



GESTIONAR SISTEMA DE MENSAJES RECIBIDOS GUARDANDO LOS DATOS EN UNA BASE DE DATOS

ACTIVIDADES

DESARROLLAR UNA APLICACIÓN WEB UTILIZANDO EL FRAMEWORK DJANGO, IMPLEMENTANDO EL PATRÓN DE DISEÑO MVT (MODELO-VISTA-TEMPLATE), PARA GESTIONAR UN SISTEMA DE MENSAJES RECIBIDOS, GUARDANDO LOS DATOS EN UNA BASE DE DATOS Y DESPLEGANDO LOS MENSAJES A TRAVÉS DE UNA INTERFAZ DE USUARIO

- CREAR EL PROYECTO DJANGO
- DESARROLLO DEL MODELO
- IMPLEMENTACIÓN DE LA VISTA
- DISEÑO DE LA PLANTILLA
- CONFIGURACIÓN DEL PROYECTO
- CONTROL DE VERSIONES

MAXIMILIANO ARIEL FAVA



1- Creación del Proyecto:

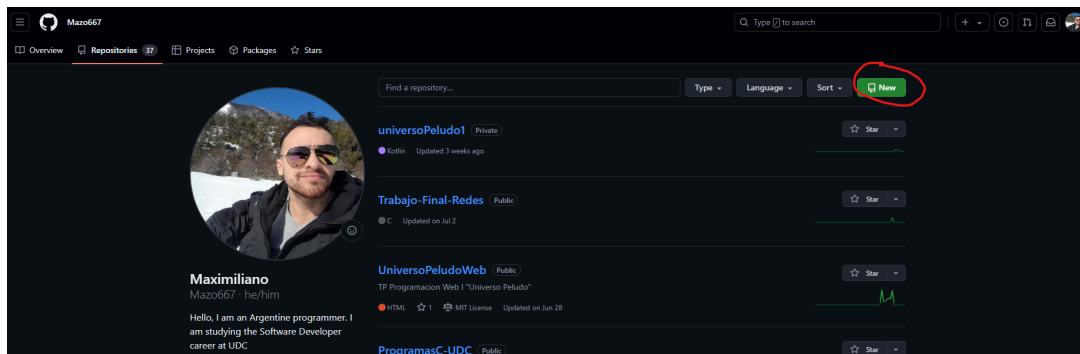
Se crea el entorno virtual del proyecto usando el comando: `python3 -m venv "nombre del proyecto"`.

```
root@Maximiliano: /home/pr x + v - □ ×
root@Maximiliano:/home/programacionweb2# python3 -m venv tplenv
root@Maximiliano:/home/programacionweb2# ls
tplenv
root@Maximiliano:/home/programacionweb2#
```

Una vez tengo mi entorno virtual creado, lo activamos para eso nos dirigimos a la carpeta bin y activamos.

```
root@Maximiliano: /home/pr x + v - □ ×
root@Maximiliano:/home/programacionweb2# source tplenv/bin/activate
(tplenv) root@Maximiliano:/home/programacionweb2#
```

El siguiente paso es ingresar a github.com, en caso de no tener cuenta se debe crear una; para crear un nuevo repositorio dentro de la pestaña de repositorios como se demuestra a continuación.

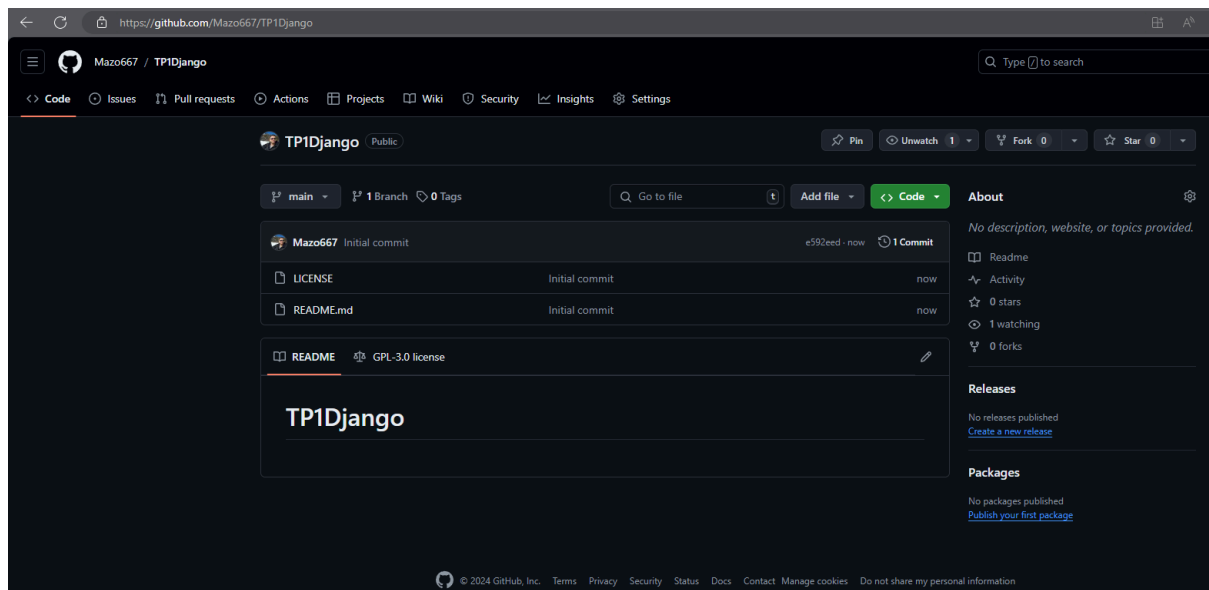


Una vez que hayamos hecho click en “new” nos abrirá una nueva página para configurar nuestro repositorio.

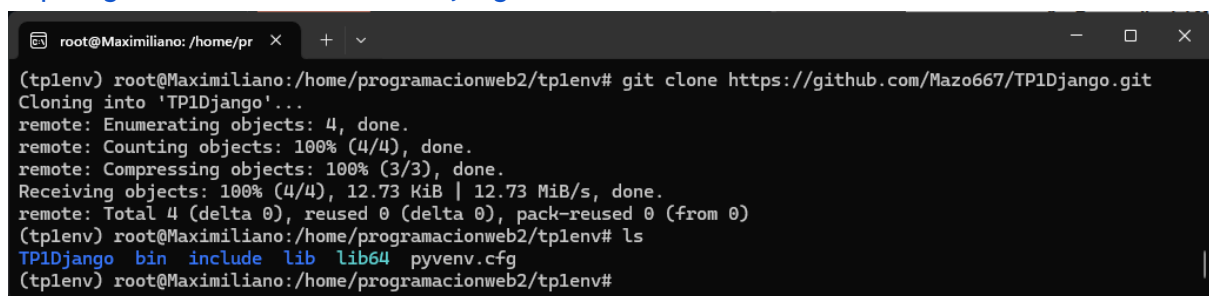
Le asignamos un nombre a nuestro repositorio, luego en este caso haremos que la visibilidad del repositorio sea público (que cualquiera pueda acceder), también tildamos la opción para agregar un archivo “*Readme*” para más tarde y como último paso agregare una licencia a nuestro repositorio, la que elijo va a hacer “*GNU General Public License v3.0*” para garantizar la libertad de usar, estudiar, compartir (copiar) y modificar el software.

A screenshot of the 'Create a new repository' form on GitHub. The form includes fields for Owner (Mazo667), Repository name (TP1Django), Description, Visibility (Public selected), Initialize this repository with (Add a README file selected), Add .gitignore, Choose a license (GNU General Public License v3.0 selected), and a 'Create repository' button. The form also includes a note about repository names and a link to learn more about READMEs.

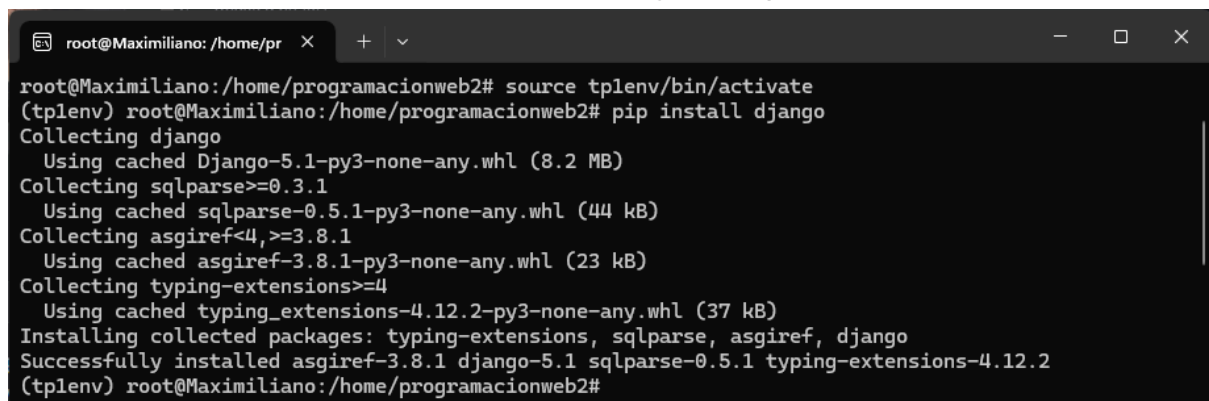
A partir de este momento ya tendremos nuestro repositorio en la nube como se puede ver a continuación...



Ahora clonaremos nuestro repositorio dentro de nuestro entorno virtual en mi caso es <https://github.com/Mazo667/TP1Django>

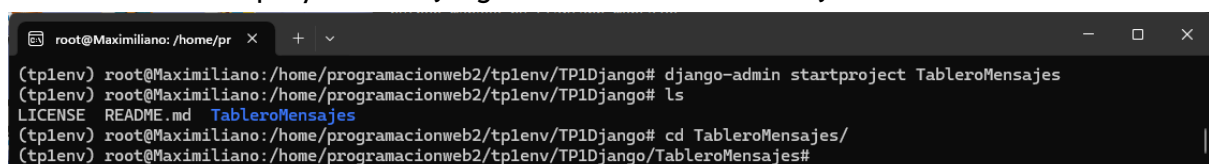


Dentro de nuestro entorno virtual instalamos Django con ayuda de pip.



Cabe aclarar que se instaló la versión django 5.1 debido a que a mi parecer es la versión recomendada por los desarrolladores de django y actualizada.

Crearemos nuestro proyecto de django llamado "TableroMensajes".



Dentro de nuestro proyecto crearemos una aplicación llamada “mensajes”. Con el archivo `manage.py` en el cual podremos crear nuevas aplicaciones, ejecutar el servidor, crear migraciones, ejecutar migraciones, etc..

```
root@Maximiliano: /home/pr x + v
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django# django-admin startproject TableroMensajes
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django# ls
LICENSE README.md TableroMensajes
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django# cd TableroMensajes/
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django/TableroMensajes# python manage.py startapp mensajes
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django/TableroMensajes#
```

Agregamos nuestros archivos a nuestro repositorio con `git add .` y verificamos con `git status`.

```
root@Maximiliano: /home/pr x + v
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django/TableroMensajes# git add .
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django/TableroMensajes# git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   TableroMensajes/__init__.py
    new file:   TableroMensajes/__pycache__/__init__.cpython-310.pyc
    new file:   TableroMensajes/__pycache__/settings.cpython-310.pyc
    new file:   TableroMensajes/asgi.py
    new file:   TableroMensajes/settings.py
    new file:   TableroMensajes/urls.py
    new file:   TableroMensajes/wsgi.py
    new file:   manage.py
    new file:   mensajes/__init__.py
    new file:   mensajes/admin.py
    new file:   mensajes/apps.py
    new file:   mensajes/migrations/__init__.py
    new file:   mensajes/models.py
    new file:   mensajes/tests.py
    new file:   mensajes/views.py

(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django/TableroMensajes#
```

Luego crearemos nuestro primer commit con `git commit`.

```
root@Maximiliano: /home/pr x + v
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django/TableroMensajes# git commit -m "Creamos el proyecto TableroMensajes con la aplicacion mensajes"
[main 94b5d19] Creamos el proyecto TableroMensajes con la aplicacion mensajes
15 files changed, 217 insertions(+)
create mode 100644 TableroMensajes/TableroMensajes/__init__.py
create mode 100644 TableroMensajes/TableroMensajes/__pycache__/__init__.cpython-310.pyc
create mode 100644 TableroMensajes/TableroMensajes/__pycache__/settings.cpython-310.pyc
create mode 100644 TableroMensajes/TableroMensajes/asgi.py
create mode 100644 TableroMensajes/TableroMensajes/settings.py
create mode 100644 TableroMensajes/TableroMensajes/urls.py
create mode 100644 TableroMensajes/TableroMensajes/wsgi.py
create mode 100755 TableroMensajes/manage.py
create mode 100644 TableroMensajes/mensajes/__init__.py
create mode 100644 TableroMensajes/mensajes/admin.py
create mode 100644 TableroMensajes/mensajes/apps.py
create mode 100644 TableroMensajes/mensajes/migrations/__init__.py
create mode 100644 TableroMensajes/mensajes/models.py
create mode 100644 TableroMensajes/mensajes/tests.py
create mode 100644 TableroMensajes/mensajes/views.py
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django/TableroMensajes#
```

Breve explicación del mensaje que se creó. “*master*” es el nombre de la rama en la que estamos trabajando, “(*root-commit*)” indica que es el primer commit en el repositorio y “*cf2d957*” es el hash único del commit, debido a que cada commit en Git tiene un identificador único para rastrear cambios. Después aparece que 15 archivos fueron modificados en este commit junto a 217 líneas de código añadidos a estos archivos. Y al final un listado de archivos creados o modificados en este commit.

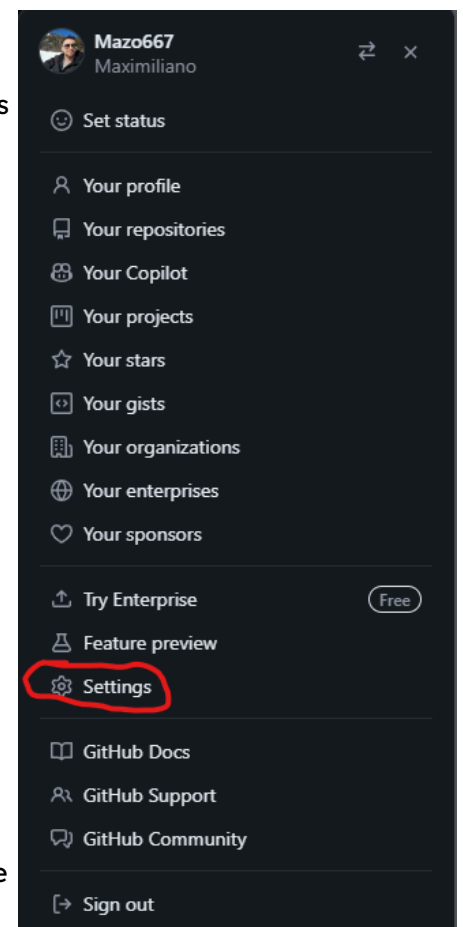
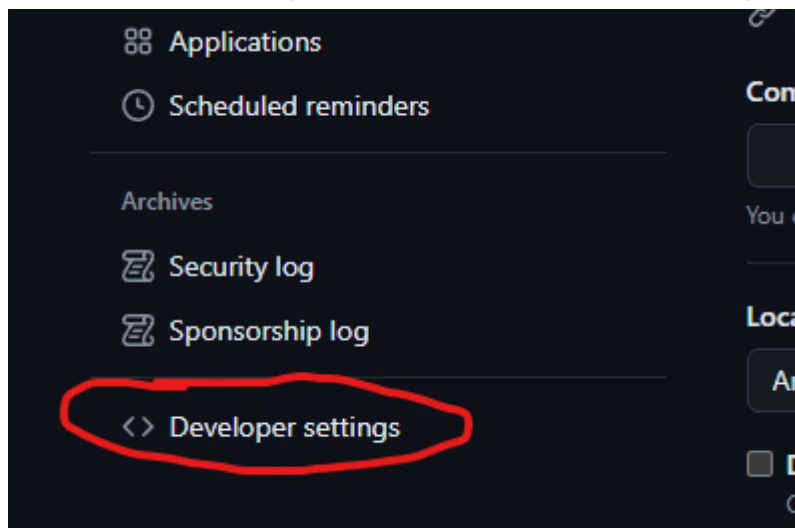
- *create mode 100644*: Indica que se ha creado un archivo con permisos de lectura y escritura para el propietario y solo lectura para el grupo y otros.
- *create mode 100755*: Indica que se ha creado un archivo ejecutable como “*manage.py*” que tiene permisos de lectura y escritura para el propietario.

Subiremos nuestro proyecto a github con el comando push de git. Cabe destacar que para que podamos hacer push git nos pedirá nombre de usuario y contraseña. Para la contraseña use tokens ¹.

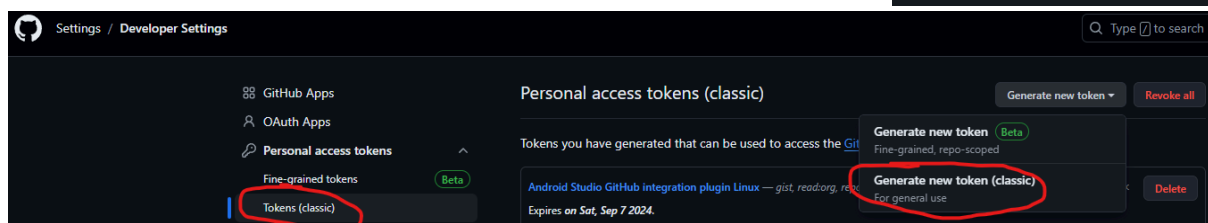
```
root@Maximiliano: /home/pr x + v
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django/TableroMensajes# git push -u origin main
Username for 'https://github.com': Mazo667
Password for 'https://Mazo667@github.com':
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 12 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (20/20), 5.08 KiB | 5.08 MiB/s, done.
Total 20 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Mazo667/TP1Django.git
140766e..94b5d19 main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
(tplenv) root@Maximiliano:/home/programacionweb2/tplenv/TP1Django/TableroMensajes#
```

Complementario

Para generar un token dentro de nuestro perfil de github nos vamos a la pestaña de settings. Seleccionamos Developer settings.



Dentro de estas opciones seleccionamos Tokens (classic) y generate new token (classic).



Dentro encontraremos una variedad de opciones, donde seleccionamos los permisos que tendrá dicho token.

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

Django

What's this token for?

Expiration

30 days The token will expire on Sat, Sep 21 2024

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys

Una vez seleccionado las opciones que necesitamos. Nos generará el token...

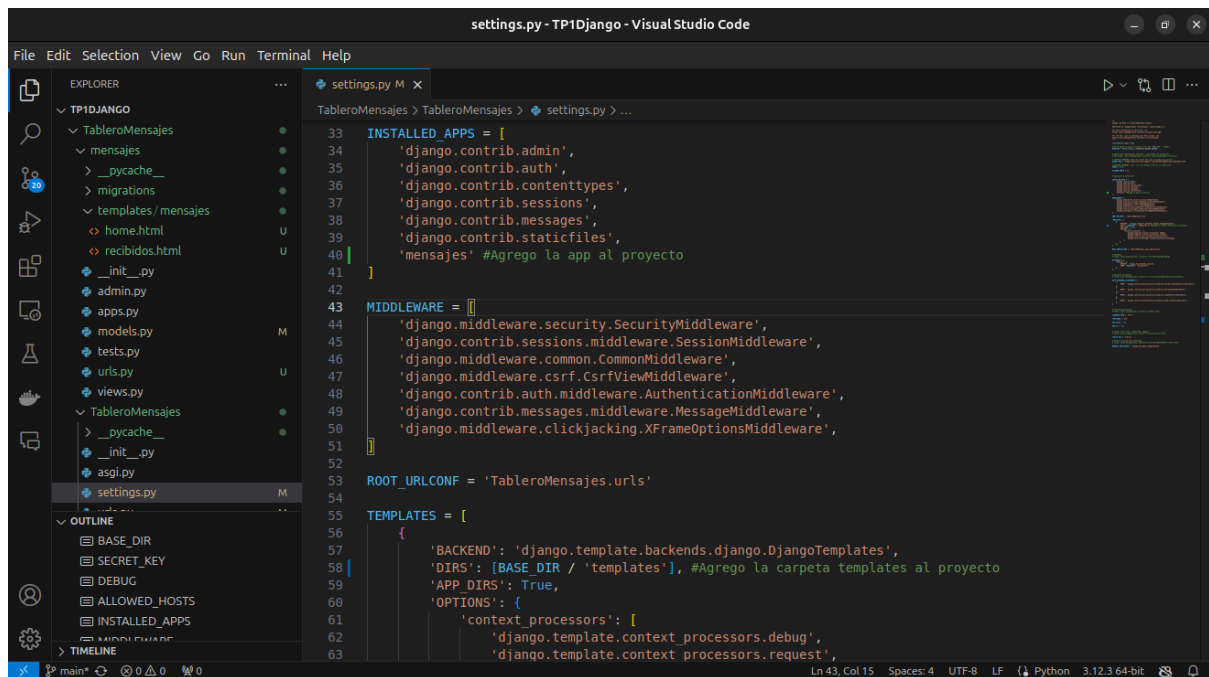
Personal access tokens (classic) Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

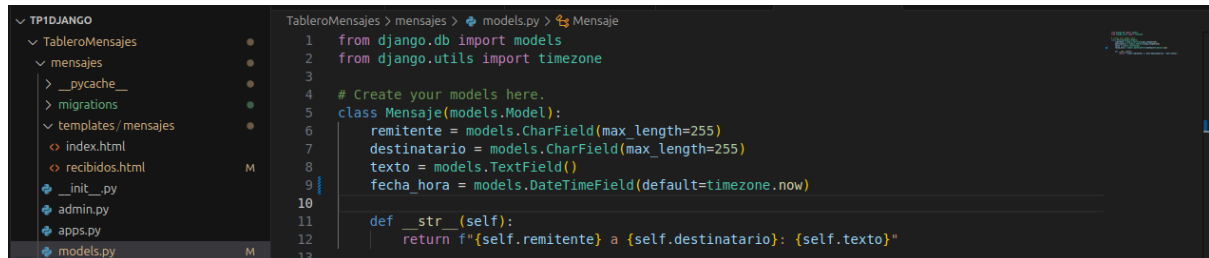
✓ ghp_Hu0CDxUmHUG1Z2qdNhjqdxVo01uG3r3cFV44 Copy Delete

El siguiente paso es dentro del archivo settings.py, agregar la app “Mensajes” al proyecto y agregar a la lista de direcciones la carpeta de templates.



2- Desarrollo del modelo:

Se crea el modelo “Mensaje”, dentro del archivo models.py.



Este modelo tendrá los siguientes atributos:

- texto: Donde se almacena el texto completo de cada mensaje.
- remitente: El usuario que envió el mensaje.
- destinatario: El usuario que recibió el mensaje.
- fecha_hora: La fecha y hora que se envió el mensaje.

3- Implementación de la Vista:

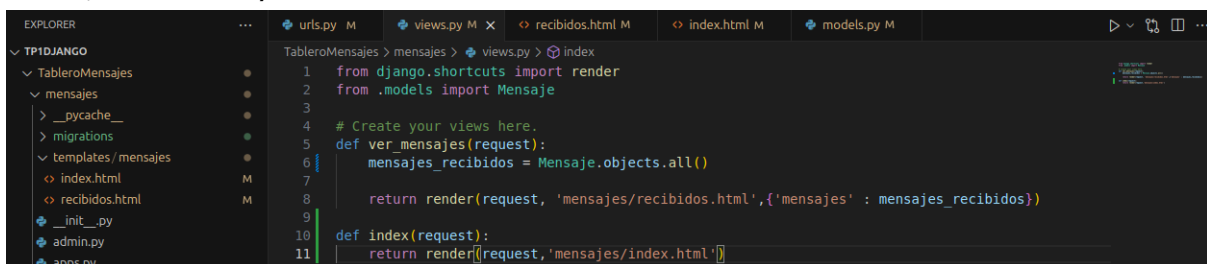
Creamos un archivo llamado “urls.py” en la carpeta de la aplicación “Mensajes”, donde declaramos el patrón de url que tendrá nuestra aplicación.



```
1 from django.urls import path
2 from .views import ver_mensajes, index
3
4 app_name = 'mensajes'
5
6 urlpatterns = [
7     path('', index, name='index'),
8     path('recibidos/', ver_mensajes, name='ver_mensajes'),
9 ]
```

Dentro de la lista de urlpatterns, declaramos los diferentes urls que tendrá nuestra aplicación y usaremos los templates de mensajes que creamos.

Luego dentro del archivo “views.py” de nuestra aplicación declaramos la vista de los mensajes recibidos y su indice.



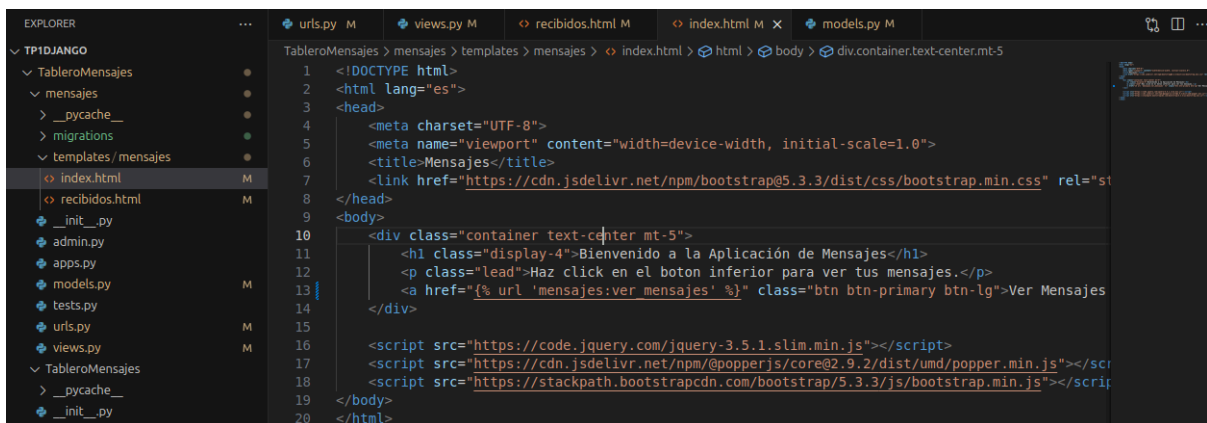
```
1 from django.shortcuts import render
2 from .models import Mensaje
3
4 # Create your views here.
5 def ver_mensajes(request):
6     mensajes_recibidos = Mensaje.objects.all()
7
8     return render(request, 'mensajes/recibidos.html', {'mensajes': mensajes_recibidos})
9
10 def index(request):
11     return render(request, 'mensajes/index.html')
```

4- Diseño de la Plantilla:

Dentro de la carpeta de “templates” crearemos una carpeta llamada “mensajes” donde irán los documentos html (nuestros templates) que le mostrarán al usuario los mensajes recibidos.

Primero haremos nuestro index de mensajes al ingresar a la url

“<http://127.0.0.1:8000/mensajes/>” donde nos mostrará un mensaje de bienvenida con un botón a los mensajes recibidos.

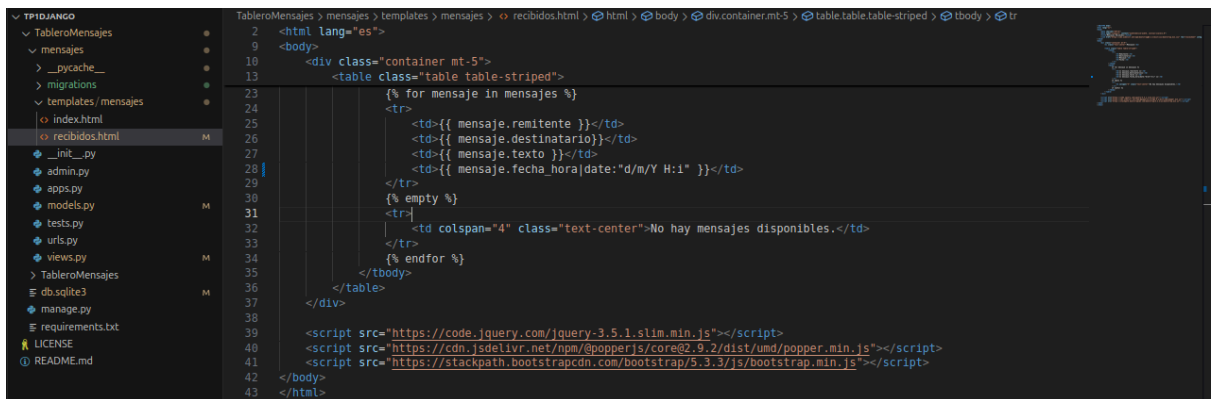


```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Mensajes</title>
7     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
8 </head>
9 <body>
10     <div class="container text-center mt-5">
11         <h1 class="display-4">Bienvenido a la Aplicación de Mensajes</h1>
12         <p class="lead">Haz click en el boton inferior para ver tus mensajes.</p>
13         <a href="{% url 'mensajes:ver_mensajes' %}" class="btn btn-primary btn-lg">Ver Mensajes
14     </div>
15
16     <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
17     <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"></script>
18     <script src="https://stackpath.bootstrapcdn.com/bootstrap/5.3.3/js/bootstrap.min.js"></script>
19 </body>
20 </html>
```

Nos quedaría así, donde al hacer click en el botón “Ver Mensajes Recibidos” nos llevará a la otra página donde podremos ver todos los mensajes.



Después crearemos otro template llamado “`recibidos.html`” donde nos mostrará todos los mensajes recibidos.



Y nos quedaría algo así...



En este caso que nuestra base de datos está vacía, no mostrará ningún mensaje.

Para poblar nuestra base de datos Django provee una API, Django provee 2 formas principales de interactuar con la base de datos, la primera usando el método “.save ()” de un objeto y el “object manager” manejador de objetos.

Para crear objetos en Django abrimos una shell que nos permitirá manipular objetos, por lo cual usaremos el comando `python manage.py shell`.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python - TableroMensajes + - [ ] [ ] ... [ ] [x]
(tplenv) maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp1env/TP1Django/TableroMensajes$ python manage.py shell
Python 3.12.3 (main, Jul 31 2024, 17:43:48) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from mensajes.models import Mensaje
>>> mensaje1 = Mensaje(texto="Hola como estas?", remitente='Samuel', destinatario='Ariel')
>>> mensaje1.save()
>>> mensaje2 = Mensaje(texto="Hola mañana estas libre?", remitente='Pablo', destinatario='Ariel')
>>> mensaje2.save()
>>> mensaje3 = Mensaje(texto="Mañana paso por vos", remitente='Pedro', destinatario='Ariel')
>>> mensaje3.save()
```

Con el resultado final...



Mensajes

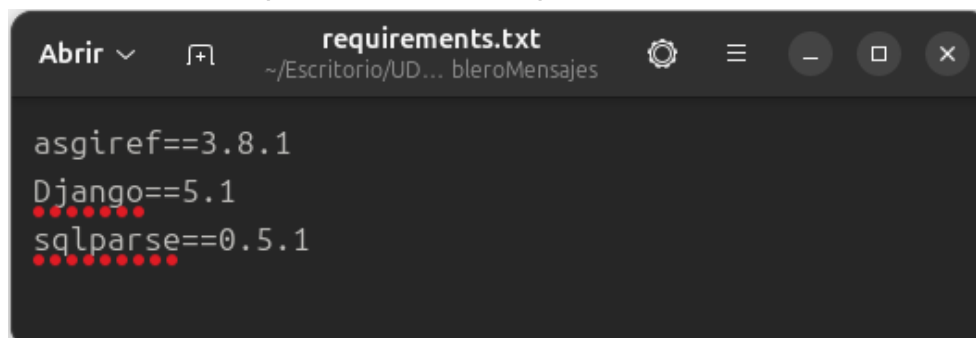
Remitente	Destinatario	Mensaje	Fecha
Samuel	Ariel	Hola como estas?	24/08/2024 00:00
Pablo	Ariel	Hola mañana estas libre?	24/08/2024 00:00
Pedro	Ariel	Mañana paso por vos	24/08/2024 00:00
Jose	Ariel	Buenos días!	25/08/2024 10:19
Alfredo	Jose	Podrias pasarme por la oficina mañana	25/08/2024 10:23

5- Configuración del Proyecto:

Como nuestro entorno virtual ya está creado, crearemos el archivo “requirements.txt” que contenga todas las dependencias necesarias para ejecutar tu proyecto Django para eso ejecutamos el siguiente comando `pip freeze > requirements.txt`

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp1env/TP1Django/TableroMensajes
(tplenv) maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp1env/TP1Django/TableroMensajes$ pip freeze > requirements.txt
(tplenv) maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp1env/TP1Django/TableroMensajes$ ls
(tplenv) maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp1env/TP1Django/TableroMensajes$
```

Como podemos ver ya se cree el archivo y lo abrimos..



Luego dentro del entorno virtual podremos usar el comando `pip install -r requirements.txt`, para instalar todas las librerías que hay dentro de ese archivo.

6- Control de Versiones:

El proyecto está subido a un repositorio de mi cuenta personal, en la siguiente URL:

<https://github.com/Mazo667/TP1Django>

Versiones Usadas:

- python 3.12
- Django 5.1
- pip 24.0
- virtualenv 20.25.0+ds

Software Usado:

- Visual Studio
- DB Browser for SQLite
- Terminal de Linux
- Git / GitHub

Bibliografía:

- Ultimate Django for Web App Development Using Python: Build Modern, Reliable and Scalable Production-Grade Web Applications with Django and Python. - Leonardo Luis Lazzaro
- Pasos a seguir para migrar el Modelo Tarea - Campus UDC - Julio Casco
- Encuentro sincronico PW2 - Introducción a GitHub - Primer Trabajo Práctico - 22 de agosto 2024
- Perplexity AI