Tutorial: Construcción de una aplicación Hola Mundo con Django y SQLite

Ingeniería de software I

Profesor: Oscar Eduardo Álvarez Rodríguez

Ladybugs

Universidad Nacional de Colombia

Bogotá

Tabla de contenido

1. Introducción

Dentro del desarrollo de software el saber construir una aplicación web es esencial, por esta razón hemos creado este tutorial el cual guía la construcción de una aplicación web simple, desde cero usando el framework **Django** de Python.

Como producto final obtendremos una aplicación que pueda:

- Conectarse a una base de datos.
- Crear e instanciar una entidad.
- Mostrar un mensaje en una interfaz web simple (Template) con un registro que provenga de la base de datos.

2. Requerimientos

Para el buen desarrollo de la aplicación usaremos las siguientes tecnologías:

Componente	Herramienta
Lenguaje	Python 3.10+
Framework	Django 5.x
ORM	ORM nativo de Django `(from django.db import models)`
Base de datos	SQLite
Servidor local	*runserver* de Django
Interfaz	HTML básico con plantilla Django
Editor de código	VS Code, PyCharm, Sublime Text, etc.
Conexión	Conexión a internet

3. Configuración del entorno

3.1. Crear entorno virtual

La creación de un entorno virtual es esencial, pues aislamos las dependencias del proyecto, evitando así problemas y conflictos a futuro.

En tu terminal, ejecuta:

```
None
mkdir hola_mundo_django      # Crear el directorio del proyecto
cd hola_mundo_django

python3 -m venv venv         # Crear el entorno virtual (venv)

source venv/bin/activate     # Activar el entorno virtual Linux/macOS (Bash)
.\venv\Scripts\activate      # Activar en Windows (CMD o PowerShell)
```

3.2. Instalar dependencias

Dentro de nuestro entorno virtual solo necesitamos instalar el framework principal (**Django)** lo cual se logra con la siguiente línea de código en tu terminal:

```
None
pip install django
```

3.3. Crear proyecto y aplicación

En Django, un Proyecto es el contenedor general y una Aplicación (app) es un módulo con una funcionalidad específica, luego procedemos a crearlos, en tu terminal ejecuta:

```
None
django-admin startproject mi_proyecto . # Crear el proyecto (el contenedor
                                        principal)

python manage.py startapp mi_app        # Crear la aplicación (el módulo de
                                        nuestro "Hola Mundo")
```

En resumen la estructura de nuestro proyecto se verá de esta manera:

```
None
hola_mundo_django/
|── mi_proyecto/
|    |── settings.py                  # Configuración principal
|    |── urls.py                      # URLS principales
|── mi_app/
|    |── models.py                    # ¡Aquí definiremos nuestra BD!
|    |── views.py                     # Lógica de la aplicación
|    |── ...
|── manage.py
```

```
|── venv/
|── db.sqlite3
```

4. Configuración base de datos (SQLite)

La configuración de la base de datos en este caso SQLite es muy sencilla, pues ya viene pre configurada por defecto en el archivo `mi_proyecto/settings.py` se ve de esta manera:

Python

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

En este punto debemos registrar nuestra nueva aplicación (`mi_app`) en la lista de aplicaciones instaladas. Dentro de `mi_proyecto/settings.py` agrega esta nueva línea:

Python

```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'mi_app.apps.MiAppConfig',              # <--- ¡AGREGA NUESTRO REGISTRO!
```

```
]
```

5. Uso del ORM de Django

En Django, se define nuestro esquema de base de datos (tablas) como clases de Python en el archivo `models.py`

5.1. Definición del modelo

En `mi_app/models.py`

```python
from django.db import models

# Definición de la entidad 'Mensaje HolaMundo' (será una tabla en SQLite)
class MensajeHolaMundo(models.Model):

    contenido = models.CharField(max_length=200)
    fecha_creacion = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Mensaje: {self.contenido}"
```

5.2. Crear la Tabla en SQLite (Migración)

Ahora le indicaremos a Django que cree la tabla `MensajeHolaMundo` en la base de datos `bd.sqlite3` .

De vuelta en tu terminal ejecutamos:

```
None

python manage.py makemigrations mi_app   # 1. Crear el archivo de migración
                                           (el "plan" de la tabla)

python manage.py migrate                 # 2. Aplicar la migración (ejecutar
                                           el "plan" en la BD)
```

5.3. Instalación mínima de entidades

Ahora usamos la `shell` de Python/Django para crear y guardar el primer registro directamente en la base de datos a través de ORM. En tu terminal ejecuta:

```
None

python manage.py shell            # Entrar a la shell interactiva de Django
```

Dentro de la `shell:`

```
Python

from mi_app.models import MensajeHolaMundo   # Importar el Modelo

# Crear y guardar un nuevo registro (Instanciación mínima de entidad)
primer_mensaje = MensajeHolaMundo(contenido="¡Hola Mundo desde Django y
SQLite!")
primer_mensaje.save()

print(MensajeHolaMundo.objects.all())        # Verificar que se haya
                                               guardado usando el ORM

exit()                                       # Salir de la shell
```

6. Crear Vista y Plantilla

Aquí vamos a implementar el patrón MVT (Modelo-Vista-Template)

6.1. La Vista (Lógica y Conexión de la base de datos)

La vista obtiene el registro de la base de datos usando el ORM y lo envía al Template.

Dentro de `mi_app/views.py`

```python
from django.shortcuts import render
from .models import MensajeHolaMundo     # Importamos nuestra entidad/Modelo

# Definición de la Vista basada en función
def index(request):

    # Conexión a la BD y consulta a través del ORM
    # .first() obtiene el primer registro disponible
    registro = MensajeHolaMundo.objects.first()

    # Preparamos el contexto para enviarlo al Template
    contexto = {
        'registro_db': registro.contenido if registro else "Error: No se
encontró registro en BD."
    }

    # Mostramos el template 'index.html' con los datos
    return render(request, 'index.html', contexto)
```

6.2. La plantilla

En este momento vamos a crear el archivo HTML que muestra el registro. Django busca templates en el directorio `mi_app/templates/` .

En `mi_app/templates/index.html`

HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Hola Mundo Django</title>
</head>
<body>
    <h1>Bienvenido a la App Django</h1>

    <div style="padding: 20px; border: 2px solid green; display:
inline-block;">
        <h2>Registro de la Base de Datos (a través de las capas MVT):</h2>
        <p style="font-size: 1.2em; font-weight: bold; color: navy;">
            {{ registro_db }}
        </p>
    </div>

    <br>
    <button onclick="alert('Funcionalidad simple de UI')">Click
aquí</button>
</body>
</html>
```

7. Configurar URLs

Necesitamos dos archivos `urls.py`: uno para la aplicación (`mi_app`) y otro para el proyecto (`mi_proyecto`).

7.1. URLs de la aplicación

En un nuevo archivo `mi_app/urls.py`

Python

```
from django.urls import path
from . import views
```

```
urlpatterns = [
    # Si la URL es la raíz de la app ('/') se llama a la función index del
views.py
    path('', views.index, name='index'),
]
```

7.2. URLs del Proyecto

En `mi_proyecto/urls.py` modifiquemos:

Python

```
from django.contrib import admin
from django.urls import path, include          # Importar include

urlpatterns = [
    path('admin/', admin.site.urls),
    # Conecta la URL base del proyecto ('') con las URLs de 'mi_app'
    path('', include('mi_app.urls')),          # <--- ¡AGREGA ESTA LÍNEA!
]
```

8. Ejecutar la aplicación

Listo!!!, llegó la hora de ejecutar el servidor de desarrollo de Django desde el directorio principal (`hola_mundo_django`).

En tu terminal ejecuta:

None

```
python manage.py runserver
```

Abre tu navegador y ve a la dirección `http://127.0.0.1:8000/`

Verás el título y el mensaje: **"¡Hola Mundo desde Django y SQLite!",** confirmando que el flujo completo **(URL → Vista → ORM → BD → Template)** ha funcionado.

9. Archivos .yml

Para esta configuración básica con Django y SQLite, no se requiere de un archivo .yml, si embargo en muchos proyectos estos archivos son usados para la configuración de Docker, veamos un ejemplo de estructura mínima si se usara:

```
None
# docker-compose.yml
version: '3.8'

services:
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    # Si se usara PostgreSQL o MySQL, se agregaría aquí el servicio de BD.
```