

Programación Web II

Trabajo Práctico N°2

Estudiante Enrique Armando Bogarin
2do Año - Tecnicatura Universitaria en Desarrollo de Software
Prof. y Lic. Julio César Casco

Repositorio del proyecto:
<https://github.com/Boga-in/WebII TP2>

Objetivos

*Desarrollar un proyecto Django siguiendo el patrón de diseño Modelo-Vista-Plantilla (MVT)

*Uso del Django Template Language (DTL), modularización y reutilización de plantillas

*Implementación de funcionalidades clave como la creación, visualización y eliminación de mensajes.

Introducción

En este proyecto planeo completar los objetivos listados con anterioridad en un proyecto Django “Tablero_de_Mensajes” donde se podrá listar los mensajes que podemos tanto crear como eliminar.

Comenzaré con un pequeño tutorial de como configurar el entorno y seguiré con el desarrollo del proyecto y la aplicación. La aplicación contará con un inicio como pantalla principal y un navbar que guiará al usuario a las funcionalidades.

Creación de Entorno y Proyecto

1- Crearemos un entorno virtual

```
entorno
boga@bogaSystemPc:~/Escritorio/EntornosVirtuales$ python3.11 -m virtualenv entTP2
created virtual environment CPython3.11.0.candidate.1-64 in 675ms
creator CPython3Posix(dest=/home/boga/Escritorio/EntornosVirtuales/entTP2, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/boga/.local/share/virtualenv)
added seed packages: pip==24.2, setuptools==72.1.0, wheel==0.44.0
activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
boga@bogaSystemPc:~/Escritorio/EntornosVirtuales$ ls
entorno  entTP2
boga@bogaSystemPc:~/Escritorio/EntornosVirtuales$
```

2- Instalaremos, ya dentro del entorno, la version Django que utilizaremos en nuestro proyecto

```
boga@bogaSystemPc:~$ source /home/boga/Escritorio/EntornosVirtuales/entTP2/bin/activate
(entTP2) boga@bogaSystemPc:~$ cd Escritorio/ProyectosDjango/
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango$ ls
'PWII TP1'  Tarea_De_Mensajes
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango$ pip install django==4.2
Collecting django==4.2
  Using cached Django-4.2-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref<4,>=3.6.0 (from django==4.2)
  Using cached asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django==4.2)
  Using cached sqlparse-0.5.1-py3-none-any.whl.metadata (3.9 kB)
Using cached Django-4.2-py3-none-any.whl (8.0 MB)
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-4.2 sqlparse-0.5.1
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango$
```

3- Crearemos nuestro repositorio de Github donde alojaremos nuestro proyecto

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 * Boga-in / Repository name * WebIITP2
WebIITP2 is available.

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

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: None

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

Choose a license

License: None

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](#).

 You are creating a public repository in your personal account.

Create repository

4- Haremos un clone de nuestro repositorio en la carpeta donde tendremos nuestro proyecto

```
Mira 'git help git' para una vista general del sistema.
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango$ git clone git@github.com:Boga-in/WebIITP2.git
Clonando en 'WebIITP2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Recibiendo objetos: 100% (3/3), listo.
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango$
```

5- Crearemos nuestro proyecto con el administrador de Django

```
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$ django-admin startproject Tablero_de_Mensajes .
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$ ls
manage.py  README.md  Tablero_de_Mensajes
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$
```

6- Seguiremos con la creación de de la App en la carpeta donde alojamos al proyecto

```
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$ python manage.py startapp mensajes
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$ ls
manage.py  mensajes  README.md  Tablero_de_Mensajes
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$
```

7- Haremos un add de todo el proyecto creado a Git

```
status
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$ git add .
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$ 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: Tablero_de_Mensajes/__init__.py
nuevos archivos: Tablero_de_Mensajes/__pycache__/__init__.cpython-311.pyc
nuevos archivos: Tablero_de_Mensajes/__pycache__/settings.cpython-311.pyc
nuevos archivos: Tablero_de_Mensajes/asgi.py
nuevos archivos: Tablero_de_Mensajes/settings.py
nuevos archivos: Tablero_de_Mensajes/urls.py
nuevos archivos: Tablero_de_Mensajes/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
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$
```

8- Entonces un commit con un mensaje correspondiente para luego hacer su debido push al repositorio

```

(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$ git commit -m "Primer commit - Proyecto
Tablero_de_Mensajes y app Mensajes"
[main 08c0e2c] Primer commit - Proyecto Tablero_de_Mensajes y app Mensajes
15 files changed, 217 insertions(+)
create mode 100644 Tablero_de_Mensajes/__init__.py
create mode 100644 Tablero_de_Mensajes/__pycache__/__init__.cpython-311.pyc
create mode 100644 Tablero_de_Mensajes/__pycache__/settings.cpython-311.pyc
create mode 100644 Tablero_de_Mensajes/asgi.py
create mode 100644 Tablero_de_Mensajes/settings.py
create mode 100644 Tablero_de_Mensajes/urls.py
create mode 100644 Tablero_de_Mensajes/wsgi.py
create mode 100755 manage.py
create mode 100644 mensajes/__init__.py
create mode 100644 mensajes/admin.py
create mode 100644 mensajes/apps.py
create mode 100644 mensajes/migrations/__init__.py
create mode 100644 mensajes/models.py
create mode 100644 mensajes/tests.py
create mode 100644 mensajes/views.py
(entTP2) boga@bogaSystemPc:~/Escritorio/ProyectosDjango/WebIITP2$ █

```

Con estos pasos podremos comenzar a desarrollar nuestra app.

Modelo de mensaje

Contará con cuatro atributos base, dos de ellos limitados a 100 caracteres cada uno y uno siendo un atributo de tipo DateTime al que se le asigna la fecha y hora del momento en que se cree un nuevo mensaje.

```

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

class Mensaje(models.Model):
    remitente = models.CharField(max_length=100)
    destinatario = models.CharField(max_length=100)
    texto = models.TextField()
    fecha_hora = models.DateTimeField(default=timezone.now)

```

Creación del Formulario

Para esta aplicación usaremos un formulario que reciba los datos requeridos correspondientes al modelo

```
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': 4}),
        }
        labels = {
            'remitente': 'Remitente',
            'destinatario': 'Destinatario',
            'texto': 'Mensaje',
        }
```

Funcionalidades implementadas

ver_mensajes : Esta función sirve para obtener todos los mensajes alojados en la base de datos y ser mostrados en la lista que utilizamos más adelante en un template.

```
def ver_mensajes(request):
    mensajes_recibidos = Mensaje.objects.all().order_by('fecha_hora')
    return render(request, 'recibidos.html', {'mensajes':
mensajes_recibidos})
```

crear_mensaje: Se llamará al formulario vacío al ser un método GET y cuando sea cargado podrá ser utilizada la función con un método POST donde lo colocado en el formulario, si es válido, se guardará en la base de datos y luego redirigirá al usuario a la lista de mensajes creados.

```
def crear_mensaje(request):
    if request.method == 'POST':
        form = MensajeForm(request.POST)
        if form.is_valid():
            form.save() # Guardar el mensaje en la base de datos
            return redirect('mensajes:ver_mensajes') # Redirigir a la
            lista de mensajes después de crear
        else:
            form = MensajeForm()

    return render(request, 'crear_mensaje.html', {'form': form})
```

eliminar_mensaje: Primero intenta obtener el mensaje que debe ser eliminado. Utiliza la función **get_object_or_404**, que busca en la base de datos el mensaje con el **mensaje_id** pasado como parámetro. Si el mensaje no existe, se lanzará una respuesta **404** (página no encontrada), lo que asegura que no se intente eliminar un mensaje que no existe. Permite a los usuarios eliminar un mensaje específico de la base de datos. En este caso usando el botón de confirmación de eliminación para asegurar que es lo deseado la eliminación del mensaje. Dependiendo de si la solicitud es de tipo **POST** o **GET**, la función muestra una confirmación o procede a eliminar el mensaje.

```
def eliminar_mensaje(request, mensaje_id):
    mensaje = get_object_or_404(Mensaje, id=mensaje_id)
    if request.method == 'POST':
        mensaje.delete()
        return redirect('mensajes:ver_mensajes') # Redirigir a la lista
        de mensajes después de eliminar

    return render(request, 'eliminar_mensaje.html', {'mensaje':
mensaje})
```

filtrar_mensajes: Intente utilizar otra forma de filtrar a los mensajes por los remitentes o destinatarios en un principio, pero al tener problemas cambie al filtro que use en el primer trabajo y lo mejore para que pueda filtrar por ambos valores e incluso al mismo tiempo.

```
def filtrar_mensajes(request):
    # Obtener listas de remitentes y destinatarios distintos
    remitentes = Mensaje.objects.values_list('remitente',
flat=True).distinct()
    destinatarios = Mensaje.objects.values_list('destinatario',
flat=True).distinct()

    mensajes_filtrados = None # Hice que venga vacía porque sino
mostraba todos los mensajes

    if request.method == 'POST':
        remitente_seleccionado = request.POST.get('remitente')
        destinatario_seleccionado = request.POST.get('destinatario')

        mensajes_filtrados = Mensaje.objects.all()

        if remitente_seleccionado:
            mensajes_filtrados =
mensajes_filtrados.filter(remitente=remitente_seleccionado)
        if destinatario_seleccionado:
            mensajes_filtrados =
mensajes_filtrados.filter(destinatario=destinatario_seleccionado)

    return render(request, 'filtrar.html', {
        'remitentes': remitentes,
        'destinatarios': destinatarios,
        'mensajes': mensajes_filtrados,
    })
```


ENLACES

A continuacion el archivo urls.py

```
from django.urls import path
from .views import (
    ver_mensajes, index, crear_mensaje,
    filtrar_mensajes, eliminar_mensaje
)

app_name = 'mensajes'

urlpatterns = [
    path('', index, name='index'),
    path('recibidos/', ver_mensajes, name='ver_mensajes'),
    path('crear/', crear_mensaje, name='crear_mensaje'),
    path('filtrar/', filtrar_mensajes, name='filtrar_mensajes'),
    path('eliminar/<int:mensaje_id>/', eliminar_mensaje,
name='eliminar_mensaje'),
]
```

Templates

1- Base.html : Es la plantilla base que sirve como el diseño principal de la aplicación. Contiene las partes comunes del sitio, como el encabezado, el pie de página, y la estructura básica del HTML. Otras plantillas extienden esta base y añaden contenido específico.

{% block content %} ...{% endblock %}: Definen una sección en la plantilla base que puede ser reemplazada por contenido específico en otras plantillas.

De esta misma, la primera vez que ingresamos a la aplicación, se renderiza la plantilla de index así mismo cuando seleccionamos la opción INICIO

2. recibidos.html: Muestra una tabla con los mensajes recibidos, permitiendo su visualización y eliminación

- `{{ mensaje.remitente }}`: Muestra el remitente del mensaje.
- `{{ mensaje.texto }}`: Muestra el texto del mensaje.
- `{{ mensaje.fecha_hora|date:"d/m/Y H:i" }}`: Muestra la fecha y hora formateada.
- `{% for mensaje in mensajes %}`: Itera sobre la lista de mensajes recibidos.
- `{% empty %}`: Muestra un mensaje alternativo si no hay mensajes.
- `{% csrf_token %}`: Se asegura de que el formulario de eliminación esté protegido contra ataques de falsificación de solicitudes entre sitios (CSRF).

[Inicio](#)

- [Mensajes Recibidos](#)
- [Crear Mensaje](#)
- [Filtrar Mensajes](#)

Tablero de Mensajes

Remitente	Destinatario	Mensaje	Fecha	Acciones
Bogarín	Franco	hola buenos dias	10/09/2024 20:09	<button>Eliminar</button>
Marco	Franco	sadasdasdasd	10/09/2024 20:47	<button>Eliminar</button>

App de Mensajes - Enrique Bogarin © 2024

3. `filtrar.html` : Esta plantilla muestra un formulario para que el usuario filtre mensajes por remitente o destinatario. Dependiendo de los filtros seleccionados, muestra una lista de mensajes que coinciden con los criterios de búsqueda.

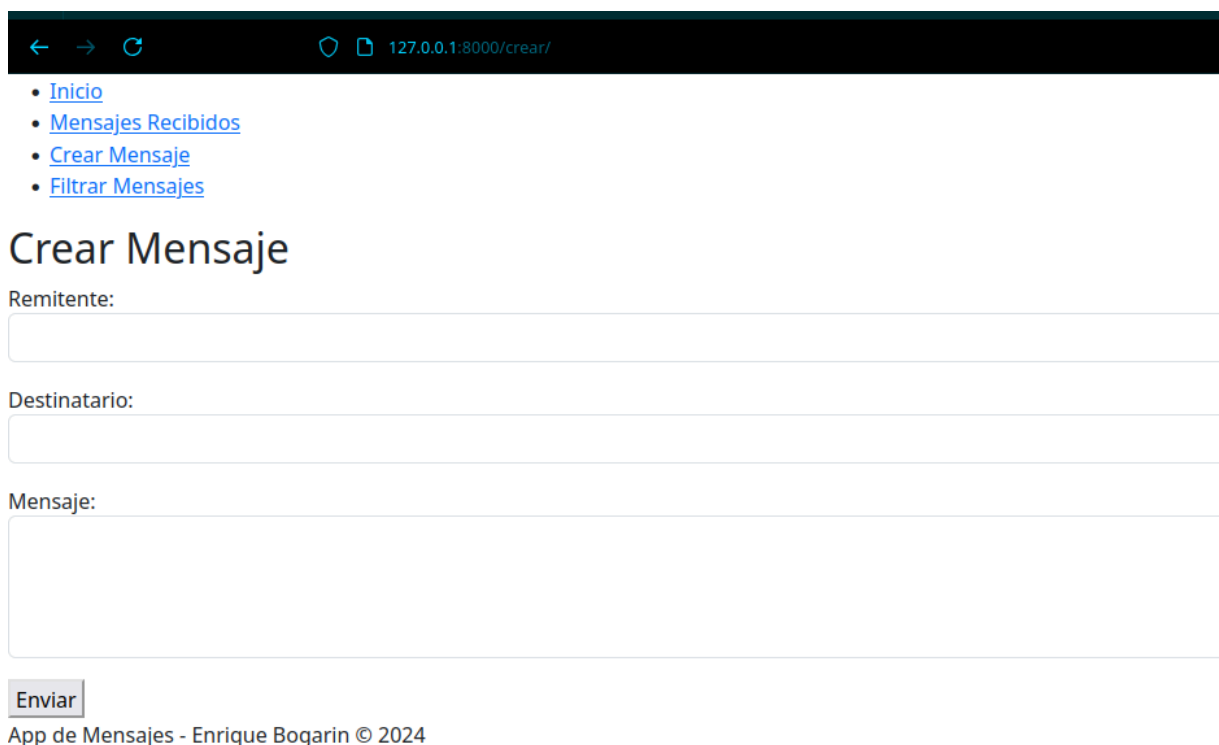
- `{% csrf_token %}`: Asegura la protección contra ataques CSRF al incluir un token en los formularios.
- `{% if mensajes %}`: Condicional que verifica si hay mensajes filtrados para mostrar.
- `{% for mensaje in mensajes %}`: Bucle para iterar y mostrar todos los mensajes que han sido filtrados.
- `{{ mensaje.remitente }}`, `{{ mensaje.destinatario }}`, etc.: Variables que se sustituyen con los datos de cada mensaje.



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/filtrar/`. The page has a dark header with navigation links: [Inicio](#), [Mensajes Recibidos](#), [Crear Mensaje](#), and [Filtrar Mensajes](#). Below the header, the main content area is titled "Filtrar Mensajes". It features two dropdown menus for filtering: "Filtrar por remitente:" with a dropdown menu showing "Seleccione un remitente", and "Filtrar por destinatario:" with a dropdown menu showing "Seleccione un destinatario". A "Filtrar" button is located to the right of the second dropdown. Below the filters, the section "Mensajes Filtrados" displays a list of messages. The first message shown has the following details: **Remitente:** Bogarin, **Destinatario:** Franco, **Mensaje:** hola buenos días, and **Fecha:** Sept. 10, 2024, 8:09 p.m. At the bottom of the page, the footer text reads "App de Mensajes - Enrique Bogarin © 2024".

4. `crear_mensaje.html`: Esta plantilla contiene un formulario para crear un nuevo mensaje. Utiliza un formulario generado por Django Forms para recolectar datos.

- `{{ form.as_p }}`: Renderiza los campos del formulario usando etiquetas `<p>` HTML.
- `{% csrf_token %}`: Protege el formulario contra ataques CSRF.



← → ↻ 127.0.0.1:8000/crear/

- [Inicio](#)
- [Mensajes Recibidos](#)
- [Crear Mensaje](#)
- [Filtrar Mensajes](#)

Crear Mensaje

Remitente:

Destinatario:

Mensaje:

Enviar

App de Mensajes - Enrique Bogarin © 2024

Versiones usadas

asgiref 3.8.1

Django 4.2

sqlparse 0.5.1

Python 3.11.0rc1

pip 24.2

Software Utilizado

Visual Studio Code

Terminal de Linux

Git hub

Bibliografía

Plantillas en Django - Campus UDC - Profesor Julio Casco

Clases Sincrónicas - Campus UDC - Profesor Julio Casco

Django, La guía Definitiva - Cap 4