

INFORME TRABAJO PRACTICO II

PROGRAMACION WEB II

PROGAMADO EN PYTHON



TABLERO MENSAJES FUNCIONALIDADES CLAVE COMO LA CREACIÓN, VISUALIZACIÓN Y ELIMINACIÓN DE MENSAJES

ACTIVIDADES

DESARROLLAR UN PROYECTO DJANGO DENOMINADO TABLEROMENSAJES SIGUIENDO EL PATRÓN DE DISEÑO MODELO-VISTA-PLANTILLA (MVT).

- CREACIÓN DE VISTAS BASADAS EN FUNCIONES Y EN CLASES
- USO DEL DJANGO TEMPLATE LANGUAGE (DTL)
- MODULARIZACIÓN
- REUTILIZACIÓN DE PLANTILLAS

MAXIMILIANO ARIEL FAVA



TEC. UNIV. EN DESARROLLO DE SOFTWARE

1. Configuración del Proyecto Django:

Se crea un entorno virtual para el proyecto usando el comando:

`virtualenv "nombre-entorno-virtual"`

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2
maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2$ virtualenv tp2env
created virtual environment CPython3.12.3.final.0-64 in 318ms
  creator CPython3Posix(dest=/home/maximiliano/Escritorio/UDC/ProgramacionWeb2/tp2env, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=/home/maximiliano/.local/share/virtualenv)
  added seed packages: pip==24.0
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2$ ls
clase20-08  tp1env  tp2env
maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2$
```

Ahora para crear nuestro proyecto de Django (5.1.1) necesitaremos instalar el mismo para eso accederemos a nuestro entorno virtual y con la ayuda pip instalaremos Django en nuestro entorno virtual.

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2
maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2$ source tp2env/bin/activate
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2$ pip install django
Collecting django
  Downloading Django-5.1.1-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.8.1 (from django)
  Using cached asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django)
  Using cached sqlparse-0.5.1-py3-none-any.whl.metadata (3.9 kB)
Downloading Django-5.1.1-py3-none-any.whl (8.2 MB)
 8.2/8.2 MB 3.3 MB/s eta 0:00:00
Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.5.1-py3-none-any.whl (44 kB)
Installing collected packages: sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.1.1 sqlparse-0.5.1
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2$
```

Antes de seguir avanzando localmente, crearemos nuestro repositorio en la página de GitHub.com. Aclarando las siguientes características.

- Nuestro repositorio será público.
- Agregaremos un archivo README para configurarlo más tarde.
- Nuestro repositorio tendrá la licencia GNU General Public Licence v3.0, para asegurarnos que sea de código abierto.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * / Repository name *
TP2WebII is available.

Great repository names are short and memorable. Need inspiration? How about [friendly-doodle](#) ?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
.gitignore template:

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License:

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set [main](#) as the default branch. Change the default name in your [settings](#).

Teniendo el repositorio creado haremos un git clone con la url de nuestro repositorio.

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env$ git clone https://github.com/Mazo667/TP2WebII.git
Clonando en 'TP2WebII'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Recibiendo objetos: 100% (4/4), 12.74 KiB | 143.00 KiB/s, listo.
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env$ ls
bin  lib  pyenv.cfg  TP2WebII
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env$
```

Una vez configurado y listo nuestro repositorio git, podremos seguir avanzando con nuestro proyecto de Django. Para eso usando el administrador de django crearemos nuestro proyecto “TableroMensajes”.

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII$ django-admin startproject TableroMensajes
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII$ ls
LICENSE  README.md  TableroMensajes
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII$
```

Con nuestro proyecto creado, nos desplazamos a la carpeta de nuestro proyecto y crearemos la app “mensajes”.

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$ cd TableroMensajes/
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$ python3 manage.py startapp mensajes
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$ ls
manage.py  mensajes  TableroMensajes
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$
```

Con nuestro proyecto y app creado dentro de nuestro repositorio, añadimos los archivos creados al repositorio local y verificaremos con git status.

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$ git add .
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$ git status
En la rama main
Tu rama está actualizada con 'origin/main'.

Cambios a ser confirmados:
(usa "git restore --staged <archivo>..." para sacar del área de stage)
nuevos archivos: TableroMensajes/__init__.py
nuevos archivos: TableroMensajes/__pycache__/__init__.cpython-312.pyc
nuevos archivos: TableroMensajes/__pycache__/settings.cpython-312.pyc
nuevos archivos: TableroMensajes/asgi.py
nuevos archivos: TableroMensajes/settings.py
nuevos archivos: TableroMensajes/urls.py
nuevos archivos: TableroMensajes/wsgi.py
nuevos archivos: manage.py
nuevos archivos: mensajes/__init__.py
nuevos archivos: mensajes/admin.py
nuevos archivos: mensajes/apps.py
nuevos archivos: mensajes/migrations/__init__.py
nuevos archivos: mensajes/models.py
nuevos archivos: mensajes/tests.py
nuevos archivos: mensajes/views.py

(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$
```

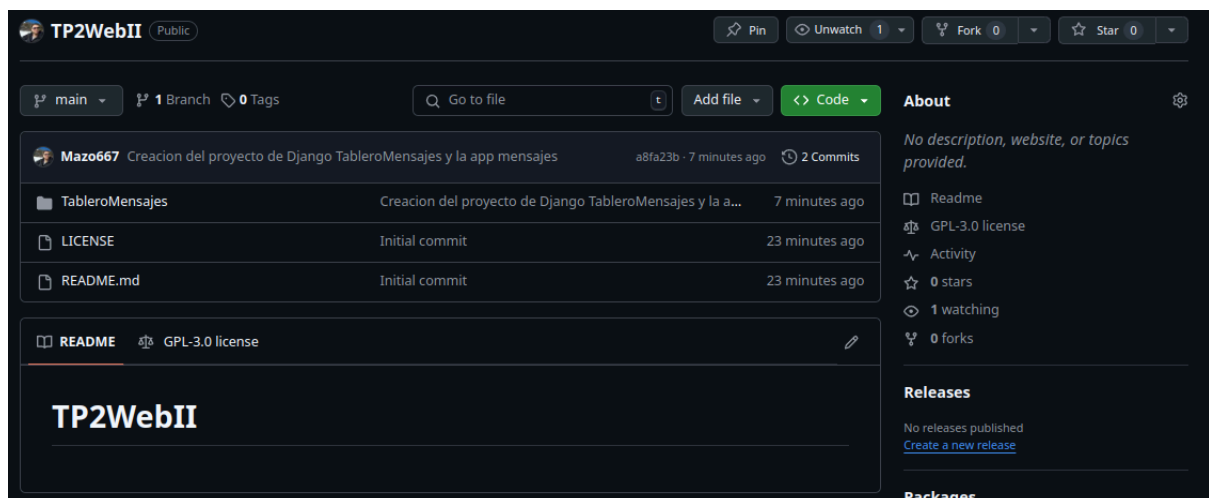
Crearemos nuestro primer commit con git commit.

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMen...
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMe
nsajes$ git commit -m "Creacion del proyecto de Django TableroMensajes y la app mensajes"
[main a8fa23b] Creacion del proyecto de Django TableroMensajes y la app mensajes
15 files changed, 217 insertions(+)
create mode 100644 TableroMensajes/TableroMensajes/__init__.py
create mode 100644 TableroMensajes/TableroMensajes/__pycache__/__init__.cpython-312.pyc
create mode 100644 TableroMensajes/TableroMensajes/__pycache__/settings.cpython-312.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
```

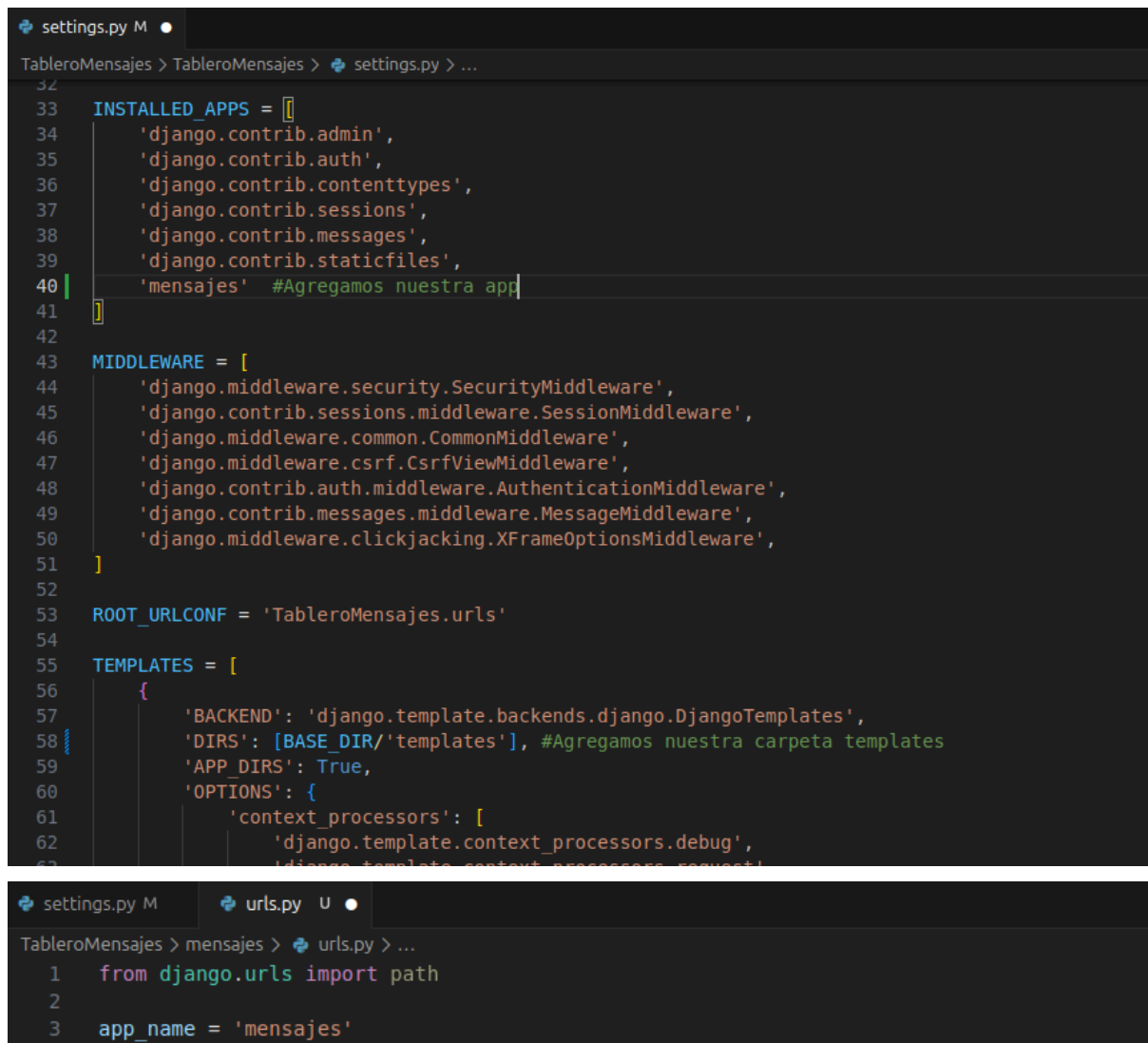
Luego haremos un push para subir los archivos de nuestro repositorio local a nuestro repositorio en GitHub.

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMen...
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMe
nsajes$ git push -u origin main
Username for 'https://github.com': Mazo667
Password for 'https://Mazo667@github.com':
Enumerando objetos: 21, listo.
Contando objetos: 100% (21/21), listo.
Compresión delta usando hasta 8 hilos
Comprimiendo objetos: 100% (18/18), listo.
Escribiendo objetos: 100% (20/20), 5.32 KiB | 1.33 MiB/s, listo.
Total 20 (delta 1), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Mazo667/TP2WebII.git
 625beb0..a8fa23b  main -> main
rama 'main' configurada para rastrear 'origin/main'.
(tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMe
nsajes$
```

Listo, logramos que nuestro proyecto “TableroMensajes” esté en nuestro repositorio GitHub.



El siguiente paso para finalizar con la configuración de nuestro proyecto es agregar en `settings.py` nuestra app, respetando el patrón Modelo-Vista-Plantilla crearemos una carpeta llamada `templates` con dejaremos todas nuestras plantillas y crearemos el archivo `urls.py` en nuestra app la cual configuramos mas adelante.



The image shows a code editor with two tabs: `settings.py` and `urls.py`. The `settings.py` tab is active, showing the following code:

```
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     'mensajes' #Agregamos nuestra app
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
52
53 ROOT_URLCONF = 'TableroMensajes.urls'
54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         'DIRS': [BASE_DIR/'templates'], #Agregamos nuestra carpeta templates
59         'APP_DIRS': True,
60         'OPTIONS': {
61             'context_processors': [
62                 'django.template.context_processors.debug',
63                 'django.template.context_processors.request',
```

The `urls.py` tab is also visible, showing the following code:

```
1 from django.urls import path
2
3 app_name = 'mensajes'
```


2. Implementación de Funcionalidades:

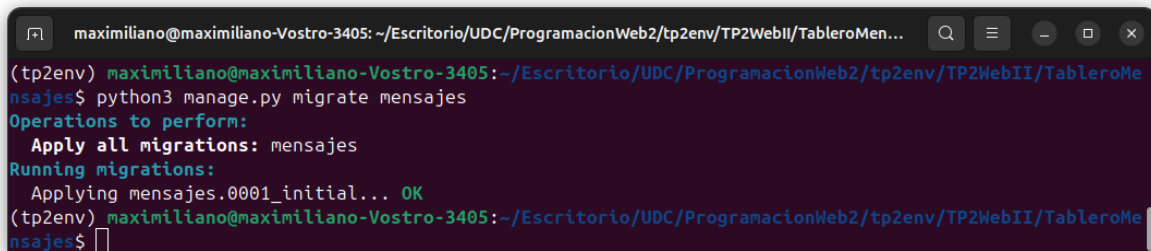
Como primer paso crearemos el modelo “Mensaje”, el cual estará conformado de la siguiente manera:

```
from django.db import models
from django.utils import timezone

class Mensaje(models.Model):
    remitente = models.CharField(max_length=255)
    destinatario = models.CharField(max_length=255)
    texto = models.TextField()
    fecha_hora = models.DateTimeField((auto_now=True))

    def __str__(self):
        return f"{self.remitente} a {self.destinatario}: {self.texto}"
```

Una buena práctica es que una vez hayamos creado el modelo hacemos las migraciones correspondientes para eso con el archivo `manage.py` ejecutaremos `makemigrations` y luego `migrate "nombre-app"`.

A screenshot of a terminal window with a dark background. The prompt is 'maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMe...'. The user enters 'python3 manage.py migrate mensajes'. The output shows 'Operations to perform: Apply all migrations: mensajes' and 'Running migrations: Applying mensajes.0001_initial... OK'. The prompt returns to 'nsajes\$'.

COMPLEMENTARIO:

El comando `makemigrations` crea nuevas migraciones basado en los cambios detectados en tus modelos. Django mantiene un registro de su esquema en la tabla llamada `django_migrations` y `makemigrations` básicamente, compara sus modelos con el esquema almacenado en esta tabla para decidir qué cambios deben realizarse.

Las migraciones también pueden se pueden revertir, deshaciendo los cambios aplicados.

```
python3 manage.py migrate "nombre-app" "0017" #Migró al estado 0017
```

```
python3 manage.py migrate "nombre-app" zero #Deshazo todos las migraciones para volver al inicio de la base de datos del esquema.
```

(Leondardo Luis Lazzaro, “Ultimate Django for Web App Development Using Python: Build Modern, Reliable and Scalable Production-Grade Web Applications with Django and Python”, 2024, Cap-4.Understanding Django Migrations).

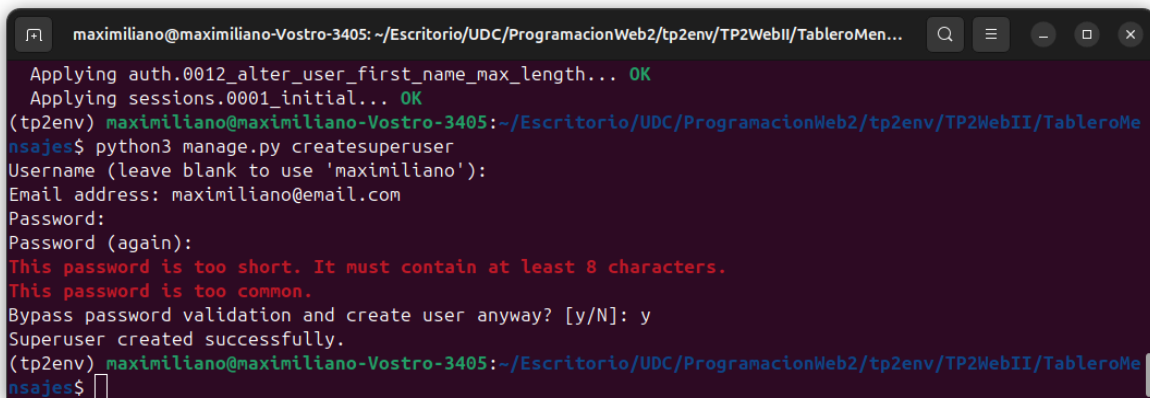
Ahora antes de darle funcionalidad a nuestro proyecto cabe resaltar que según (Leonardo Luis Lazzaro, 2024) podremos registrar nuestros modelos y manipular los datos. Para eso una alternativa al shell es abrir el archivo `mensajes/admin.py` y agregar la siguiente clase:

```
from django.contrib import admin
from mensajes.models import Mensaje

class MensajeAdmin(admin.ModelAdmin):
    list_display = ("remiteinte", "destinatario", "texto", "fecha_hora")

admin.site.register(Mensaje, MensajeAdmin)
```

Después haremos migraciones a nivel de proyecto y ejecutaremos el shell para crear un superusuario con un email y contraseña.

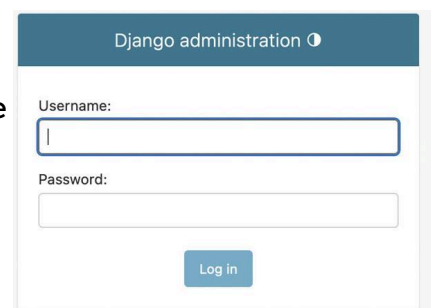
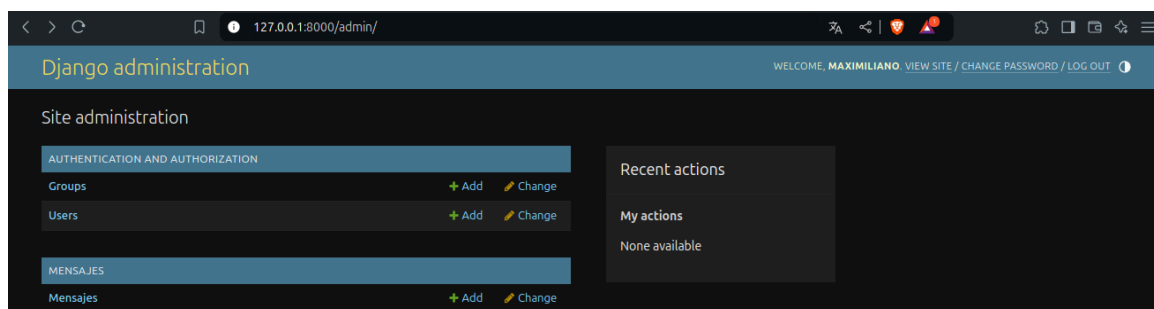
A terminal window with a dark background. The prompt is maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMe... The output shows two successful migrations: Applying auth.0012_alter_user_first_name_max_length... OK and Applying sessions.0001_initial... OK. Then, the command python3 manage.py createsuperuser is executed. The prompts are: Username (leave blank to use 'maximiliano'): nsajes\$, Email address: maximiliano@email.com, Password: nsajes\$, Password (again): nsajes\$. The output shows password validation errors: This password is too short. It must contain at least 8 characters. and This password is too common. Then, the prompt Bypass password validation and create user anyway? [y/N]: y is entered. The output shows Superuser created successfully. The final prompt is (tp2env) maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMe nsajes\$.

Ahora levantaremos el servidor e ingresaremos a

<http://127.0.0.1:8000/admin/>

Dentro de este panel nos pedirá nuestro usuario y contraseña que hemos creado.

Una vez que nos hayamos logueado veremos en el panel de administrador, el cual nos figurara nuestro modelo Mensaje.

A screenshot of the Django administration login page. The title is "Django administration". There are two input fields: "Username:" and "Password:". Below the fields is a "Log in" button.

Cuando hagamos click en Add nos llevará a otra página donde podremos crear un nuevo mensaje.

The screenshot shows the Django administration interface. The left sidebar has a menu with 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'MENSAJES' (Mensajes). The main content area is titled 'Change mensaje' and shows a form for editing a message. The form fields are: 'Remitente' (Alejandro), 'Destinatario' (Ariel), and 'Texto' (Hola como estas?). At the bottom, there are buttons for 'SAVE', 'Save and add another', 'Save and continue editing', and 'Delete'.

Cuando hayamos creado un nuevo mensaje, nos aparecerá en una lista.

The screenshot shows the Django administration interface displaying a list of messages. The table has columns: 'REMITENTE', 'DESTINATARIO', 'TEXTO', and 'FECHA HORA'. There is one message listed: 'Alejandro' to 'Ariel' with the text 'Hola como estas?' and timestamp 'Sept. 6, 2024, 10:04 a.m.'.

REMITENTE	DESTINATARIO	TEXTO	FECHA HORA
Alejandro	Ariel	Hola como estas?	Sept. 6, 2024, 10:04 a.m.

Verificando con el shell podremos ver que nuestro mensaje fue creado.

```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMen...
>>> from mensajes.Mensajes import Mensaje
Traceback (most recent call last):
  File "<console>", line 1, in <module>
ModuleNotFoundError: No module named 'mensajes.Mensajes'
>>> from mensajes.models import Mensaje
>>> mensajes = Mensaje.objects.all()
>>> print(mensajes)
<QuerySet [<Mensaje: Alejandro a Ariel: Hola como estas?>]>
>>>
```

Para este Trabajo Práctico respetando las consignas dadas, continuamos con el desarrollo de las funcionalidades.

Cabe destacar que podremos seguir experimentando, cómo agregar un estado a los mensajes como "archivado". También podemos crear grupos en Django para organizar y gestionar usuarios de manera más eficiente.

En Django, los grupos permiten agrupar usuarios según roles o permisos específicos. Estos grupos pueden ser útiles para asignar permisos a múltiples usuarios a la vez, en lugar de tener que hacerlo individualmente.

Django proporciona vistas basadas en clases para escenarios comunes, lo que le permite crear menos vistas de código.

Las vistas basadas en clases lo tientan a escribir la lógica de negocio en la implementación de la clase. Recuerde que las vistas basadas en clases siguen siendo vistas, así que evite escribir la lógica de negocios en vistas basadas en clases.

(Leondardo Luis Lazzaro, "Ultimate Django for Web App Development Using Python: Build Modern, Reliable and Scalable Production-Grade Web Applications with Django and Python", 2024, Cap-5.Introducing Django's Generic Views).

Empezamos a desarrollar nuestras clases vista para listar, crear y eliminar mensajes para eso crearemos las siguientes vistas basadas en clases en `views.py`:

```
from django.views import View
from django.shortcuts import render, redirect, get_object_or_404
from .models import Mensaje
from .forms import MensajeForm

def index(request):
    return render(request, 'mensajes/index.html')

def mensajeCreated(request):
    return render(request, 'mensajes/mensaje_created.html')

class MensajeListViewByUser(View):
    def get(self, request):
        usuario = request.GET.get('usuario', None)

        if usuario:
            mensajes_enviados = Mensaje.objects.filter(remitente__icontains=usuario)
            mensajes_recibidos = Mensaje.objects.filter(destinatario__icontains=usuario)
        else:
            return render(request, 'mensajes/mensaje_user.html', {'mensajes_enviados':
[], 'mensajes_recibidos': [], 'usuario': ''})

        return render(request, 'mensajes/mensaje_user.html', {'mensajes_enviados':
mensajes_enviados, 'mensajes_recibidos': mensajes_recibidos, 'usuario': usuario})

class MensajeListView(View):
    def get(self, request):
        mensajes = Mensaje.objects.all()
        return render(request, 'mensajes/mensaje_list.html', {'mensajes': mensajes})
```

```

class MensajeDetailView(View):
    def get(self, request, pk):
        mensaje = get_object_or_404(Mensaje, pk=pk)
        return render(request, 'mensajes/mensaje_detail.html', {'mensaje': mensaje})
class MensajeCreateView(View):
    def get(self, request):
        form = MensajeForm()
        return render(request, 'mensajes/mensaje_create.html', {'form': form})

    def post(self, request):
        form = MensajeForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('mensajes:mensaje_created')
        return render(request, 'mensajes/index.html', {'form': form})

class MensajeDeleteView(View):
    def post(self, request, pk):
        mensaje = get_object_or_404(Mensaje, pk=pk)
        mensaje.delete()
        return redirect('mensajes:mensaje_list')

```

Detallaremos cada una de las vistas:

- `def index(request):` Esta vista basada en una función, tiene el objetivo de mostrar el índice de la aplicación web. Donde dará un mensaje de bienvenida y dos botones, los cuales son “Ver mensajes” y “Crear un mensaje”.
- `def mensajeCreated:` Esta vista basada en una función es simple, su objetivo es mostrar que el mensaje fue creado correctamente con un botón para volver al índice.
- `class MensajeListView(View):` Esta vista basada en una clase, tiene como objetivos obtener todos los mensajes de la base de datos. Para pasarlos como parámetro a la plantilla `mensaje_list.html`.
- `class MensajeListViewByUser(View):` Esta vista basada en una clase tiene como objetivo obtener todos los mensajes de un determinado usuario.
- `class MensajeCreateView(View):` Esta vista basada en una clase, tiene dos objetivos, primero obtener el formulario de mensaje (explicado más adelante) y pasarlo como parametro a la plantilla `mensaje_create.html`. Una vez que el formulario sea validado se crea el mensaje y se redirige a la vista `mensajeCreated`.
- `class MensajeDeleteView(View):` Esta vista basada en una clase tiene como objetivo recibir un id de un mensaje y en caso de encontrarlo eliminarlo y redirigir a la misma página.

A continuación crearemos un archivo llamado `forms.py` para la creación de mensajes pero antes vamos a entender los formularios en Django.

El uso de formularios nos permite capturar las entradas por parte del usuario en una aplicación web. El formulario puede contener varios elementos como text fields, checkboxes, radio buttons y mucho más.

Los elementos form, input, label, select y textarea son comúnmente usados en formularios HTML. El elemento de formulario contiene todas las entradas y elementos de etiqueta. Tiene 2 atributos esenciales: action y method.

Action es la URL donde la data del formulario es enviado y el method se refiere a los métodos HTTP, el cual pueden ser POST o GET. Si bien estos son cruciales para la funcionalidad básica del formulario, otros atributos como 'enctype' pueden ser necesarios dependiendo de los requisitos específicos del formulario, como la carga de archivos.

Los elementos de entrada tienen tres atributos esenciales: type, name y value. El tipo determina qué entrada se mostrará: texto, correo electrónico, contraseña, casilla de verificación, opción de radio o envío. El atributo de nombre especifica el nombre de los datos enviados cuando se envía el formulario y el valor contiene el valor inicial opcional de la entrada.

(Leondardo Luis Lazzaro, "Ultimate Django for Web App Development Using Python: Build Modern, Reliable and Scalable Production-Grade Web Applications with Django and Python", 2024, Cap-7.Understanding Django Forms).

A continuación desarrollaremos el formulario de nuestra app.

```
from django import forms
from .models import Mensaje

class MensajeForm(forms.ModelForm):
    class Meta:
        model = Mensaje
        fields = ['remitente', 'destinatario', 'texto']
        widgets = {
            'remitente': forms.TextInput(attrs={'class': 'form-control'}),
            'destinatario': forms.TextInput(attrs={'class': 'form-control'}),
            'texto': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),
        }
        labels = {
            'remitente': 'Remitente',
            'destinatario': 'Destinatario',
            'texto': 'Mensaje',
        }
```

Otra configuración importante es en el archivo `urls.py` de nuestra aplicación para enrutamiento del mismo. Por lo quedará conformado de la siguiente manera:

```
from django.urls import path
from .views import MensajeCreateView, MensajeDeleteView, MensajeListView,
MensajeDetailView, MensajeListViewByUser, index, mensajeCreated

app_name = 'mensajes'

urlpatterns = [
    path('', index, name='index'),
    path('mensajes/', MensajeListView.as_view(), name='mensaje_list'),
    path('mensajes/<int:pk>', MensajeDetailView.as_view(), name='mensaje_detail'),
    path('mensajes/usuario/', MensajeListViewByUser.as_view(), name='mensaje_by_user'),
    path('mensaje/crear', MensajeCreateView.as_view(), name='mensaje_create'),
    path('mensaje/creado/', mensajeCreated, name='mensaje_created'),
    path('mensajes/eliminar/<int:pk>', MensajeDeleteView.as_view(), name='mensaje_delete')
]
```

Dentro de nuestra lista de urls encontraremos 5 paths, explicando cada uno de ellos:

- `''`: Este path es nuestro inicio y mostrará la vista `index`.
- `'mensajes/'`: Este path mostrará la vista `mensaje_list`, que contiene una lista de todos los mensajes.
- `'mensajes/crear'`: Este path permite crear un mensaje mostrando la vista `mensaje_create`.
- `'mensajes/creado'`: Este path mostrará que el mensaje fue creado con éxito.
- `'mensajes/usuario'`: Este path mostrará los mensajes de un usuario.
- `'mensajes/eliminar/<int:pk>'`: Este path recibirá un número entero que representa el id del mensaje a eliminar. En caso que exista será eliminado.

Según (*Leondardo Luis Lazzaro, Chapter 5-Handling Dynamic URLs with Path Converters*) `<int:pk>` es un convertidor de path. Primero **“int”** es el tipo a convertir, diciéndole al framework que tipo es. La segunda parte **“pk”** sirve como nombre de la variable a la que se asigna el valor coincidente; esta variable luego se pasa como parámetro a la vista.

Django tiene varios convertidores integrados: `string`: coincide con cualquier cadena no vacía, excepto el separador de ruta, `'/'`.

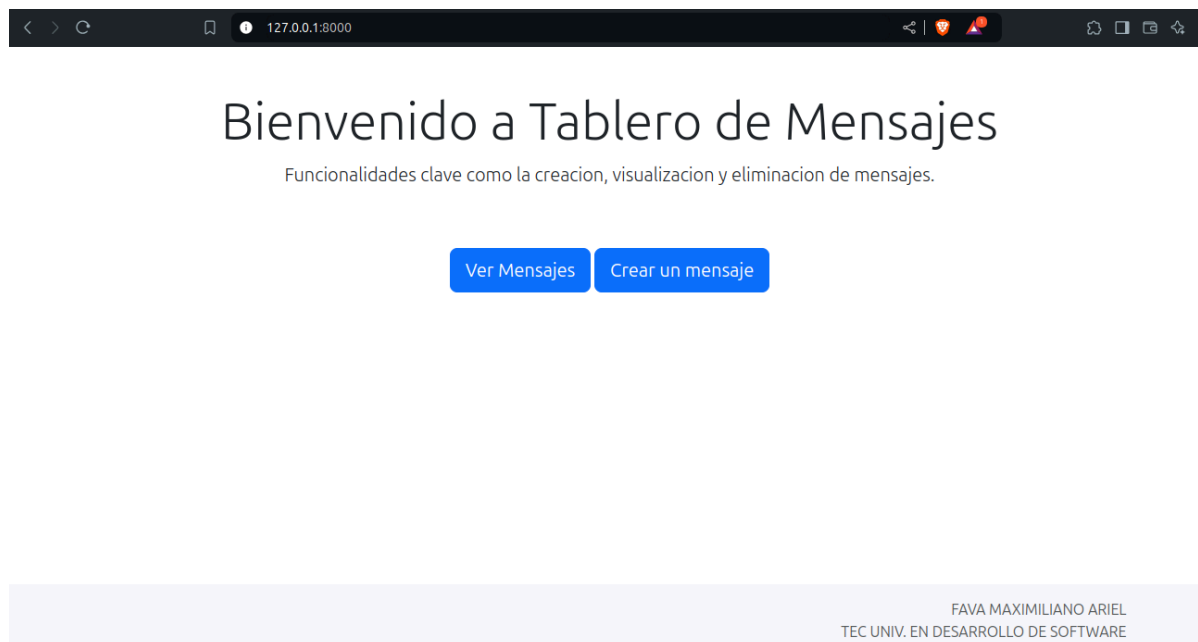
- **int**: coincide con cero o cualquier entero positivo.
- **slug**: coincide con cualquier cadena alfanumérica ASCII, guiones y guiones bajos.
- **uuid**: coincide con un UUID formateado.
- **path**: coincide con cualquier cadena no vacía, incluido el separador de ruta, `'/'`.

3. Estructura Plantillas

Las plantillas que creamos para este proyecto son las siguientes:

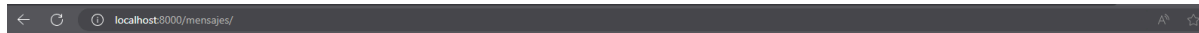
- `base.html`: La cual tiene un documento HTML básico con estilos, que incluye el uso de librería bootstrap, usando lenguaje DTL 'Django Template Language', esta plantilla **será la plantilla base** que incluirá los blocks `header_title` y un `header_description` los cuales describirán para que sirve cada página. Luego dentro de un container tendremos nuestro block `content` dónde irá el contenido de cada plantilla.
- `index.html`: Esta plantilla tiene la función de ser la página de inicio de la aplicación de mensajes.
- `mensaje_create.html`: Esta plantilla es para la creación de un nuevo mensaje.
- `mensaje_created.html`: Esta plantilla es para mostrar un mensaje de que el mensaje fue creado correctamente y un botón que permite volver al inicio.
- `mensaje_detail.html`: Esta plantilla cumple la función de mostrar un solo mensaje según su id.
- `mensaje_list.html`: Esta plantilla hace una lista de todos los mensajes que hay y la opción de eliminar el mensaje con un botón.

`index.html`



Página principal de nuestra aplicación que nos permitirá elegir entre ver todos los mensajes o crear un mensaje.

mensaje_list.html



Lista de Mensajes

Aquí puedes ver los mensajes enviados y recibidos.

Mensajes

Remitente	Destinatario	Mensaje	Fecha	Hora	
Alejandro	Ariel	Hola como estas?	06/09/2024	10:04	Eliminar
Ramon	Horacio	Mañana estas para jugar futbol a las 17:00 horas?	07/09/2024	03:11	Eliminar
Dario	Oscar	Cuanto sale el kilo de asado?	07/09/2024	04:15	Eliminar
Ariel	Leonardo	Mañana vas al colegio?	07/09/2024	23:28	Eliminar
Federico	Felipe	Hola Felipe, ¿Como estas?	07/09/2024	23:37	Eliminar
Felipe	Federico	Hola Federico, bien y tu?	07/09/2024	23:38	Eliminar
Federico	Felipe	Yo también bien, che ya terminaste el proyecto?	07/09/2024	23:39	Eliminar
Felipe	Federico	Casi me falta revisar algunos detalles y a vos?	07/09/2024	23:39	Eliminar
Federico	Felipe	Me estaría faltando el ultimo punto como lo hiciste?	07/09/2024	23:40	Eliminar
Felipe	Federico	La verdad te paso un libro de Leonardo Luis Lazzaro que explica muy bien y puede ayudarte. Mas precisamente en el capitulo 8.	07/09/2024	23:42	Eliminar

Podremos ver todos los mensajes que se encuentran en la base de datos, además tenemos la opción de elegir los mensajes enviados o recibidos por un usuario.

Maximiliano	Nicolas	¡De una! Nos vemos en 5 minutos.	07/09/2024	23:49	Eliminar
Carlos	Pepe	Esta noche estas disponible?	08/09/2024	00:56	Eliminar

[Volver al menu principal](#)

[Buscar por usuario](#)

FAVA MAXIMILIANO ARIEL
TEC UNIV. EN DESARROLLO DE SOFTWARE

mensaje_create.html



Creacion de Mensaje

Para crear un mensaje por favor complete todos los campos

Remitente:

Destinatario:

Mensaje:

[Enviar](#)

[Volver a la lista de mensajes](#)

FAVA MAXIMILIANO ARIEL
TEC UNIV. EN DESARROLLO DE SOFTWARE

Este formulario nos permitirá crear un mensaje, al hacer clic en “Enviar”.

mensaje_created.html

127.0.0.1:8000/mensaje/creado/

Mensaje Creado con Exito

Haga click en el boton para regresar

Volver

FAVA MAXIMILIANO ARIEL
TEC UNIV. EN DESARROLLO DE SOFTWARE

mensaje_user.html

localhost:8000/mensajes/usuario/?usuario=Maximiliano

Lista de Mensajes por Maximiliano

Aquí puedes ver los mensajes enviados y recibidos por Maximiliano.

Buscar por usuario, destinatario o remitente

Buscar

Mensajes Enviados por Maximiliano

Remitente	Destinatario	Mensaje	Fecha	Hora	
Maximiliano	Nicolas	Dale, ¿qué onda? ¿Qué necesitas saber?	07/09/2024	23:48	Eliminar
Maximiliano	Nicolas	¡De una! Nos vemos en 5 minutos.	07/09/2024	23:49	Eliminar

Mensajes Recibidos por Maximiliano

Remitente	Destinatario	Mensaje	Fecha	Hora	
Nicolas	Maximiliano	Maximiliano, tenemos que charlar sobre la reunión de mañana.	07/09/2024	23:48	Eliminar
Nicolas	Maximiliano	Estoy libre ahora, ¿quieres que nos juntemos?	07/09/2024	23:48	Eliminar

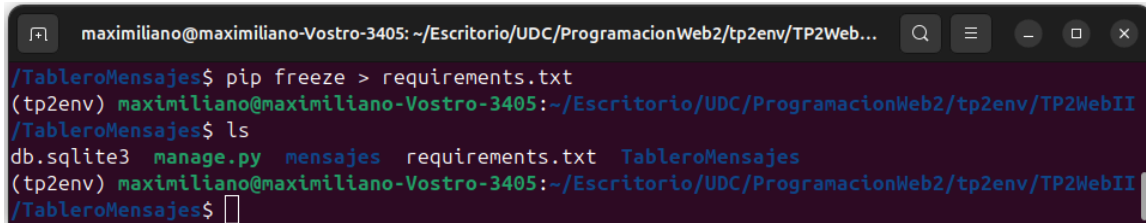
Volver

FAVA MAXIMILIANO ARIEL
TEC UNIV. EN DESARROLLO DE SOFTWARE

Aquí podremos buscar los mensajes por usuario.

4. Repositorio GitHub:

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`



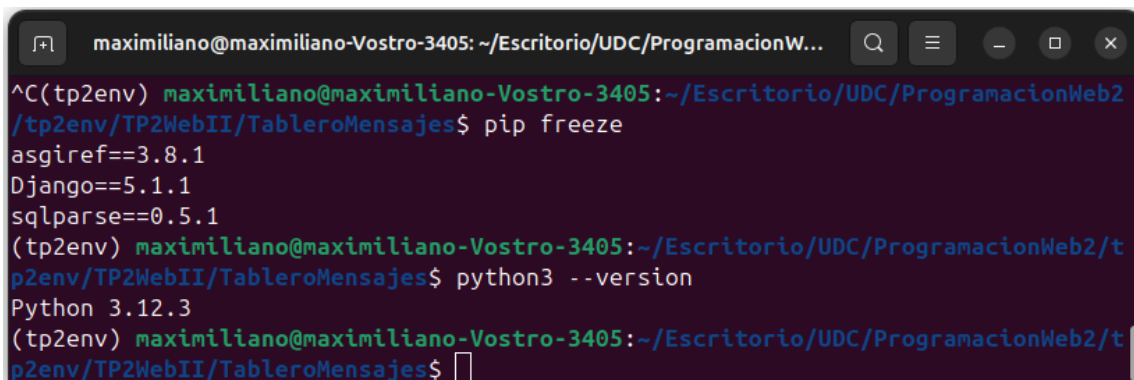
```
maximiliano@maximiliano-Vostro-3405: ~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$ pip freeze > requirements.txt
(maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$ ls
db.sqlite3  manage.py  mensajes  requirements.txt  TableroMensajes
(maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$
```

Y como desde el principio del proyecto se creo que repositorio Github con el archivo README.md, saltaremos esa parte.

Link al repositorio: <https://github.com/Mazo667/TP2WebII>

Versiones Usadas:

- python 3.12.3
- Django 5.1
- pip 24.0
- virtualenv 20.25.0+ds



```
^C(maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$ pip freeze
asgiref==3.8.1
Django==5.1.1
sqlparse==0.5.1
(maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$ python3 --version
Python 3.12.3
(maximiliano@maximiliano-Vostro-3405:~/Escritorio/UDC/ProgramacionWeb2/tp2env/TP2WebII/TableroMensajes$
```

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
- Plantillas en Django - Campus UDC - Julio Casco
- Vistas en Django - Uso de urls.py para el Enrutamiento - Campus UDC - Julio Casco
- Perplexity AI
- <https://getbootstrap.com/docs/5.3/getting-started/introduction/>