

Administrar Universidad y Envío de Correos Electrónicos

Oscar García (YouTube UskoKruM2010)

Jesús David Guevara Munar (Aprendiz ADSI 2397491)

Campoalegre – Huila (2022)



Contenido

<i>Administrar Universidad y Envío de Correos Electrónicos</i>	1
<i>1. Preparación de entorno de trabajo</i>	2
<i>2. Creación y configuración de proyecto MIUniversidad</i>	3
<i>3. Creación de apps o módulos del proyecto</i>	5
<i>4. Creación de modelos y base de datos, panel de administración</i>	6
<i>5. Envío de correos electrónicos</i>	14

1. 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:

```
C:\Users\jedag>python --version
Python 3.9.5
```

PIP: Package Installer for Python

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

```
C:\Users\jedag>pip install Django==4.1
Collecting Django==4.1
  Downloading Django-4.1-py3-none-any.whl (8.1 MB)
----- 8.1/8.1 MB 189.4 kB/s eta 0:00:00
Requirement already satisfied: tzdata in c:\python3\lib\site-packages (from Django==4.1) (2022.1)
Requirement already satisfied: asgiref<4,>=3.5.2 in c:\python3\lib\site-packages (from Django==4.1) (3.5.2)
Requirement already satisfied: sqlparse>=0.2.2 in c:\python3\lib\site-packages (from Django==4.1) (0.4.2)
Installing collected packages: Django
  Attempting uninstall: Django
    Found existing installation: Django 4.0.6
    Uninstalling Django-4.0.6:
      Successfully uninstalled Django-4.0.6
Successfully installed Django-4.1
```

Comprobar si Django está instalado:

```
C:\Users\jedag>python
Python 3.9.5 (tags/v3.9.5:0a7dcdb, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.VERSION
(4, 1, 0, 'final', 0)
>>> exit()
```

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

2. Creación y configuración de proyecto *MiUniversidad*

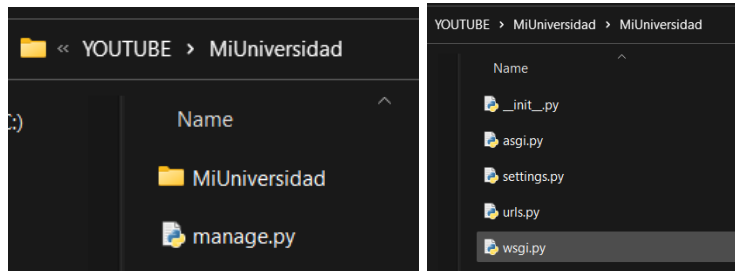
Nos dirigimos desde la consola de comando al directorio donde se pretende crear el proyecto:

```
C:\Practicas_DJANGO\YOUTUBE>
```

Creamos el proyecto:

```
C:\Practicas_DJANGO\YOUTUBE>django-admin startproject MiUniversidad
```

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.

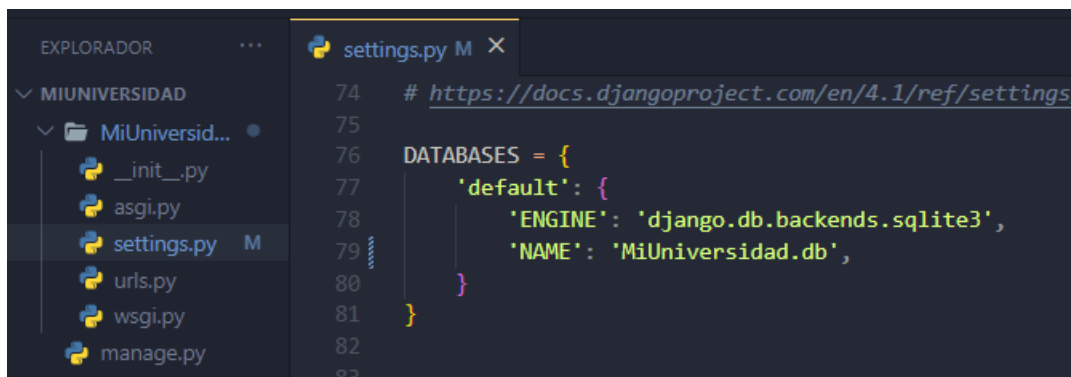
setting.py: configura parámetros importantes del proyecto (lenguaje, zona horario y motor de BD).

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

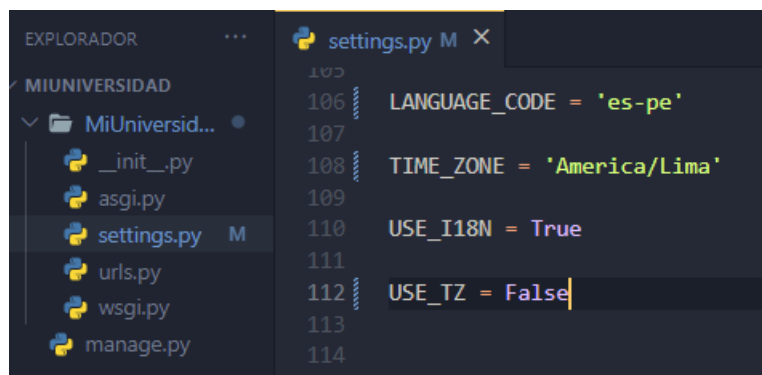
Abrimos el proyecto Django en el editor de texto y observamos la siguiente estructura:



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

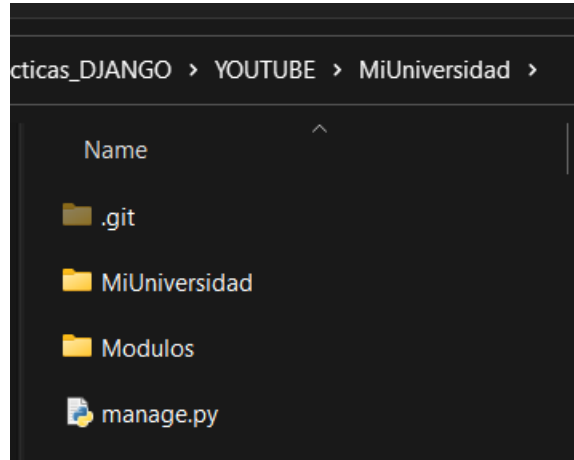


Configuramos los siguientes parámetros (idioma y zona horaria) en el archivo **settings.py**



3. Creación de apps o módulos del proyecto

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



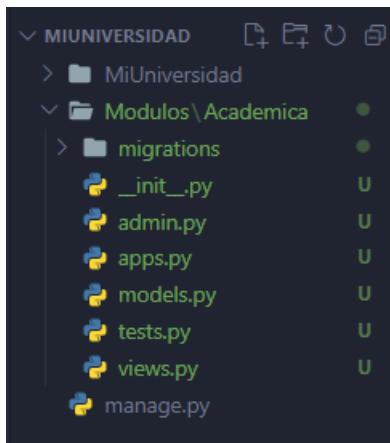
Desde la consola ingresamos a la carpeta **Modulos**

```
PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad\Modulos>
```

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

```
PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad\Modulos> django-admin startapp Academica
```

Verificamos la estructura de la aplicación creada:

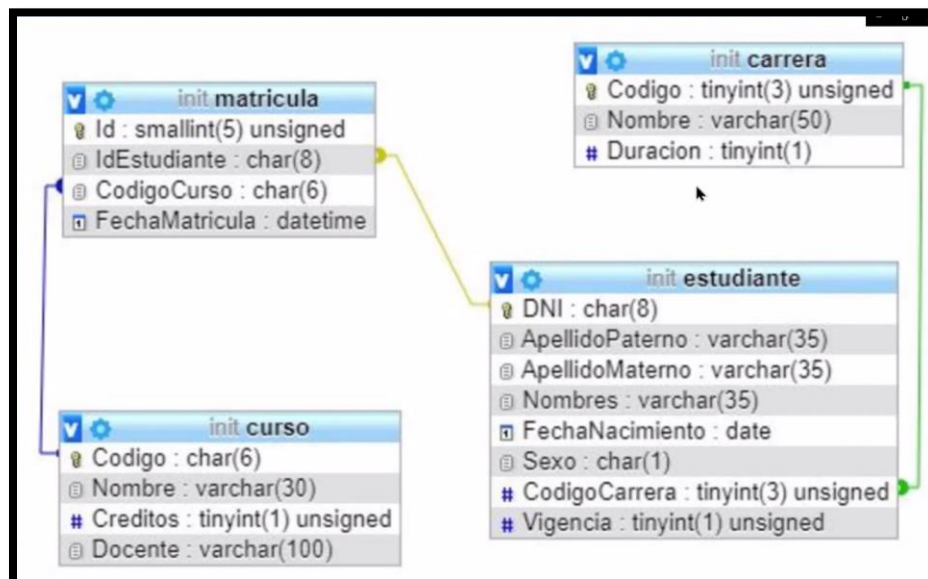


- **admin.py** nos permite en el panel de administración registrar las entidades que podemos modificar.
- **migrations** refleja lo que hacemos en código en la base de datos correspondiente.

ORM (Object-Relational Mapping): Mapeo objeto-relacional

Es una técnica de programación para convertir datos entre el sistema de tipos de datos usado en un lenguaje de programación que soporte el paradigma de la Programación Orientada a Objetos y los tipos de datos soportados por un motor de base de datos relacional específico, como medio de persistencia de datos.

4. 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**

```
class Carrera(models.Model): # Class that allows to create and implement the model
    codigo = models.CharField(max_length=3, primary_key=True) # text data
    nombre = models.CharField(max_length=50) # text data
    duracion = models.PositiveSmallIntegerField(default=5) # numerical data
```

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

```
class Estudiante(models.Model):
    dni = models.CharField(max_length=8, primary_key=True)
    apellidoPaterno = models.CharField(max_length=35)
    apellidoMaterno = models.CharField(max_length=35)
    nombres = models.CharField(max_length=35)
    fechaNacimiento = models.DateField() # date data
    sexos = [
        ('F', 'Femenino'),
        ('M', 'Masculino')
    ]
    sexo = models.CharField(max_length=1, choices=sexos, default='F') # enumeration data
    carrera = models.ForeignKey(Carrera, null=False, blank=False, on_delete=models.CASCADE) # foreign key from carrera
    vigencia = models.BooleanField(default=True) # boolean data

    def nombreCompleto(self): # Returns the full name in a given format
        txt = "{0} {1}, {2}"
        return txt.format(self.apellidoPaterno, self.apellidoMaterno, self.nombres)
```

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

```
class Curso(models.Model):
    codigo = models.CharField(max_length=6, primary_key=True)
    nombre = models.CharField(max_length=30)
    creditos = models.PositiveSmallIntegerField()
    docente = models.CharField(max_length=100)
```

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

```
class Matricula(models.Model):
    id = models.AutoField(primary_key=True) # autoincrementing data
    estudiante = models.ForeignKey(Estudiante, null=False, blank=False, on_delete=models.CASCADE)
    curso = models.ForeignKey(Curso, null=False, blank=False, on_delete=models.CASCADE)
    fechaMatricula = models.DateTimeField(auto_now_add=True)
```

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

```
admin.py U X
1  from django.contrib import admin
2
3  from Modulos.Academica.models import *
4
5  # Register your models here.
6
7  admin.site.register(Carrera)
8  admin.site.register(Estudiente)
9  admin.site.register(Curso)
10 admin.site.register(Matricula)
11
```

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

```
Directory: C:\Practicas_DJANGO\YOUTUBE\MiUniversidad

Mode                LastWriteTime         Length Name
----                -
d-----          18/08/2022   8:21 p. m.         MiUniversidad
d-----          19/08/2022   9:17 a. m.         Modulos
-a----          18/08/2022   8:21 p. m.         691 manage.py

PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad>
```

Crear la base de datos

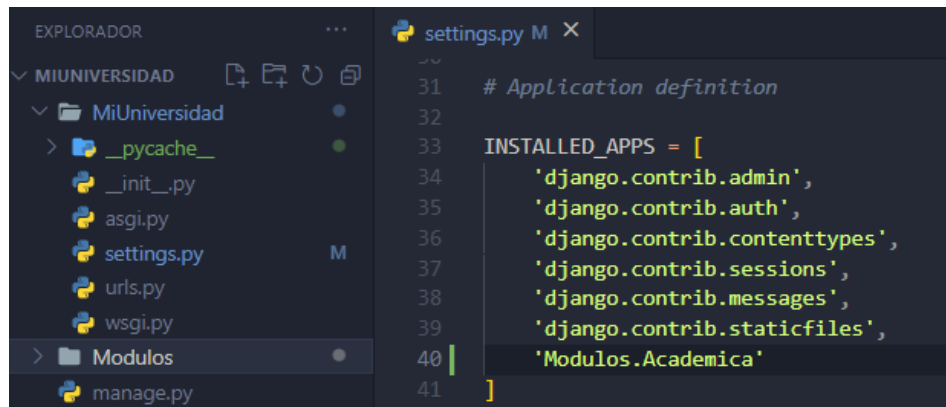
```
PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad> python manage.py migrate
```

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

```
Practicas_DJANGO > YOUTUBE > MiUniversidad

Name
----
.git
MiUniversidad
Modulos
manage.py
MiUniversidad.db
```

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

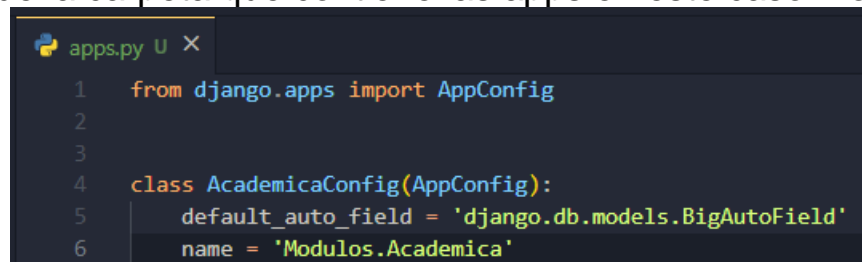


```

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     'Modulos.Academica'
41 ]

```

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**:

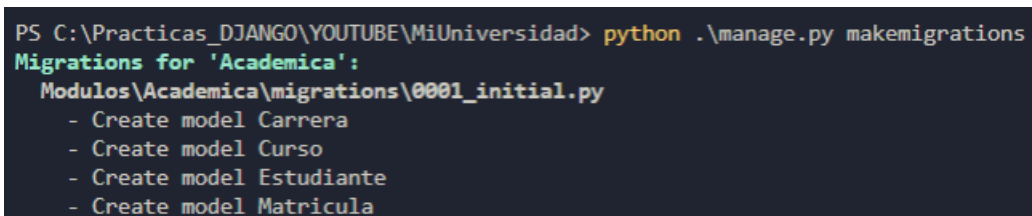


```

1 from django.apps import AppConfig
2
3
4 class AcademicaConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'Modulos.Academica'

```

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

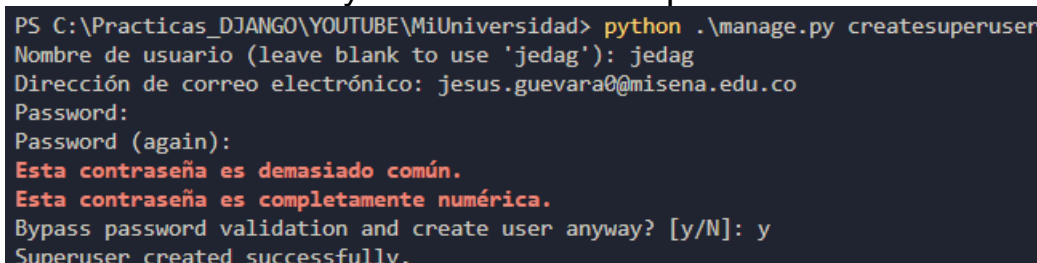


```

PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad> python .\manage.py makemigrations
Migrations for 'Academica':
  Modulos\Academica\migrations\0001_initial.py
    - Create model Carrera
    - Create model Curso
    - Create model Estudiante
    - Create model Matricula

```

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



```

PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad> python .\manage.py createsuperuser
Nombre de usuario (leave blank to use 'jedag'): jedag
Dirección de correo electrónico: jesus.guevara0@misena.edu.co
Password:
Password (again):
Esta contraseña es demasiado común.
Esta contraseña es completamente numérica.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

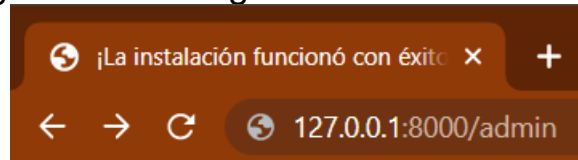
```

Iniciamos el servidor

```
PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 19, 2022 - 11:24:17
Django version 4.1, using settings 'MiUniversidad.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

En el navegador ingresamos la siguiente dirección



Ingresamos las credenciales del super usuario que creamos anteriormente.

A screenshot of the Django Admin login page. The title is "Administración de Django". It has two input fields: "Nombre de usuario:" with the text "jedag" and "Contraseña:" with masked characters ".....". Below the fields is a blue button labeled "Iniciar sesión".

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

Añadir estudiante

Dni:



ApellidoPaterno:

ApellidoMaterno:

Nombres:

FechaNacimiento: Hoy | 

Sexo: ▼

Carrera: ▼   

Añadir carrera

Codigo:

Nombre:

Duracion:

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

Seleccione carrera a modificar

Acción: ▼

- ☐ CARRERA
- ☒ Carrera object (432)

Agregamos la siguiente función `__str__` a la clase `carrera` para mostrar los

nombres formateados de cada objeto o registro encontrado:

```
class Carrera(models.Model): # Class that allows to create and implement the models
    codigo = models.CharField(max_length=3, primary_key=True) # text data
    nombre = models.CharField(max_length=50)
    duracion = models.PositiveSmallIntegerField(default=5) # numerical data

    def __str__(self): # Name each object found
        txt = "{0} {Duración: {1} años(s)}"
        return txt.format(self.nombre, self.duracion)
```

Reiniciamos todo ejecutando los siguientes comandos

```
PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad> python manage.py migrate
Operations to perform:
  Apply all migrations: Academica, admin, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.
PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad> python manage.py makemigrations
No changes detected
PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad> python manage.py migrate
Operations to perform:
  Apply all migrations: Academica, admin, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.
PS C:\Practicas_DJANGO\YOUTUBE\MiUniversidad> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 19, 2022 - 14:18:05
Django version 4.1, using settings 'MiUniversidad.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

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

Seleccione carrera a modificar

Acción:

<input type="checkbox"/>	CARRERA
<input checked="" type="checkbox"/>	Ingeniería de Software (Duración: 6 años(s))

1 carrera

Agregamos la función `__str__` a la clase **Estudiante** del mismo archivo **models.py**:

```
class Estudiante(models.Model):
    dni = models.CharField(max_length=8, primary_key=True)
    apellidoPaterno = models.CharField(max_length=35)
    apellidoMaterno = models.CharField(max_length=35)
    nombres = models.CharField(max_length=35)
    fechaNacimiento = models.DateField() # date data
    sexos = [
        ('F', 'Femenino'),
        ('M', 'Masculino')
    ]
    sexo = models.CharField(max_length=1, choices=sexos, default='F') # enumeration data
    carrera = models.ForeignKey(Carrera, null=False, blank=False, on_delete=models.CASCADE)
    vigencia = models.BooleanField(default=True) # boolean data

    def nombreCompleto(self): # Returns the full name in a given format
        txt = "{0} {1}, {2}"
        return txt.format(self.apellidoPaterno, self.apellidoMaterno, self.nombres)

    def __str__(self):
        txt = "{0} / Carrera: {1} / {2}"
        if self.vigencia:
            estadoEstudiante = "VIGENTE"
        else:
            estadoEstudiante = "DE BAJA"
        return txt.format(self.nombreCompleto(), self.carrera, estadoEstudiante)
```

Agregamos la función `__str__` a la clase **Curso**

```
class Curso(models.Model):
    codigo = models.CharField(max_length=6, primary_key=True)
    nombre = models.CharField(max_length=30)
    creditos = models.PositiveSmallIntegerField()
    docente = models.CharField(max_length=100)

    def __str__(self):
        txt = "{0} ({1}) / Docente: {2}"
        return txt.format(self.nombre, self.codigo, self.docente)
```

Agregamos la función `__str__` a la clase **Matricula**

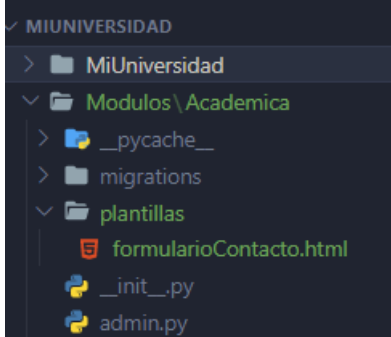
```
class Matricula(models.Model):
    id = models.AutoField(primary_key=True) # autoincrementing data
    estudiante = models.ForeignKey(Estudiante, null=False, blank=False, on_delete=models.CASCADE)
    curso = models.ForeignKey(Curso, null=False, blank=False, on_delete=models.CASCADE)
    fechaMatricula = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        txt = "{0} matriculad{1} en el curso {2} / Fecha: {3}"
        if self.estudiante.sexo == "F":
            letraSexo = "a"
        else:
            letraSexo = "o"
        fecMat = self.fechaMatricula.strftime("%A %d/%m/%Y %H:%M:%S")
        return txt.format(self.estudiante.nombreCompleto(), letraSexo, self.curso, fecMat)
```

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__`

5. 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

```

formularioContacto.html U x
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Formulario de Contacto</title>
8  </head>
9  <body>
10     <h1>Contáctenos</h1>
11     <form>
12         <p>
13             Asunto:
14             <input type="text" name="txtAsunto"/>
15         </p>
16         <p>
17             Email:
18             <input type="email" name="txtEmail"/>
19         </p>
20         <p>
21             Mensaje:
22             <textarea name="txtMensaje" cols="30" rows="5" style="resize: none;"></textarea>
23         </p>
24         <input type="submit" value="Enviar">
25     </form>
26 </body>
27 </html>
  
```

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

```

views.py M X
1  from django.shortcuts import render
2
3  # Create your views here.
4
5  def formularioContacto(request):
6      return render(request, "formularioContacto.html")
7

```

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

```

urls.py M X
16  from django.contrib import admin
17  from django.urls import path
18
19  from Modulos.Academica.views import formularioContacto
20
21  urlpatterns = [
22      path('admin/', admin.site.urls),
23      path('formularioContacto/', formularioContacto)
24  ]
25

```

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.

```

EXPLORADOR  settings.py M X
54
55  TEMPLATES = [
56      {
57          'BACKEND': 'django.template.backends.django.DjangoTemplates',
58          'DIRS': ['C:/Practicas_DJANGO/YOUTUBE/MiUniversidad/Modulos/Academica/plantillas'],
59          'APP_DIRS': True,
60          'OPTIONS': {
61              'context_processors': [
62                  'django.template.context_processors.debug',
63                  'django.template.context_processors.request',
64                  'django.contrib.auth.context_processors.auth',
65                  'django.contrib.messages.context_processors.messages',
66              ],
67          },
68      },
69  ]
70

```

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



The screenshot shows a web browser window with the title 'Formulario de Contacto'. The address bar displays '127.0.0.1:8000/formularioContacto/'. The main content area has a heading 'Contáctenos' and a form with three input fields: 'Asunto:', 'Email:', and 'Mensaje:'. Below the 'Mensaje:' field is an 'Enviar' button.

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

```
formularioContacto.html U X
11 <form action="/contactar/" method="POST">
12 {% csrf_token %}
```

El paquete `django.contrib.csrf` provee protección contra Cross-site request forgery (CSRF) (falsificación de peticiones inter-sitio).

Se presenta cuando un sitio Web malicioso induce a un usuario a cargar sin saberlo una URL desde un sitio al cual dicho usuario ya se ha autenticado, por lo tanto saca ventaja de su estado autenticado.

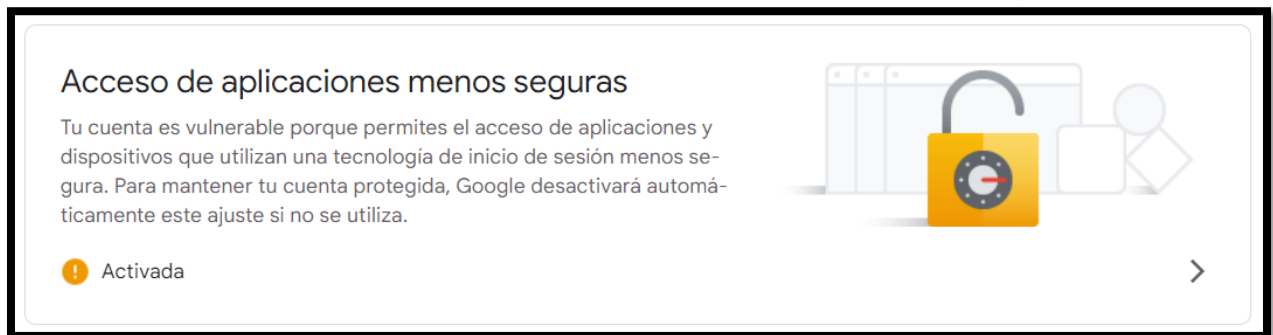
Agregamos la siguiente configuración en el archivo **settings.py** del proyecto para poder realizar envíos de correos electrónicos (después de la línea de **STATIC_URL**).


```
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 cuenta 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")
```