****

**Materia:** Laboratorio de Algoritmos.

**Profesor:** Ignacio García.

**Curso:** 4 AO.

**Alumnos:** Joaquín Piñeiro.

**2025**

Tabla de Contenidos

[1.](#_heading=h.fhbsuvs165ka) ¿Qué es Django y por qué lo usaríamos? 2

[2.](#_heading=h.xdgtgc2dv8r5) ¿Qué es el patrón MTV (Model-Template-View) en django? (simplificado de MVC). Compará MTV con MVC. 2

[3.](#_heading=h.iuoa7k5je20a) ¿Qué entendemos por app en Django? 3

[4.](#_heading=h.mz7jz9ijt822) ¿Qué es el flujo request-response en django? 3

[5.](#_heading=h.f7n5dfqp6xaz) ¿Qué es el concepto de ORM (Object-Relational Mapping)? 4

[6.](#_heading=h.uszwdwrm2nu3) ¿Qué son los templates en django? 4

[7.](#_heading=h.mo08mggwe29b) Leé con detenimiento este django at a glance: 4

[8.](#_heading=h.6smr60q02elh) ¿Cómo se lo instala? 5

[9.](#_heading=h.gqptllph29ic) Glosario 5

[10. Fuente 6](#_heading=h.kt5j8ymp0npl)

# ¿Qué es Django y por qué lo usaríamos?

Django es un framework de desarrollo web de código abierto, escrito en el lenguaje de programación **Python**. Se destaca por su rapidez en el desarrollo, ya que permite crear aplicaciones en menos tiempo ya que incluye funcionalidades listas para su uso (gestión de usuarios, formularios de autenticación, panel de administración automático, etc.), además incorpora medidas de seguridad de desarrollo por defecto que ayudan a proteger las aplicaciones ante los ataques más comunes como por ejemplo inyecciones SQL, Cross-site Scripting, Falsificación de solicitudes (CSRF), en cuanto a su diseño, soporta aplicaciones que crecen en cantidad de usuarios y datos, diferentes redes sociales como Instagram utilizan Django, tiene una comunidad activa y la documentación oficial es muy completa, podemos agregar que existen muchas extensiones (paquetes) que nos ayudan a ampliar la funcionalidad.

Usaríamos Django por lo mencionado anteriormente, permite crear aplicaciones web rápido, seguras y bien organizadas, cuenta con una estructura clara.

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

El patrón **MTV** en Django organiza la aplicación en tres componentes. El **Model** define la estructura y el manejo de los datos, por ejemplo, cómo se guardan en la base de datos. El **Template** es la parte visual, es decir, el HTML que verá el usuario en su navegador. La **View** contiene la lógica que decide qué información recuperar y qué plantilla mostrar. Comparado con MVC, el Model es igual, pero la View en MVC se llama Template en Django, y el Controller de MVC es lo que Django llama View. Hacen lo mismo, solo cambian los nombres.

1. ¿Qué entendemos por app en Django?

Una app, es una parte de nuestro proyecto que hace algo en particular. Por ejemplo, podemos tener una que se ocupe a el manejo de los usuarios, otra para un blog y otra para una tienda ó carrito de compras. Cada app incluye su propio código, modelos, vistas, plantillas y rutas. Sería como un “módulo” independiente que podemos reutilizar en otros proyectos si queremos. Un proyecto Django grande normalmente se divide en varias apps, cada una encargada de una función concreta. Así, el código queda ordenado, más fácil de mantener y de agrandar cuando nuestra aplicación comienza a crecer.

# ¿Qué es el flujo request-response en django?

El flujo **request-response** en Django es el proceso que ocurre cuando un usuario solicita una página. Primero, el navegador envía una request (petición) al servidor. Django recibe esa request y busca en las URLs qué vista (View) debe manejarla. La vista hace el proceso de la información, por ejemplo, consultando la base de datos, y crea una respuesta. Luego, esa respuesta se convierte en un response, generalmente HTML, que se envía al navegador del usuario. Así, el usuario ve la página que pidió. Este secuencia se repite cada vez que hace clic o navega.



# ¿Qué es el concepto de ORM (Object-Relational Mapping)?

El **ORM (Object-Relational Mapping)** es una manera de trabajar con bases de datos usando código en lugar de escribir **SQL directamente**. En Django, el ORM nos permite crear, leer, actualizar y borrar datos usando clases y objetos de Python. Por ejemplo, podemos definir un modelo como una clase y Django lo convierte en una tabla en la base de datos. Esto, lo que hace es que en lugar de hacer consultas SQL complicadas, solo escribes métodos de Python. El código es más fácil de entender, más seguro y rápido de desarrollar.

1. ¿Qué son los templates en django?

Los templates en Django, básicamente son archivos que van a definir cómo se verá la información en la página web. Normalmente son archivos HTML con un lenguaje especial de Django que permite insertar datos dinámicos, como nombres de usuarios o listas de productos. Cuando una vista envía datos, el template los coloca en el lugar correcto dentro del HTML. Así, puedes crear páginas que cambian según el contenido sin que tengamos que escribir todo el código a mano nuevamente. Los templates son la parte visual de tu aplicación web.

1. Leé con detenimiento este django at a glance:

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

Básicamente lo que encontramos en esa URL, es lo que venimos hablando en este documento, que Django es un framework que nos permite crear aplicaciones web rápido y fácil. Definimos nuestros modelos en Python, que van a representar tablas de la base de datos. Luego, Django nos crea esas tablas y nos da una API para guardar y que consultemos datos sin necesidad de escribir SQL. También nos genera automáticamente un panel de administración profesional para manejar nuestros contenidos. Organizas las rutas con URLs limpias y nos creas vistas que controlan qué nos muestra. Usa templates para diseñar las páginas web. Todo está separado en partes claras, modelos, vistas y plantillas, pero funciona de manera integrado.

1. ¿Cómo se lo instala?

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

Primero, instalamos Python desde python.org. Para verificar la instalación, escribimos lo siguiente:

**python --version**

Luego, instalamos Django con este comando:

**pip install Django**

Tenemos que comprobar que Django lo instalamos bien, abrimos Python escribiendo:

**python**

Dentro del intérprete de Python, escribimos:

**import django**

**print(django.get\_version())**

Si nos aparece el número de versión, está instalado correctamente. Django usa SQLite por defecto, así que no necesitamos configurar una base de datos.

1. Glosario

**Framework:** es un conjunto de herramientas y componentes que permiten crear aplicaciones web de manera más rápida y estructurada, evitando tener que desarrollar todo desde cero.

**Inyecciones SQL:** Una inyección SQL es cuando alguien mete código malicioso en un formulario o URL para engañar la base de datos y robar, borrar o cambiar información.

**XSS (Cross-site Scripting):** es un tipo de ataque donde se inserta código malicioso (como JavaScript) en una web, para que otros usuarios lo ejecuten y así robar datos o suplantar identidad, robar cookies de sesión, redireccionar, etc.

**Falsificación de solicitudes (CSRF):** es un ataque donde un sitio malicioso hace que tu navegador envíe acciones no autorizadas a otra web donde estás conectado, como cambiar tu contraseña sin permiso.

**Extensiones Django (paquetes):** las extensiones o paquetes de Django son complementos que agregan funciones extra, como autenticación avanzada, creación de APIs o manejo de imágenes, para ampliar tu proyecto fácilmente.

1. Fuente

[Django Project](https://www.djangoproject.com/)

[Django Documentation](https://docs.djangoproject.com)

[Security in Django](https://docs.djangoproject.com/en/5.2/topics/security/)

[Django Security Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Django_Security_Cheat_Sheet.html)