

django

Lunes 17 de Febrero de 2025
Píldora
Juan Domingo Ortín

¿Por qué se llama Django?

- El nombre **Django** fue elegido en honor al guitarrista **Django Reinhardt**, un famoso músico de jazz gitano del siglo XX.

¿Por qué ese nombre?

- Django **Reinhardt** era **rápido** y **elegante** tocando la guitarra, y el **framework** busca ser **rápido** y **eficiente** en desarrollo web.
- El creador del framework, Adrian Holovaty, era fanático del jazz y decidió rendir homenaje a Reinhardt.
- Creado en 2005 por **Adrian Holovaty** y **Simon Willison**.

Django Unchained

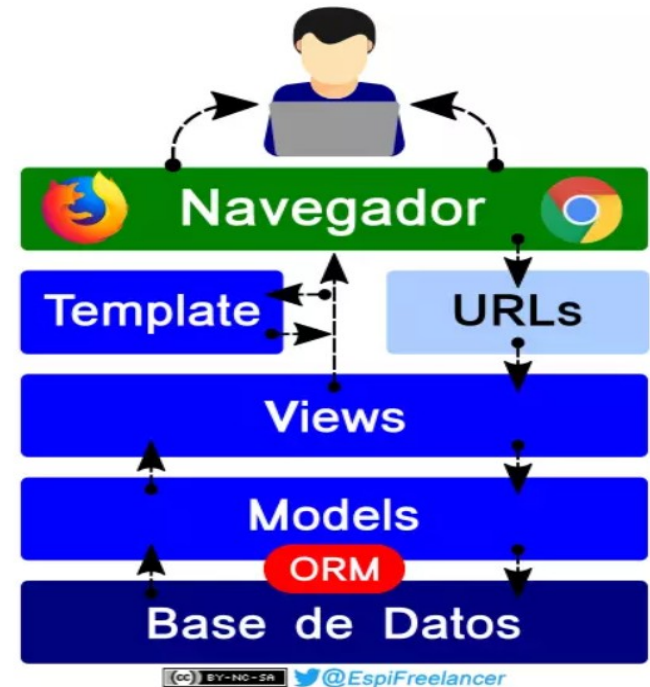


Django Reinhardt



¿Qué es Django?

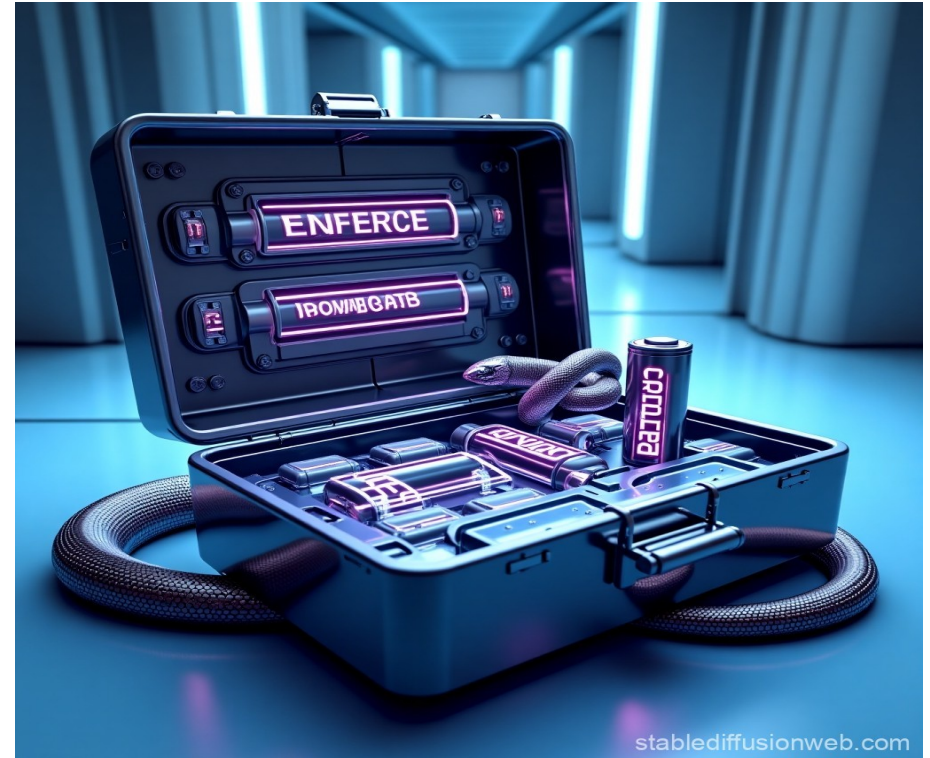
- Patrón **MTV** (Modelo-Template-Vista).
- Framework de **desarrollo**.
- **Open-source** y mantenido por la comunidad.
- Basado en **Python**, con enfoque en productividad.
- **Seguridad integrada** (protección contra SQL Injection, CSRF, XSS).
- **Escalabilidad y modularidad**.



Patrón MTV



¿Por qué usar Django?

- “Batería incluida”(batteries included) 🪫.
- ORM (Object-Relational Mapping) para bases de datos.
- Sistema de autenticación robusto.
- Panel de administración automático.
- Middleware para seguridad y rendimiento.



stablediffusionweb.com

Proyecto: "Task Manager" (Gestor de Tareas)

-  **Objetivo**
 - Crear una pequeña aplicación de gestión de tareas donde los usuarios pueden agregar, visualizar y marcar tareas como completadas.
-  **Características**
 - ✓ Usa SQLite para persistencia de datos.
 - ✓ Incluye un modelo "Task" para representar las tareas.
 - ✓ Usa el panel de administración para gestionar las tareas.
 - ✓ Despliegue gratuito y local en Windows (cmd) y Linux (terminal).

PASO 1: Creación de la carpeta padre

- Crear carpeta de proyecto:

```
mkdir "nombre-carpeta-proyecto"
```

- Entrar en la carpeta:

```
cd "nombre-carpeta-proyecto"
```


PASO 2: Creación del entorno virtual

- En Linux con `uv` :

```
uv venv --python python3.13 .venv  
source .venv/bin/activate  
uv init --bare
```

- En Linux con `venv` :

```
python3 -m venv .venv  
source .venv/bin/activate
```



- En Windows con `venv` :

```
python -m venv .venv  
.venv\Scripts\activate
```

PASO 3: Inicializar el Proyecto

Para crear un archivo `pyproject.toml` en un proyecto sin dependencias iniciales, se usa:

```
bash
```

 Copiar  Editar

```
uv init --bare
```

PASO 3: Inicializar el Proyecto

¿Qué hace este comando?

- ✓ `uv init` : Inicializa un nuevo proyecto en el directorio actual.
- ✓ `--bare` : Indica que no se agregarán dependencias por defecto, creando solo el archivo `pyproject.toml`.

¿Qué es `pyproject.toml` y para qué sirve?



Es un archivo de configuración estándar en proyectos Python que permite:

- Gestionar dependencias con `uv` o `pip`.
- Definir configuraciones del proyecto y herramientas como `black` o `pytest`.
- Mantener un entorno de desarrollo más organizado.



PASO 3: Inicializar el Proyecto

Ejemplo de un `pyproject.toml` generado por `uv init --bare`:

toml

 Copiar  Editar

```
[project]
name = "mi_proyecto"
version = "0.1.0"
```

 Este archivo es el equivalente a `requirements.txt`, pero con más funcionalidades y compatibilidad con herramientas modernas. 

PASO 4: Instalación Django Framework

- En Linux y Windows:

```
uv add django  
pip install django  
django-admin --version
```

PASO 5: Creación de proyecto Django 'taskmanager'

- Salimos a la carpeta raíz del proyecto y creamos el proyecto 'taskmanager':

```
$ django-admin startproject taskmanager  
$ cd taskmanager  
$ ls -l
```

```
lr-x 1 juandomingo juandomingo 667 feb 16 00:16 manage.py  
lr-x 2 juandomingo juandomingo 4096 feb 16 00:16 taskmanager
```

PASO 6: Servidor

- Hacemos correr el servidor para comprobar que funciona:

```
$ python manage.py runserver
```

```
Watching for file changes with StatReloader
```

```
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
[You have 18 unapplied migration(s). Your project may not work properly.
```

```
Run 'python manage.py migrate' to apply them.
```

```
Django version 5.1.6, using settings 'taskmanager.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CONTROL-C.
```

PASO 6: Servidor

- Realizamos las migraciones pendientes:

```
$ python manage.py migrate
```

- Ejecutamos el servidor de nuevo:

```
$ python manage.py runserver
```

```
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
Django version 5.1.6, using settings 'taskmanager.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```


PASO 7: Creación de la aplicación 'tasks'

- Generamos la aplicación dentro del proyecto:

```
python manage.py startapp tasks
```

- Registramos la aplicación en `taskmanager/settings.py`:

```
INSTALLED_APPS = [  
    "django.contrib.admin",  
    "django.contrib.auth",  
    "django.contrib.contenttypes",  
    "django.contrib.sessions",  
    "django.contrib.messages",  
    "django.contrib.staticfiles",  
    "tasks", # Agregamos nuestra aplicación  
]
```

PASO 8: Creación modelo 'Task'

- Definimos el modelo en `tasks/models.py`:

```
from django.db import models
class Task(models.Model):
    title = models.CharField(max_length=200)
    completed = models.BooleanField(default=False)
    def __str__(self):
        return self.title
```



PASO 8: Creación modelo 'Task'

- Aplicamos la migración para crear la tabla en SQLite:

```
$ python manage.py makemigrations  
$ python manage.py migrate
```

PASO 9: Registrar el Modelo en el Panel de Administración

- Editamos `tasks/admin.py`:

```
from django.contrib import admin
from .models import Task
admin.site.register(Task)
```

PASO 10: Superusuario

- Creamos un superusuario para acceder al panel de administración:

```
$ python manage.py createsuperuser
```

- Ingresamos `username`, `email` y `password`. Luego accedemos a `http://127.0.0.1:8000/admin`.

PASO 11: Mostrar lista de tareas

- Crear la Vista y Template para Mostrar Tareas:
- Editamos `tasks/views.py`:

```
from django.shortcuts import render
from .models import Task
def task_list(request):
    tasks = Task.objects.all()
    return render(request, "tasks/task_list.html", {"tasks": tasks})
```

PASO 11: Mostrar lista de tareas

- Creamos el template `task_list.html` en `tasks/templates/tasks/` :

```
<!DOCTYPE html>
<html>
<head>
  <title>Task Manager</title>
</head>
<body>
  <h1>Task List</h1>
  <ul>
    {% for task in tasks %}
      <li>{{ task.title }} - {% if task.completed %}
    {% endfor %}
  </ul>
</body>
</html>
```

PASO 12: Configurar las URL's

- Editamos `taskmanager/urls.py`:

```
from django.contrib import admin
from django.urls import path
from tasks.views import task_list
urlpatterns = [
    path("admin/", admin.site.urls),
    path("", task_list, name="task-list"),
]
```


PASO 13: Ejecutar el Servidor

- Iniciamos el servidor:

```
python manage.py runserver
```

- Si hay migraciones pendientes, aplicarlas:

```
python manage.py migrate
```

- Reiniciar el servidor si es necesario:

```
python manage.py runserver
```

- Accedemos a `http://127.0.0.1:8000/` para ver la lista de tareas.

The Django logo, featuring the word "django" in a white, lowercase, sans-serif font, centered on a dark green rectangular background.

django

FIN