siguientes líneas en el formulario del archivo **formularioContacto.html**

****



Agregamos la siguiente configuración en el archivo **settings.py** del proyecto para poder realizar envíos de correos electrónicos (después de la línea de **STATIC\_URL**).

STATIC\_URL = 'static/'

*# Configurations to send an email*

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_HOST = "smtp.gmail.com"

EMAIL\_USE\_TLS = True

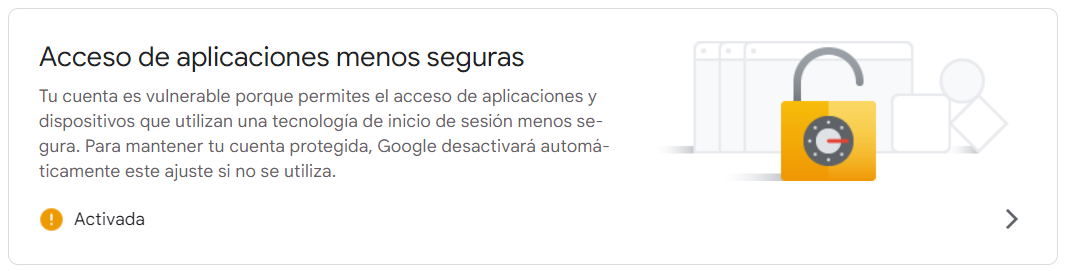
EMAIL\_PORT = 587

EMAIL\_HOST\_USER = "jesus.guevara0@misena.edu.co"

EMAIL\_HOST\_PASSWORD = "Jedague(3103G22022)"

Se debe tener en que tanto el user y el password son datos que pueden cambiar desde donde se pretende originar el correo.

Para permitir el envío de correos por Google ingresamos en la opción de **Gestionar Cuenta -> Seguridad.** Permitimos el acceso de aplicaciones menos seguras.



Agregamos el siguiente código en el archivo **Academia/views.py**

from django.core.mail import send\_mail

from django.conf import settings;

def contactar(request):

    if request.method == "POST":

        asunto = request.POST["txtAsunto"]

        mensaje = request.POST["txtMensaje"] + " / Email: " + request.POST["txtEmail"]

        email\_desde = settings.EMAIL\_HOST\_USER

        email\_para = ["jesus.guevara0@misena.edu.co"]

        send\_mail(asunto, mensaje, email\_desde, email\_para, fail\_silently=False)

        return render(request, "contactoExitoso.html")

    return render(request, "formularioContacto.html")

Creamos el archivo dentro del directorio **Academia/plantillas** y agregamos lo siguiente:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>Formulario de contacto enviado con éxito.</h1>

</body>

</html>

En el archivo **MiUniversidad/urls.py** importamos la vista (**contactar**) y creamos el puente (**contactar)**

from Modulos.Academica.views import formularioContacto, contactar

urlpatterns = [

    path('admin/', admin.site.urls),

    path('formularioContacto/', formularioContacto),

    path('contactar/', contactar)

]

Finalmente probamos la función de enviar correos electrónicos iniciando el servidor nuevamente e ingresar en el navegador la siguiente dirección: ***http://127.0.0.1:8000/formularioContacto/***

Si se desea configura el módulo de manera tal que pueda enviar correos al correo electrónico ingresado en el formulario realizamos el siguiente ajuste en el archivo **Academia/views.py** función **contactar:**

def contactar(request):

    if request.method == "POST":

        asunto = request.POST["txtAsunto"]

        mensaje = request.POST["txtMensaje"]

        email\_desde = settings.EMAIL\_HOST\_USER

        email\_para = [request.POST["txtEmail"]]

        send\_mail(asunto, mensaje, email\_desde, email\_para, fail\_silently=False)

        return render(request, "contactoExitoso.html")

    return render(request, "formularioContacto.html")