

***Institución:*** Instituto Industrial Luis. A. Huergo

***Materia:*** Laboratorio de Algoritmos y Estructuras de Datos

***Docente:*** Ignacio Miguel García

***Estudiante:*** Teo Szychowski

**Resumen**

Este trabajo presenta una introducción al uso del *framework* Django en el desarrollo de aplicaciones web, abordando sus fundamentos y ventajas principales. Se explica su estructura basada en el patrón MTV (Modelo-Template-Vista), diferenciándolo del clásico patrón MVC. Además, se detallan conceptos esenciales como las *apps* modulares, el flujo *request-response*, el uso del ORM (Object-Relational Mapping) para la interacción con bases de datos, y la función de los *templates* para la presentación visual. También se describe el proceso de instalación necesario para comenzar a trabajar con Django. Este recorrido teórico tiene como propósito facilitar la comprensión de Django como herramienta eficiente, segura y escalable en entornos de desarrollo modernos.

*Palabras clave:* Django, desarrollo web, patrón MTV, ORM, templates, Python, instalación

**Abstract**

This paper presents an introduction to the Django framework in web application development, highlighting its main features and advantages. It explains its architecture based on the MTV pattern (Model-Template-View), and contrasts it with the traditional MVC approach. Key concepts are covered, such as modular apps, the request-response flow, the use of Object-Relational Mapping (ORM) to interact with databases, and templates for visual presentation. The installation process is also described to help beginners get started. This theoretical overview aims to support the understanding of Django as an efficient, secure, and scalable tool for modern development environments.

*Keywords:* Django, web development, MTV pattern, ORM, templates, Python, installation

**Introducción**

El presente trabajo tiene como propósito explorar y comprender los fundamentos del framework Django, orientado al desarrollo de aplicaciones web con el lenguaje de programación Python. Para lograrlo, se formularon una serie de preguntas clave relacionadas con sus componentes internos, su arquitectura, y su instalación, las cuales estructuran el cuerpo del documento. Paralelamente, se llevó a cabo la implementación práctica del tutorial oficial de Django, dividido en ocho partes, con el objetivo de reforzar los conceptos abordados desde la experiencia directa.

Django ha adquirido gran popularidad debido a su enfoque en la eficiencia, seguridad y escalabilidad. Entender cómo se estructura internamente y cómo interactúan sus elementos principales resulta fundamental para cualquier desarrollador que aspire a construir aplicaciones web robustas utilizando herramientas modernas. Este trabajo se enfoca, por tanto, en presentar tanto la base conceptual como la experiencia práctica que permite aplicar Django en proyectos reales.

**Método**

La metodología del presente trabajo se compone de dos ejes principales. En primer lugar, se realizó un abordaje teórico mediante un cuestionario que contempla los temas centrales del framework Django. Las preguntas fueron seleccionadas por el docente a cargo y responden a conceptos básicos necesarios para el entendimiento general del sistema. Estas respuestas fueron elaboradas en lenguaje académico, integrando definiciones claras y ejemplos explicativos.

En segundo lugar, se desarrolló de manera práctica el tutorial oficial de Django (*Writing your first Django app*, partes 1 a 8), disponible en el sitio web oficial del framework. Este tutorial guía paso a paso la creación de una aplicación funcional (una encuesta web) y permite aplicar los conceptos abordados teóricamente. La ejecución del tutorial se realizó localmente, utilizando el entorno de desarrollo de Python y los comandos sugeridos por la documentación.

**Resultados**

El trabajo permitió alcanzar un entendimiento sólido de los fundamentos del framework Django. Las respuestas teóricas abarcaron con claridad los principales componentes del patrón MTV, el funcionamiento del ORM, la utilidad de las *apps* y la estructura del flujo *request-response*. Asimismo, se explicó el uso de los *templates* como herramienta de presentación, así como el procedimiento completo para instalar Django correctamente en un entorno local.

En cuanto al trabajo práctico, se logró implementar sin dificultades la aplicación propuesta en el tutorial oficial, comprendiendo su estructura modular, la creación de modelos y vistas, la gestión del panel administrativo, la configuración de rutas y la incorporación de plantillas HTML. La funcionalidad final incluía la creación de preguntas con opciones, la visualización y votación de encuestas, y la representación de resultados.

**Discusión**

Los resultados obtenidos confirman la eficacia de Django como framework para el desarrollo web. Desde un enfoque pedagógico, la combinación de teoría guiada por preguntas clave junto con la práctica del tutorial oficial se mostró altamente efectiva para afianzar conceptos y familiarizarse con el entorno de trabajo. El enfoque modular y la arquitectura MTV permiten una organización clara del código, facilitando su mantenimiento y escalabilidad.

Una observación relevante fue la simplicidad del proceso de instalación y despliegue inicial, lo que convierte a Django en una opción accesible para principiantes, sin por ello sacrificar su potencial profesional. La documentación oficial, clara y actualizada, también fue un factor determinante en el éxito del proceso de aprendizaje.

Entre las posibles limitaciones del estudio se encuentra la falta de comparación directa con otros frameworks (como Flask o Laravel), lo cual podría ser abordado en trabajos futuros. No obstante, dentro del alcance establecido, se logró cumplir con los objetivos propuestos y se sientan las bases para proyectos más complejos utilizando Django en contextos reales de desarrollo.

**Referencias**

Django Software Foundation. (2024). *The web framework for perfectionists with deadlines*. <https://www.djangoproject.com/>

Python Software Foundation. (2024). *Welcome to Python.org*. <https://www.python.org/>

Mozilla Developer Network. (2023). *Django web framework (Python)*. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django>

Real Python. (2024). *Python web frameworks: Flask vs Django*. <https://realpython.com/tutorials/django/>

GeeksforGeeks. (2023). *Django – Introduction*. <https://www.geeksforgeeks.org/django-introduction/>

**Índice**

1. [¿Qué es Django y por qué lo usaríamos?](#_heading=h.92zrytt5kis9)
2. [¿Qué es el patrón MTV (*Model-Template-View*) en Django? (simplificado de MVC). Compará MTV con MVC.](#_heading=h.q1m8sfszmn7i)
3. [¿Qué entendemos por *app* en Django?](#_heading=h.kkv6sh255i5s)
4. [¿Qué es el flujo *request-response* en Django?](#_heading=h.j77r6mxr0e0s)

1. [¿Qué es el concepto de ORM (*Object-Relational Mapping*)?](#_heading=h.pid4l2xkfe4r)

1. [¿Qué son los *templates* en Django?](#_heading=h.7rl76gh79csi)

1. [¿Cómo se lo instala?](#_heading=h.tgmxcrqra2nr)



1. ¿Qué Es Django Y Por Qué Lo Usaríamos?

Django es un *framework* de desarrollo web de alto nivel que facilita la creación de aplicaciones web mediante el lenguaje de programación Python. Se destaca por ofrecer una estructura predefinida que permite desarrollar proyectos de manera rápida, segura y organizada. Su filosofía de desarrollo se basa en el principio de “no te repitas” (*Don't Repeat Yourself*), promoviendo la reutilización de código y la eficiencia.

Una de las razones principales para utilizar Django es su amplio conjunto de herramientas integradas. Estas incluyen sistemas de autenticación de usuarios, panel de administración, mapeo objeto-relacional (*ORM*), manejo de formularios, sistema de plantillas y enrutamiento de URLs, entre otros. Gracias a estas funcionalidades, los desarrolladores pueden enfocarse en las particularidades de su aplicación sin tener que construir todas las funcionalidades básicas desde cero.

Django también enfatiza la seguridad. Incorpora mecanismos para prevenir ataques comunes como inyección *SQL*, falsificación de solicitudes entre sitios (*CSRF*) y secuestro de sesiones, ofreciendo un entorno más protegido por defecto. Además, sigue un enfoque estructurado, conocido como el patrón *MTV* (*Model-Template-View*), que facilita la separación de responsabilidades y el mantenimiento del código.

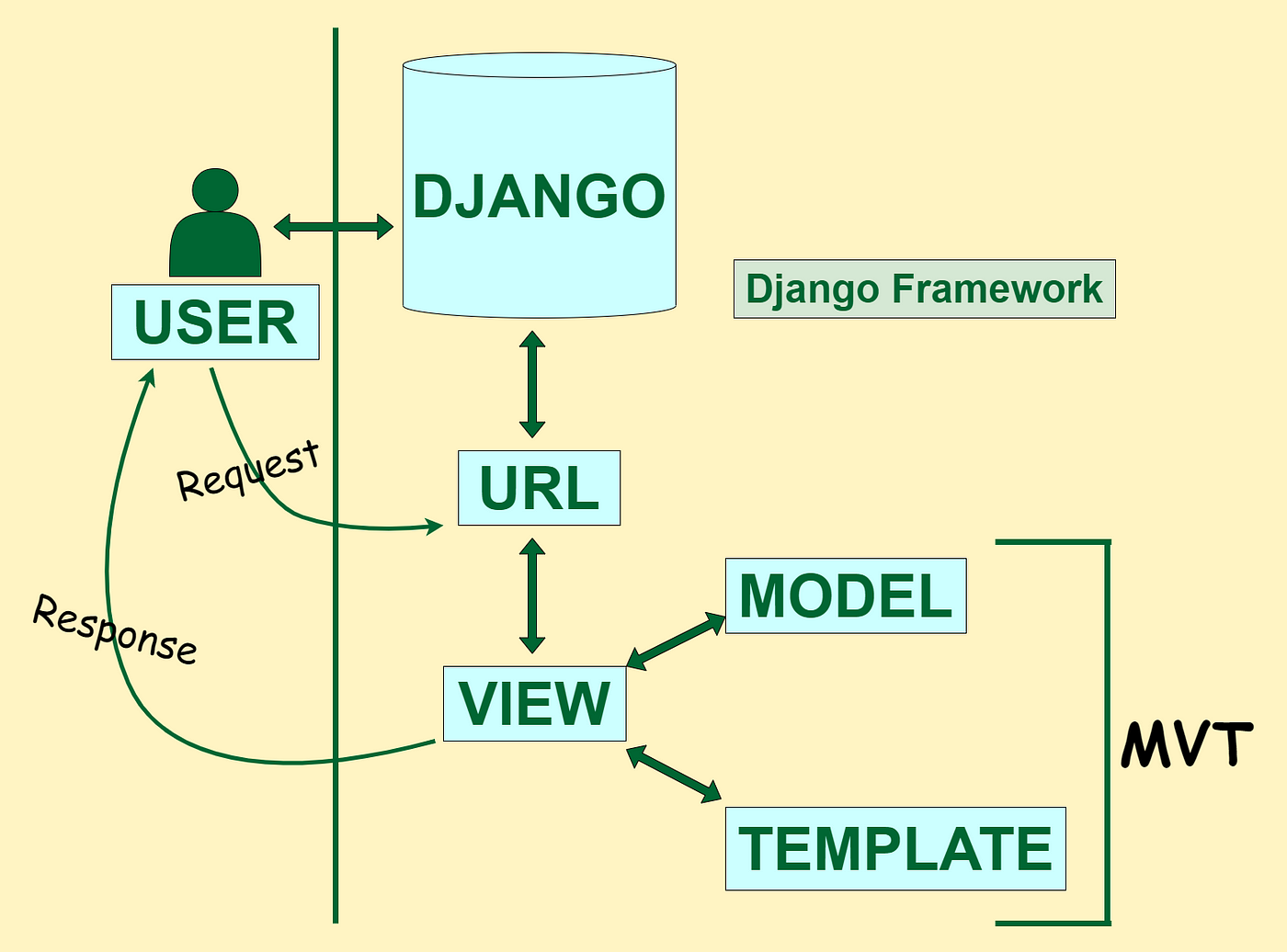
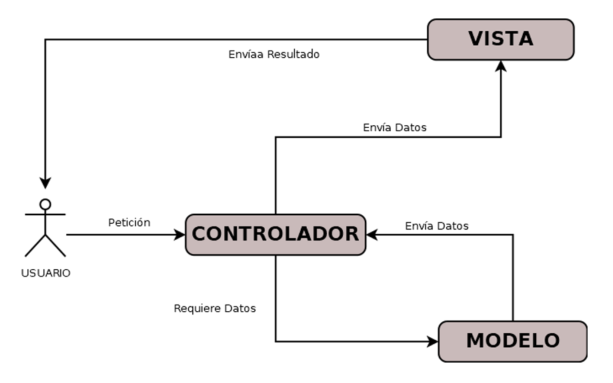
En síntesis, Django es una excelente elección para el desarrollo web debido a su rapidez de implementación, robustez, seguridad incorporada y su comunidad activa. Estas características lo convierten en una herramienta ideal tanto para principiantes como para desarrolladores con experiencia que desean construir sitios o aplicaciones web de manera profesional.

1. ¿Qué Es El Patrón MTV (Model-Template-View) En Django? Comparación Con MVC

El patrón *MTV* es un enfoque utilizado por Django para estructurar las partes principales de una aplicación web, facilitando su desarrollo y mantenimiento al separar las responsabilidades de cada componente. Este patrón representa una adaptación del clásico *MVC* (*Model-View-Controller*), con diferencias tanto en terminología como en la distribución de funciones internas.

En Django, el **Modelo** (*Model*) se encarga de gestionar todo lo relacionado con la base de datos: cómo se estructuran los datos, cómo se almacenan y cómo se recuperan. La **Vista** (*View*) contiene la lógica de negocio, se comunica con los modelos y decide qué datos enviar a la plantilla. El **Template** representa la capa de presentación, es decir, la interfaz visual que verá el usuario.

En contraste, en el patrón *MVC* tradicional, el **Modelo** también gestiona los datos, pero la **Vista** representa la interfaz del usuario, y el **Controlador** maneja la lógica de negocio y la interacción entre el modelo y la vista. En Django, esta lógica de control se encuentra dentro de las *Views*, y el Template adopta el rol visual de la vista del *MVC*.

Por lo tanto, aunque Django renombre y reorganice algunos componentes, mantiene la esencia del diseño orientado a la separación de responsabilidades. Esta adaptación permite un flujo de desarrollo más intuitivo dentro del marco del lenguaje Python, promoviendo claridad, modularidad y escalabilidad en los proyectos web.

1. ¿Qué Entendemos Por App En Django?

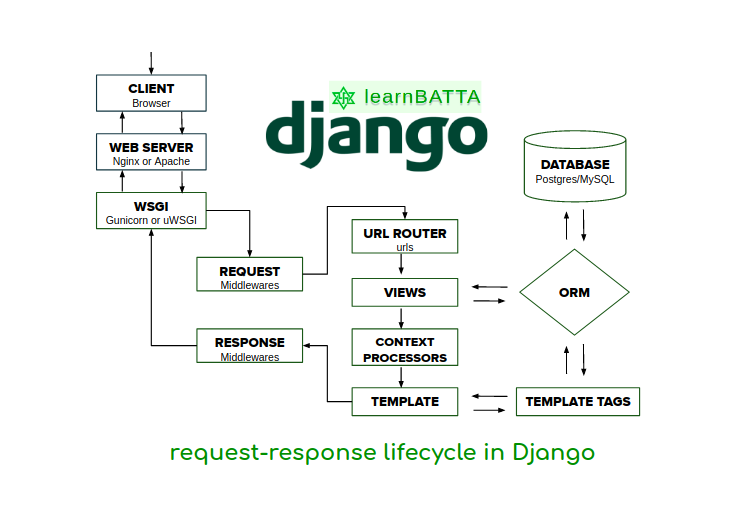
En Django, una *app* es una unidad modular de funcionalidad dentro de un proyecto web. Cada *app* cumple una función específica, como gestionar usuarios, comentarios, productos o publicaciones. En términos simples, puede pensarse como un bloque independiente que resuelve un aspecto concreto de la aplicación web. Por ejemplo, en un sitio dedicado al cine, una *app* podría encargarse exclusivamente de manejar las películas, otra de los usuarios registrados y otra de las valoraciones o reseñas.

Cada *app* posee su propia estructura de archivos y componentes, incluyendo modelos, vistas, plantillas, rutas y archivos de configuración. Esta separación favorece la organización del proyecto, especialmente en entornos colaborativos o a medida que el proyecto crece en complejidad. Además, las *apps* son reutilizables, lo que significa que pueden integrarse en otros proyectos con mínimos ajustes, lo cual promueve la eficiencia y el aprovechamiento del código existente.

Es importante destacar que, en el contexto de Django, el término *app* no hace referencia a aplicaciones móviles, sino a componentes funcionales internos de un proyecto web. Un solo proyecto puede contener múltiples *apps* que trabajan de forma integrada para ofrecer al usuario final una experiencia coherente y fluida.

1. ¿Qué Es El Flujo Request-Response En Django?

El flujo *request-response* en Django describe el mecanismo mediante el cual se establece la comunicación entre el usuario y el servidor web. Cada vez que una persona interactúa con una página web, como al abrir una URL o enviar un formulario, se genera una solicitud (*request*) que es enviada al servidor. Este ciclo es fundamental para comprender cómo Django maneja las interacciones en tiempo real con los usuarios.



Cuando el servidor recibe una solicitud, Django examina la URL para determinar qué vista (*view*) debe encargarse de procesarla. La vista actúa como intermediaria entre el usuario y los datos, ejecutando la lógica necesaria: puede consultar una base de datos, validar formularios, aplicar reglas de negocio o preparar la información que se va a mostrar. Luego, construye una respuesta (*response*), que normalmente es una página HTML renderizada, y la envía de vuelta al navegador del usuario.

Este ciclo ocurre en fracciones de segundo, pero implica múltiples pasos internos. Django abstrae la complejidad de este proceso mediante su sistema de rutas, vistas y plantillas, lo que permite a los desarrolladores centrarse en la funcionalidad sin preocuparse por los detalles bajos del manejo de solicitudes. Este flujo bien estructurado garantiza un desarrollo eficiente y una experiencia de usuario consistente.

1. ¿Qué Es El Concepto De ORM (Object-Relational Mapping)?

El concepto de *Object-Relational Mapping* (*ORM*) hace referencia a una técnica que permite interactuar con bases de datos utilizando código orientado a objetos, en lugar de escribir directamente sentencias en lenguaje estructurado de consultas (*SQL*). En el contexto de Django, el *ORM* permite definir estructuras de datos mediante clases de Python, las cuales se traducen automáticamente en tablas y operaciones sobre bases de datos.

Por ejemplo, para representar una tabla de usuarios, se crea una clase en Python con los atributos correspondientes. A través del *ORM*, se pueden realizar operaciones como búsquedas, inserciones o actualizaciones usando código Python convencional. Django se encarga de transformar estas instrucciones en sentencias *SQL* compatibles con el motor de base de datos utilizado.

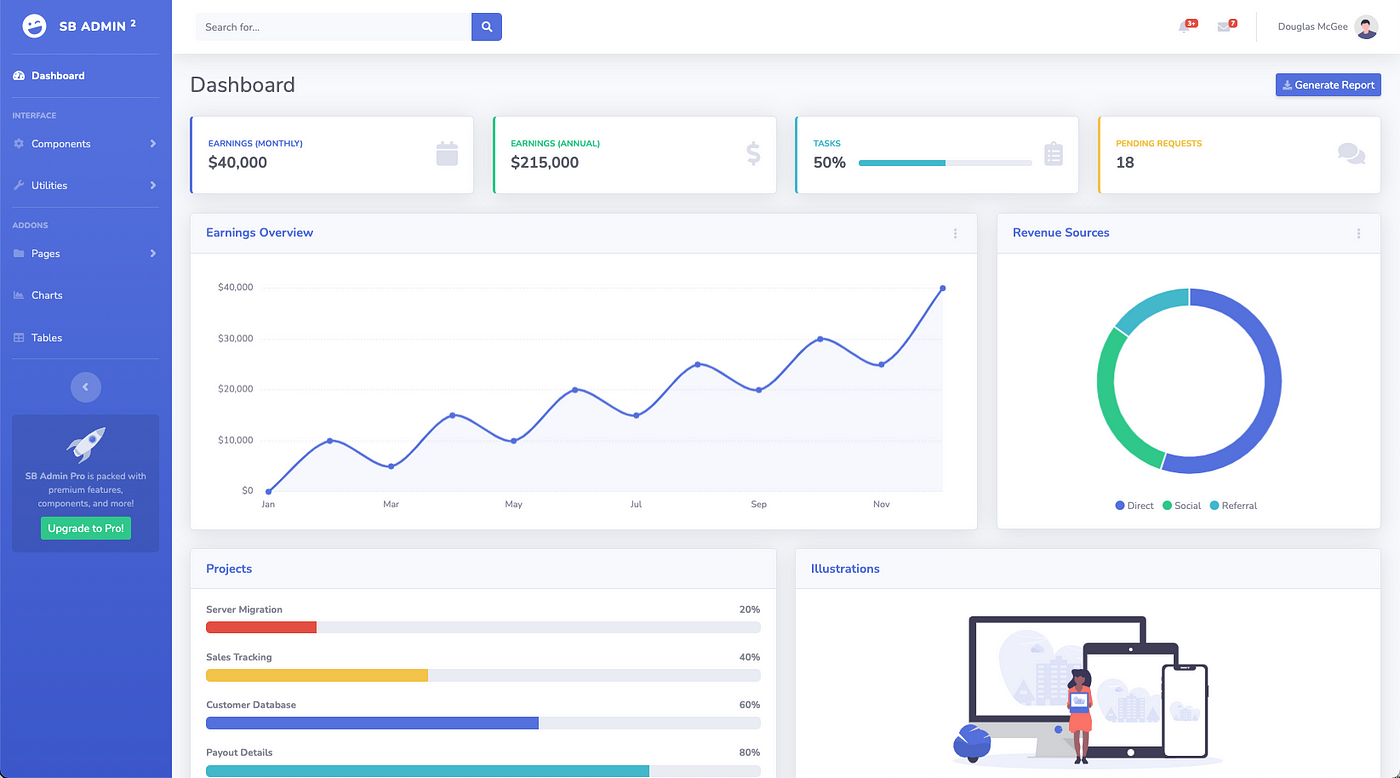
Este enfoque presenta múltiples ventajas. En primer lugar, reduce la necesidad de escribir código en distintos lenguajes, favoreciendo un desarrollo más limpio y coherente. También contribuye a la seguridad, ya que evita errores comunes asociados a consultas mal construidas. Además, el uso del *ORM* permite una mayor portabilidad: si se decide cambiar de un sistema de base de datos a otro (por ejemplo, de SQLite a PostgreSQL), el código en Python permanece prácticamente igual, ya que el *ORM* se adapta al nuevo motor de forma transparente.

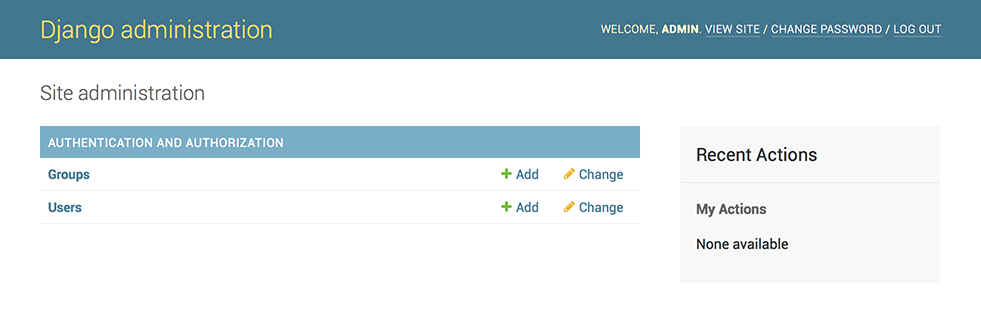
1. ¿Qué Son Los Templates En Django?

Los *templates* en Django son archivos que definen la presentación visual de una aplicación web. Generalmente escritos en HTML, pueden incluir instrucciones especiales que permiten insertar datos dinámicamente, condicionar la visualización de elementos o repetir estructuras. Estos archivos cumplen un rol fundamental en la separación entre la lógica del programa y la interfaz del usuario.

Un ejemplo común de uso de *templates* es la visualización de una lista de productos. En lugar de escribir manualmente cada producto en el archivo HTML, la vista de Django proporciona los datos al *template*, que se encarga de recorrerlos y presentarlos utilizando un lenguaje de plantillas simplificado. Esto permite una mayor flexibilidad y escalabilidad en el diseño de interfaces.

La utilización de *templates* favorece una arquitectura limpia, ya que separa la lógica del negocio (ubicada en vistas y modelos) de la presentación visual. Este enfoque facilita el mantenimiento del proyecto, especialmente en equipos de desarrollo, ya que diseñadores y programadores pueden trabajar en paralelo sobre componentes claramente definidos.





1. ¿Cómo Se Lo Instala?

Para comenzar a utilizar Django, es necesario realizar una instalación previa de algunos componentes básicos. En primer lugar, debe asegurarse de tener Python instalado, ya que Django está completamente desarrollado en este lenguaje. Python puede descargarse de forma gratuita desde su sitio web oficial ([*https://www.python.org*](https://www.python.org/)), y su instalación es sencilla en la mayoría de los sistemas operativos.

Una vez instalado Python, la forma más directa de instalar Django es mediante el gestor de paquetes *pip*, que suele venir incluido con Python. El comando a ejecutar en la terminal o consola es el siguiente:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Este comando descarga e instala automáticamente la versión estable más reciente de Django. No se requieren configuraciones adicionales ni la instalación de bases de datos externas, ya que Python incluye SQLite por defecto, una base de datos ligera que resulta adecuada para ambientes de desarrollo y pruebas.

Para verificar que Django se haya instalado correctamente, puede abrir la consola interactiva de Python ejecutando python, y luego ingresar las siguientes instrucciones:

Texto

El contenido generado por IA puede ser incorrecto.

Si se muestra un número de versión, como por ejemplo 5.2, significa que la instalación fue exitosa y el entorno está listo para comenzar.

A partir de este punto, es posible crear un nuevo proyecto con Django y seguir el flujo de desarrollo propuesto en su documentación oficial o en el presente trabajo práctico. La simplicidad del proceso de instalación es una de las características que hacen de Django una herramienta accesible para quienes comienzan en el desarrollo web.