**Investigación sobre Djando, Clases, Objetos, Librerías, Extensiones Y Creación De Aplicación Paso A Paso.**

Sebastián Jorge Peñuela

Servicio Nacional De Aprendizaje (SENA)

Análisis Y Desarrollo De Software (3066478)

Iván Malaver

Marzo 02, 2025

**Introduccion**

Como parte de mi interés en el desarrollo web, aquí podremos encontrar un framework de alto nivel escrito en Python que permite el desarrollo rápido y eficiente de aplicaciones web seguras y mantenibles. Me enfocaré en cómo Django utiliza clases y objetos para estructurar las aplicaciones, especialmente en los modelos que actúan como interfaz entre el código del servidor y la base de datos. Además, exploraré algunas de las librerías y extensiones más populares de Django, Mi objetivo es analizar en cómo estas características contribuyen a la flexibilidad y escalabilidad de las aplicaciones desarrolladas con Django, evaluando su impacto en el desarrollo eficiente de aplicaciones web.

**Definición Django**

Django es un framework web de código abierto escrito en Python, diseñado para facilitar el desarrollo rápido y seguro de sitios web complejos. Utiliza el patrón de diseño Modelo-Vista-Controlador (MVC) y se enfoca en el reusó y la extensibilidad de componentes. Actúan como interfaz entre la base de datos y el código del servidor, permitiendo el mapeo objeto-relacional. Facilitando la integración de funcionalidades adicionales y Gestiona el acceso y la seguridad de los usuarios, Permitiendo la creación de interfaces de usuario dinámicas e Incluye traducciones incorporadas, Django es ideal para proyectos que requieren desarrollo rápido y escalabilidad.

**Modelos de Django**

Los modelos en Django son clases de Python que representan la estructura de los datos de una aplicación web. Actúan como una interfaz entre el código del servidor y la base de datos, convirtiendo las tablas de la base de datos en clases u objetos en el código Python, lo que se conoce como mapeo objeto-relacional (ORM). Para definir un modelo en Django, se utiliza la clase **models.model** como base. Los campos de un modelo se definen usando clases de campo como **CharField**, **IntergerField, Date TimeField,** etc. Estos campos pueden tener argumentos opcionales **comonull, blank, default,** etc., para personalizar su comportamiento.

**Clases Django**

Las clases en Django se utilizan principalmente para definir modelos, vistas y otros componentes. Un modelo en Django es una clase que hereda de **models.model**, lo que le permite interactuar con la base de datos. Las vistas en Django también pueden ser definidas como clases, utilizando vistas basadas en clases que proporcionan métodos para manejar solicitudes HTTP. Los modelos en Django son clases que representan tablas en la base de datos. Cada modelo define campos que corresponden a columnas en la tabla.

**Objetos Django**

Los objetos en Django son instancias de las clases definidas. En el caso de los modelos, cada objeto representa una fila en la tabla de la base de datos. Por ejemplo, si tienes un modelo **Libro**. Ejemplo:

libro = Libro(titulo="El Señor de los Anillos", autor="J.R.R. Tolkien", fecha\_publicacion="1954-07-29")

libro.save()

Este objeto **Libro** es una instancia del modelo **Libro** y se almacena en la base de datos cuando se invoca el método save( ).

Asociación entre Clases: En Django, la asociación entre clases se logra mediante relaciones entre modelos. Por ejemplo, puedes relacionar un modelo **Artur** con un modelo **Libro** usando un campo **Foreignkey,** Ejemplo:

class Autor(models.Model):

nombre = models.CharField(max\_length=100)

class Libro(models.Model):

titulo = models.CharField(max\_length=100)

autor = models.ForeignKey(Autor, on\_delete=models.CASCADE)

**Librerías en Django**

Las librerías en Django se refieren a paquetes adicionales que se pueden instalar para ampliar las capacidades del framework. Estas librerías pueden ser utilizadas para tareas específicas como autenticación, APIs, procesamiento de datos, entre otras. Algunas librerías populares son las siguientes:

* **Django REST framework:** Permite crear APIs RESTful de manera eficiente, Ideal para proyectos que requieren una API robusta y escalable.
* **Django Channels:** Agrega soporte para aplicaciones en tiempo real, como chats o WebSockets, Útil para aplicaciones que requieren interacción en tiempo real.
* **Django Debug Toolbar:** Proporciona herramientas para depurar y analizar el rendimiento de la aplicación, Ayuda a identificar problemas de rendimiento y a optimizar el código.
* **Pillow:** Permite manipular imágenes, útil para aplicaciones que requieren procesamiento de imágenes.
* **Request:** Facilita la realización de peticiones HTTP, util para consumir APIs externas.

El uso de librerías en Django depende de la funcionalidad que desees implementar. Por ejemplo, si estás creando una API con Django REST Framework, definirías tus modelos y vistas de manera que se ajusten a la estructura de la API.

**Extensiones en Django**

Las extensiones en Django son paquetes adicionales que se pueden instalar para ampliar las capacidades del framework. Estas extensiones pueden mejorar la funcionalidad, el rendimiento y la facilidad de uso de tus proyectos Django. Algunas extensiones populares son las siguientes:

* **Django REST framework:** Permite crear APIs RESTful de manera eficiente, Facilita la creación de APIs robustas y escalables, lo que es crucial para aplicaciones que requieren interacción con otros servicios o aplicaciones móviles.
* **Django Haystack:**  Incorpora herramientas de búsqueda más robustas, Mejora la experiencia del usuario al proporcionar funcionalidades de búsqueda avanzadas, lo que es crucial para aplicaciones con grandes conjuntos de datos.
* **Django Debug Toolbar:**  Proporciona herramientas para depurar y analizar el rendimiento de la aplicación, Ayuda a identificar problemas de rendimiento y a optimizar el código, lo que es esencial para mantener una aplicación eficiente.
* **Django Extensions:** Añade nuevos comandos y funcionalidades al proyecto, como reconstruir la base de datos o exportar direcciones de correo electrónico, Incrementa la productividad durante el desarrollo al proporcionar herramientas adicionales para gestionar el proyecto.
* **Django Storages:**  Facilita el uso de servicios de almacenamiento en la nube, Permite escalar el almacenamiento de archivos multimedia sin afectar el rendimiento del servidor principal, lo que es ideal para aplicaciones que manejan grandes cantidades de datos.

Las extensiones de Django mejoran significativamente la funcionalidad del framework, permitiendo a los desarrolladores crear aplicaciones web complejas de manera más eficiente y escalable.

**Como se crea una aplicación Paso a Paso**

**Paso 1: Instalar Django**

1. Crea un entorno Virtual: **py -m venv env**

****

1. Activar el entorno Virtual**:** **env\Scripts\activate**



1. Instalar Django**:** **pip install django**

****

**Paso 2:** **Crea un Nuevo Proyecto de Django**

1. Crea un nuevo proyecto: **py -m django startproject myproject**

****

1. Entrar en el directorio del Proyecto: **cd myproject**

****

**Paso 3: Crea una nueva Aplicación**

1. Crea una nueva aplicación dentro del proyecto: **python manage.py startapp myapp**

****

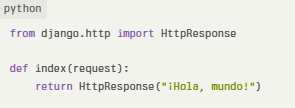
**Paso 4: Configurar la Aplicación en el proyecto**

1. **Abrir el Archivo ‘settings.py’** del proyecto **(‘myproject/settings.py’).**
2. **Agregar la aplicacion** a la lista **‘INSTALLED\_APPS’:**

****

**Paso 5: Crear Vistas y URls**

1. Crear una vista en ‘myapp/views.py’:



1. Crea un archivo ‘urls.py’ en la carpeta de la aplicación (‘myapp/urls.py’):

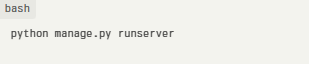


1. Incluir las URls de la aplicación en el archivo ‘urls.py’ del proyecto (‘myproject/urls.py’):



**Paso 6: Iniciar el servidor de Desarrollo**

1. **Ejecutar el servidor del desarrollo**

****

1. **Visitar la Aplicación en el navegador**

****

Con estos pasos tendremos una aplicación básica en Django Funcionando.

**Conclusión**

Mi investigación sobre Django ha revelado la potencia y flexibilidad que este framework ofrece a través de sus clases y objetos, He podido apreciar cómo las librerías y extensiones de Django, simplifican significativamente el desarrollo web al proporcionar funcionalidades predefinidas que reducen el tiempo de desarrollo. Además, he observado que la comunidad activa de Django es un factor clave en su éxito, ya que desarrolla y comparte constantemente nuevas herramientas y módulos. En resumen, mi experiencia con Django ha sido enriquecedora y me ha permitido entender mejor cómo este framework puede ser utilizado para desarrollar aplicaciones web robustas y eficientes. Además, he podido explorar cómo las extensiones adicionales pueden ampliar las capacidades de Django, permitiendo la creación de aplicaciones personalizadas según las necesidades específicas de cada proyecto. En general, creo que Django es una herramienta invaluable para cualquier desarrollador web que busque crear aplicaciones complejas de manera rápida y segura.

# Bibliografía

* *Amazon Web services (WAS)*. (02 de 03 de 2025). Obtenido de ¿Qué es Django? - Explicación del software Django: https://aws.amazon.com/es/what-is/django/#:~:text=Django%20es%20un%20software%20que,y%20la%20administraci%C3%B3n%20de%20cookies.
* *datacamp*. (2025). Obtenido de Tutorial de Django: Desarrollo web en Python - DataCamp: Tutorial de Django: Desarrollo web en Python - DataCamp
* *docs,djangoproject*. (2025). Obtenido de Documentación de Django: https://docs.djangoproject.com/es/5.1/
* *El libro Django.* (2025). funza: boblus.us.
* *ELPYTHONISTA*. (2025). Obtenido de Django - Framework para perfeccionistas con deadlines: https://elpythonista.com/django
* Holovaty, A. &.-M. (2009). *The Definitive Guide to Django: Web Development Done Right. Apress.*
* *openaceccess.ouc*. (2025). Obtenido de Django REST Framework (DRF) Secure Code Guidelines: https://openaccess.uoc.edu/bitstream/10609/147246/3/mnaderFMDP0123report.pdf
* *Python Web Development with Django. Pearson Education.* (2008). Obtenido de Chun, W. J. : https://dspace.tdea.edu.co/bitstream/handle/tdea/6989/SISTESIS.pdf?sequence=1&isAllowed=y
* *re-ma*. (2025). Obtenido de (IFCT095PO) Python y Django - Grupo Editorial RA-MA: https://www.ra-ma.es/libro/ifct095po-python-y-django\_129640/
* *Wikipedia*. (02 de 03 de 2025). Obtenido de Django (framework) : https://es.wikipedia.org/wiki/Django\_(framework)