Trabajo práctico N°3 - Laboratorio de Algoritmos y Estructuras de Datos

Preentrega Django

Santino Luciano Rodriguez Fuchs

Instituto Industrial Luis A. Huergo

4° Año Computadoras

Ignacio García

10 de Julio de 2025

1- ¿Qué es django y por qué lo usaríamos?

Django es un *framework*[1] de desarrollo web de alto nivel creado con Python, pensado para facilitar la creación de sitios web completos, seguros y escalables. Su propósito es reducir al mínimo el tiempo y el esfuerzo necesario para desarrollar una aplicación web desde cero, ofreciendo herramientas listas para usar que resuelven tareas comunes del desarrollo. Gracias a esto, Django permite que un desarrollador pase de la idea al sitio funcional en apenas unas horas, sin sacrificar calidad ni seguridad.

Una de las mayores ventajas de Django es que automatiza muchos de los procesos del desarrollo backend, lo cual lo convierte en una excelente opción incluso para quienes solo manejan conocimientos básicos de HTML, CSS y JavaScript. No es necesario construir a mano formularios, sistemas de usuarios, paneles de administración, enlaces entre páginas o conexiones con bases de datos, Django ya incluye módulos que hacen todo eso de forma automática y coherente. Esto agiliza enormemente el desarrollo, al punto que, en palabras del equipo oficial, Django está diseñado para que "puedas concentrarte en escribir tu aplicación sin tener que reinventar la rueda".

Además, se trata de un software gratuito y de código abierto, lo que significa que cualquier persona puede usarlo, adaptarlo o colaborar en su mejora sin restricciones. Su comunidad es amplia y activa, lo que facilita encontrar documentación, foros de ayuda y ejemplos prácticos de todo tipo.

Otro de los pilares fundamentales de Django es su enfoque en la seguridad. Desde el inicio, ofrece protección contra errores comunes que pueden comprometer un sitio web, como la inyección de SQL, los scripts maliciosos entre sitios, el clickjacking o la falsificación de solicitudes (CSRF). También incluye un sistema de autenticación robusto que permite gestionar usuarios, sesiones y contraseñas de manera segura.

En cuanto al diseño visual, Django utiliza un sistema de *templates*[2] que permite generar páginas HTML dinámicas sin mezclar la lógica de programación con el contenido visual. Esto hace que sea muy sencillo integrar hojas de estilo (CSS) y scripts (JavaScript) sin necesidad de conocimientos avanzados, el desarrollador puede concentrarse en la estructura general del sitio y dejar que Django se encargue de unir todo detrás de escena.

Finalmente, Django se destaca por su capacidad de escalar. Grandes empresas, organizaciones gubernamentales y plataformas científicas lo han usado con éxito para proyectos que manejan miles o millones de usuarios. Su arquitectura flexible lo hace ideal tanto para sitios simples como para sistemas complejos que necesitan soportar mucho tráfico y datos.

En definitiva, Django es un framework robusto, accesible y bien diseñado, que permite construir aplicaciones web de manera rápida, segura y profesional. Para quienes ya dominan Python y tienen una base en tecnologías web, representa una herramienta ideal para llevar proyectos del papel a internet con

[1] Estructura de desarrollo reutilizable que organiza el código mediante componentes, bibliotecas y patrones de diseño predefinidos, con el fin de facilitar la creación, mantenimiento y escalabilidad de aplicaciones.

[2] Archivo o estructura que define el diseño y la presentación de una interfaz, permitiendo separar el contenido dinámico de la lógica de programación mediante marcadores o sintaxis específica para ser completados por el motor de plantillas.

velocidad y confianza. Más detalles sobre sus características pueden encontrarse en su sitio oficial, donde se explica su filosofía, ventajas y casos de uso reales.

2- ¿Qué es el patrón MTV (Model-Template-View) en django? (simplificado de MVC). Compará MTV con MVC.

Django se basa en un enfoque estructurado para organizar el código de una aplicación web, utilizando un patrón de diseño llamado MTV, que significa Model-Template-View. Este patrón es una adaptación propia del ya conocido MVC, sigla de Model-View-Controller, ampliamente utilizado en el desarrollo de software para separar la lógica de una aplicación en tres partes principales: los datos, la interfaz y el control. Comprender cómo funciona MVC permite entender mejor la propuesta de Django y por qué su modelo MTV representa una versión simplificada y especializada para el desarrollo web.

En el patrón clásico MVC, el Modelo representa la estructura de los datos y la lógica que permite acceder a ellos. Se encarga de definir cómo se guardan, se consultan y se relacionan los datos en una aplicación. La Vista, por otro lado, es el componente que muestra esos datos al usuario, es decir, la interfaz visible, habitualmente construida en HTML y otros lenguajes de presentación. Por último, el Controlador es el encargado de manejar la lógica entre ambas partes: recibe las acciones del usuario (como clics o envíos de formularios), decide cómo procesarlas, accede o modifica datos a través del modelo, y luego determina qué vista debe ser mostrada como respuesta.

Este enfoque permite una separación clara de responsabilidades. Gracias a ello, un desarrollador puede modificar la interfaz visual sin tocar la lógica interna, o cambiar cómo se almacenan los datos sin necesidad de alterar las vistas. Esta división también facilita el trabajo en equipo, permitiendo que distintas personas trabajen sobre distintas capas sin interferirse.

Ahora bien, Django no utiliza exactamente este patrón. Aunque comparte la misma idea de dividir responsabilidades, redefine los nombres y funciones de los componentes para adaptarlos mejor al entorno web y a su arquitectura interna. Así es como surge el patrón MTV, una interpretación específica que mantiene la esencia de MVC pero redistribuye los roles de forma distinta.

En Django, el Modelo sigue cumpliendo el mismo propósito que en MVC: representa la estructura de datos y gestiona el acceso a la base de datos mediante un sistema ORM que simplifica las consultas y relaciones. La Vista, sin embargo, no es la parte visual como en MVC, sino una función de Python que recibe la petición del usuario, interactúa con los modelos para obtener o modificar información, y luego entrega una respuesta. Finalmente, el Template cumple el rol de la vista tradicional, ya que es el archivo HTML que se le presenta al usuario, con datos dinámicos incluidos gracias a la plantilla.

En la práctica, el flujo de funcionamiento del patrón MTV es el siguiente:

1. El usuario hace una petición al servidor, por ejemplo, accediendo a una URL.
2. Django dirige esa petición a una vista, que es una función (o clase) definida en Python.
3. La vista consulta o modifica datos a través del modelo si es necesario.
4. Luego, la vista selecciona una plantilla y le pasa los datos que quiere mostrar.

5. El sistema renderiza esa plantilla con la información dinámica y se genera una página HTML como respuesta al usuario.

Este sistema es más simple que MVC porque Django actúa internamente como el controlador, eliminando la necesidad de que el desarrollador lo implemente directamente. Según se explica en GeeksforGeeks, el patrón MVT "elimina la necesidad de un controlador explícito, ya que el marco de trabajo Django maneja esa parte por medio de su sistema de vistas". Es decir, lo que en otros entornos se define como "controlador", Django lo resuelve con su propio motor de enrutamiento y sus vistas, lo que simplifica mucho el código sin perder control ni flexibilidad.

3- ¿Qué entendemos por app en django?

En Django, cuando comenzamos a trabajar en un nuevo desarrollo, lo primero que creamos es lo que se conoce como un proyecto. Un proyecto es la estructura base que representa toda la aplicación web en su conjunto. Sin embargo, lo interesante de Django es que este proyecto está pensado para componerse de múltiples aplicaciones independientes, llamadas comúnmente "apps". Este diseño modular es una de las fortalezas más claras del framework, ya que permite dividir grandes desarrollos en componentes más pequeños, reutilizables y fáciles de mantener.

Una app en Django es una unidad funcional del proyecto. Es decir, representa una parte concreta del sitio web, como un sistema de comentarios, un blog, una galería de imágenes, una tienda, un foro o cualquier otra funcionalidad específica. Cada app tiene su propia lógica, modelos, vistas, URLs y plantillas, y puede desarrollarse de forma aislada, incluso con la posibilidad de ser reutilizada en otros proyectos diferentes. Django está especialmente pensado para este tipo de organización, porque fomenta el principio de separación de responsabilidades y promueve el desarrollo escalable.

La relación entre proyecto y apps es similar a la de un contenedor y sus componentes. El proyecto agrupa y coordina el funcionamiento de las distintas apps, mientras que cada app se ocupa de una tarea concreta. Por ejemplo, si estuviéramos desarrollando una red social, podríamos tener una app para los perfiles de usuario, otra para el sistema de mensajes, otra para publicaciones y otra para notificaciones. Cada una de esas partes estaría encapsulada en su propia app, sin interferir directamente con las demás, lo que permite trabajar de forma ordenada y hacer cambios en una sin afectar a las otras.

Cuando ejecutamos el comando django-admin startproject, estamos creando el proyecto principal, que incluye archivos como settings.py para la configuración general, urls.py para las rutas principales, y wsgi.py o asgi.py para la comunicación con el servidor. Luego, mediante el comando python manage.py startapp, creamos una nueva app dentro del proyecto. Esa app tendrá su propio archivo models.py para definir los datos, views.py para responder a las peticiones del usuario, urls.py para definir sus rutas internas, y un directorio de plantillas para mostrar el contenido al usuario.

En la documentación oficial de Django, se explica que una app es "un conjunto de archivos de configuración y código de Python que trabaja conjuntamente para proporcionar alguna funcionalidad". Y lo más importante es que las apps están diseñadas para ser plug-and-play, es decir, se pueden activar, desactivar o reutilizar fácilmente según se necesite. Django permite registrar estas apps dentro del archivo INSTALLED_APPS en la configuración del proyecto, y a partir de ahí, las integra automáticamente en el sistema.

Esta estructura modular tiene muchas ventajas. Por un lado, permite que equipos de desarrollo puedan trabajar en diferentes partes de un mismo proyecto sin pisarse el código. Por otro lado, permite escalar fácilmente: si una app deja de ser necesaria o debe reemplazarse, puede eliminarse o sustituirse sin afectar la totalidad del sistema. Además, si una app está bien diseñada, puede empaquetarse y usarse en otros proyectos, algo que muchas veces ocurre en bibliotecas de terceros.

En resumen, una app en Django es una parte independiente y funcional de un sitio web que vive dentro de un proyecto mayor. Mientras que el proyecto representa la totalidad de la aplicación y coordina su comportamiento, las apps representan componentes específicos que se pueden desarrollar, modificar o incluso reutilizar por separado. Este enfoque hace que Django sea un framework potente, organizado y muy flexible, ideal tanto para proyectos simples como para aplicaciones web complejas y de gran escala.