

**TRABAJO PRÁCTICO N°3 - DJANGO**

Marcyniuk Teo

Instituto Industrial Luis A. Huergo

Departamento de Computación - Laboratorio de Algoritmos y Estructuras de Datos

Profesor Ignacio Miguel Garcia

1. **¿Qué es Django y por qué lo usaríamos?**

Django es un *framework* de desarrollo web de código abierto, escrito en Python que gestiona tareas comunes de desarrollo web, como autenticación de usuarios, la administración de contenido, los mapas del sitio, las fuentes RSS y ayuda a los desarrolladores a evitar errores comunes, como la inyección SQL, los scripts entre sitios, la falsificación de solicitudes entre sitios y el *clickjacking*. Este es utilizado desde sistemas de gestión de contenido, hasta redes sociales y plataformas de computación científica, con el objetivo de satisfacer las demandas de tráfico más intensas.

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

El modelo Modelo-vista-controlador, un patrón de diseño de software cuyo objetivo es decidir qué información mostrar y cómo mostrarla, dividiendo una aplicación en tres partes interconectadas: Modelo, Vista y Controlador.

El Modelo se encarga de representar la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, teniendo en cuenta los privilegios de acceso que se hayan especificado.

Por otro lado, la Vista presenta el Modelo en un formato adecuado para interactuar y accede al modelo mediante el Controlador, que funciona como intermediario entre la 'vista' y el 'modelo', responde a eventos causados por el usuario e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información.

Django utiliza la versión simplificada de este modelo: el modelo MTV o *Model-Template-View*, que no se desvía demasiado del patrón MVC, simplemente lo implementa de una manera distinta. En este modelo, la vista es un poco diferente; esta asume el rol del controlador en Django ta que contiene la lógica que determina qué datos se muestran y qué plantillas usar. Procesa la solicitud y prepara la respuesta. También existe una Plantilla, la capa de presentación donde se definen el HTML y el diseño que se utiliza para representar los datos de la vista en un formato legible.  
Sin embargo, el controlador sigue estando presente en el MTV, nada más que de una manera interna, ya que el framework mismo funciona de controlador.

1. **¿Qué entendemos por *app* en Django?**

Entendemos por app en Django a una unidad lógica de código con una función específica, como un sistema de blog o un sistema de gestión de usuarios, que se encuentra dentro de un proyecto más grande. Es como un componente modular y reutilizable, ya que se puede reutilizar en diferentes proyectos, que son conjuntos de dichas aplicaciones junto a configuraciones para un sitio web específico.

1. **¿Qué es el flujo *request-response* en django?**

En Django, el flujo *request-response* es el mecanismo fundamental que gestiona cómo los usuarios interactúan con una aplicación web. Cuando un cliente solicita una página, Django recibe esa solicitud HTTP y crea un objeto llamado *HttpRequest*, que se pasa a la función *view* encargada de procesarla. La vista genera una respuesta adecuada en forma de un objeto *HttpResponse*, que luego se devuelve al cliente.

Para que este intercambio funcione, Django no se comunica directamente con el navegador, sino que necesita una conexión con un servidor web, un sistema que se encarga de recibir las solicitudes desde Internet y responder con los archivos o datos correspondientes. Para esto, utiliza el WSGI o *Web Server Gateway Interface*, que actúa como puente entre el servidor web y la aplicación Django. Cuando llega una solicitud, el servidor web la entrega a través del WSGI, que ejecuta la aplicación y recibe de vuelta la respuesta generada para enviarla al cliente.

Durante este trayecto, Django utiliza una serie de componentes llamados *middlewares*, que operan sobre cada solicitud y respuesta que pasa por el sistema. Son clases que pueden modificar, interrumpir o redirigir el flujo de datos. Se usan, por ejemplo, para gestionar sesiones, verificar la autenticación de los usuarios o prevenir ataques como CSRF. Django permite definir *middlewares* personalizados o usar los que ya vienen integrados, como *Authentication Middleware*.

1. **¿Qué es el concepto de ORM (*Object-Relational Mapping*)?**

El *Object-Relational Mapping*, es una técnica de programación para interactuar con bases de datos SQL sin tener que escribir directamente consultas SQL, utilizando objetos de *Python* que representan tablas de bases de datos, lo que facilita las operaciones de creación, lectura, actualización y eliminación.

Esta herramienta es fundamental debido a que muchos productos populares de base de datos solamente pueden almacenar y manipular valores escalares como enteros y cadenas, organizados en tablas normalizadas. Esta herramienta se encarga de convertir los valores de los objetos en grupos de valores simples para almacenarlos en la base de datos.

1. **¿Qué son los *templates* en Django?**

En un proyecto Django, las plantillas cumplen el rol de ser el vínculo entre la lógica de la aplicación y la interfaz que ve el usuario. Estas plantillas son archivos HTML que contienen tanto contenido estático como dinámico, permitiendo definir la estructura visual de una aplicación web mientras se integran elementos generados por el *backend*. Para lograr esto, Django utiliza su propio sistema de plantillas, conocido como *Django Template Language*, que permite insertar contenido dinámico mediante variables y etiquetas específicas dentro del HTML. Estas plantillas se representan con un contexto. La representación reemplaza las variables con sus valores, que se buscan en el contexto, y ejecuta las etiquetas. Todo lo demás se muestra tal cual.

Un proyecto de Django puede configurarse con uno o varios sistemas de plantillas, o incluso no usar ninguno si no se requiere. Además de su propio sistema, Django también ofrece soporte para otros motores como Jinja2, e incluso permite implementar backends personalizados. La API estándar que Django proporciona para trabajar con plantillas permite cargar y renderizar documentos de forma independiente del motor que se esté utilizando, lo cual brinda flexibilidad sin sacrificar la cohesión del sistema.

**¿Cómo se lo instala?**

Django se puede instalar de forma sencilla utilizando pip, el sistema de gestión de paquetes de Python. Es recomendable trabajar dentro de un entorno virtual para evitar posibles conflictos entre dependencias de distintos proyectos. Para comenzar, se debe generar un entorno virtual con el comando python -m venv env. , y luego activarlo. En Windows, esto se hace con env\Scripts\activate, mientras que en Linux o macOS se utiliza source env/bin/activate. Una vez activado el entorno, se instala Django con pip install django. Finalmente, para verificar que la instalación fue exitosa, se puede ejecutar django-admin --version. Esto deja el entorno listo para comenzar un nuevo proyecto con el comando django-admin startproject.

**Referencias:**

Django Software Foundation. *(2025)*

<https://www.djangoproject.com/>

<https://docs.djangoproject.com/en/5.2/intro/tutorial01/>

<https://docs.djangoproject.com/es/5.2/topics/templates/>

Wikipedia *(2025)*

<https://es.wikipedia.org/wiki/Django_(framework)>

<https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador>

<https://es.wikipedia.org/wiki/Mapeo_relacional_de_objetos>

Quora Assistant - *What is the difference between an MVC and an MVT framework? (2024)*

<https://www.quora.com/What-is-the-difference-between-an-MVC-and-an-MVT-framework>

Steps (2022)

<https://blog.stepskochi.com/mvc-vs-django-mvt-design-patterns/>

Reddit - *What is an APP in django? (2023)*

<https://www.reddit.com/r/django/comments/154npht/what_is_an_app_in_django/>

Geeksforgeeks (2025)

<https://www.geeksforgeeks.org/python/how-to-create-an-app-in-django/>

Medium (2023)

<https://medium.com/@developerstacks/django-request-response-cycle-7165167f54c5>

<https://emmanueldav.medium.com/django-templates-mastering-the-basics-29a99813af9f>

Moldstud (2025)

<https://moldstud.com/articles/p-exploring-the-django-request-response-cycle-an-in-depth-look-for-developers>

Urian Viera - *Guía Completa sobre ORM en Django (S.F.)*

<https://urianviera.com/django/guia-completa-sobre-orm-en-django>

**Glosario**

**Framework:** Conjunto de herramientas y componentes predefinidos que se utilizan para desarrollar aplicaciones o software de manera más eficiente y organizada al proporcionar una base sólida que funciona como una plantilla que define la estructura y las reglas básicas para un tipo específico de proyecto, evitando la necesidad de escribir código desde cero para tareas comunes.

**Fuentes RSS:** Formato web que permite a los usuarios suscribirse a actualizaciones de sitios web y blogs, recibiendo resúmenes o el contenido completo de las publicaciones directamente en un lector de feeds o agregador.

**Clickjacking:** Ataque cibernético donde un atacante engaña a un usuario para que haga click en algo diferente a lo que parece.

**Ataque CSRF:** Tipo de ataque cibernético en el que un atacante engaña a un usuario para que realice acciones no deseadas en una aplicación web en la que el usuario ya está autenticado.

**Backend:** Parte del sitio o aplicación que no es visible para el usuario, encargada de la lógica y el procesamiento de datos en segundo plano