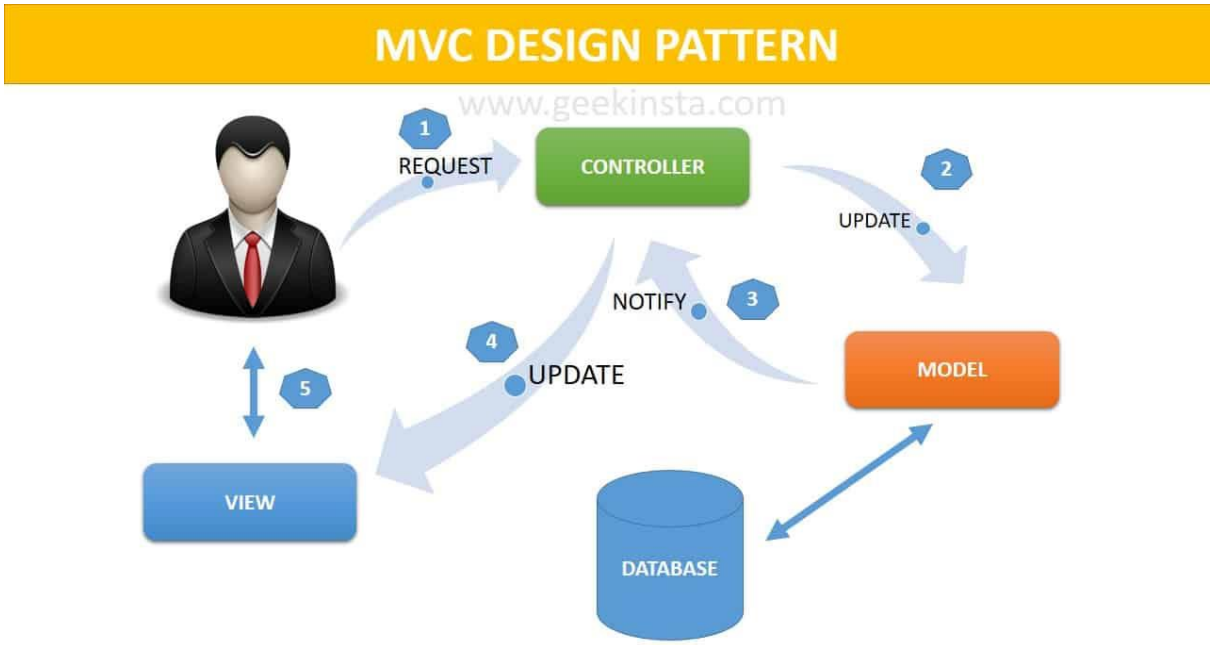
Diferencia entre la arquitectura MVC y MVT

Una pregunta que hacen muchos desarrolladores novatos es la diferencia entre MVC y MVT. Analicemos la diferencia entre estos dos patrones de forma sencilla.

Model **V**iew **C**ontroller, conocido como MVC separa el código como tres componentes. MVC separa la lógica empresarial y la capa de presentación entre sí. Se utilizaba tradicionalmente para interfaces gráficas de usuario (GUI) de escritorio. Hoy en día, la arquitectura MVC se ha vuelto popular para diseñar aplicaciones web y aplicaciones móviles.



- **Modelo** : esta capa se ocupa de la lógica relacionada con los datos. Por ejemplo, puede recuperar, cambiar y guardar datos en la base de datos.
- **Ver** : podemos llamarlo la capa de presentación. Se encarga de recopilar datos del modelo o usuario y presentarlos. En una aplicación web, todo lo que se muestra en el navegador se incluye en Ver.
- **Controlador** : controla el flujo de datos y la interacción entre la vista y el modelo. Por ejemplo, un controlador, basado en una solicitud o acción,

recopilará datos de una base de datos con la ayuda de Model y los enviará al usuario a través de Vistas.

Ventajas

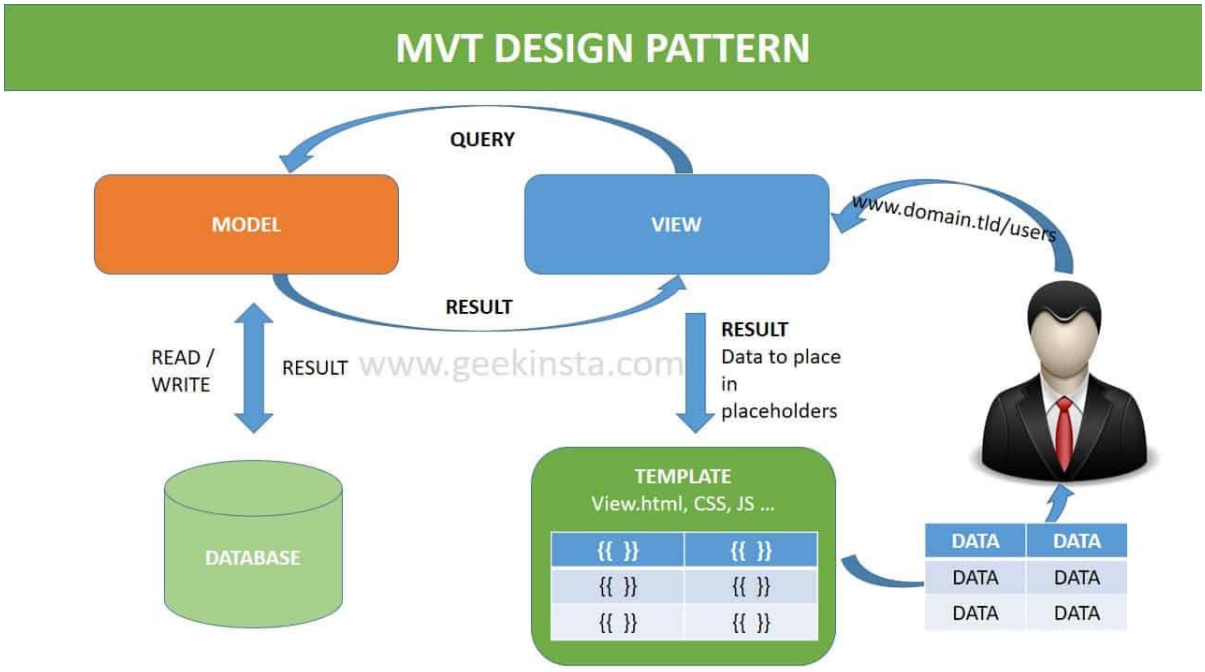
- Facilita el desarrollo de grandes aplicaciones.
- Es fácil para varios desarrolladores colaborar y trabajar juntos.

Desventajas

- La vista está controlada por el modelo y el controlador.
- No apto para pequeñas aplicaciones.

¿Qué es MVT?

Model **Vi**ew **T**emplate, ampliamente conocido como MVT es otro patrón de diseño similar a la MVC. Al igual que MVC, el patrón de diseño MVT también separa el código en tres partes.



- Modelo: igual que el modelo en MVC. Contiene el código responsable de tratar los datos y las bases de datos.
- Vista: en el patrón de diseño MVT, la vista decide qué datos deben mostrarse

- Plantilla: las plantillas se utilizan para especificar una estructura para una salida. Los datos se pueden completar en una plantilla mediante marcadores de posición. Define cómo se presentan los datos. Un ejemplo es una vista de lista genérica que podemos usar para mostrar un conjunto de registros de la base de datos.

Ventajas

- Menos acoplado.
- Adecuado para aplicaciones de pequeña a gran escala.
- Fácil de modificar.

Desventajas

- A veces, comprender el flujo puede resultar confuso.
- La modificación de modelos / vistas debe realizarse con cuidado sin afectar las plantillas.

Diferencia entre MVC y MVT

La principal diferencia entre MVC y MVT es que en un patrón de controlador de vista de modelo, tenemos que escribir todo el código específico del control. Pero en un MVT, la parte del controlador está a cargo del marco en sí.

Déjame explicarte esto con un ejemplo. Imagina que tienes que mostrar una lista de libros que tienes en una biblioteca, almacenada una tabla llamada *libros* . En la arquitectura MVC, debe escribir el código para obtener la lista de libros de la base de datos, escribir la capa de presentación (estoy hablando de HTML, CSS), mapearla con una URL y enviarla al usuario.

Pero en marcos como Django (utiliza la arquitectura MVT), no tiene que escribir ningún código relacionado con la obtención de datos de la base de datos y mapearlos con la URL. Todas estas actividades son manejadas por el propio marco. Lo único que tiene que hacer es decirle al marco qué datos deben presentarse al usuario (tabla Libros). Luego, el marco creará una vista basada en los datos y se la enviará al usuario.

El patrón MVC clásico funciona administrando el estado de una aplicación. Cuando un usuario realiza una acción o realiza una solicitud, se llama a una acción en el controlador. Luego, el controlador le dice al modelo que haga cambios y actualice la vista o devuelve una vista basada en un modelo. Por lo tanto, podemos decir que la Vista está controlada por el Controlador y el Modelo.

Sin embargo, MVT adopta un enfoque ligeramente diferente. Cuando un usuario realiza una solicitud HTTP, la vista correspondiente realiza una consulta en el modelo y recopila el conjunto de resultados del modelo. Luego, la Vista completa el resultado en una plantilla y se la envía al usuario.

A diferencia de MVC, la vista no se combina con un modelo. Esto hace que MVT esté débilmente acoplado y sea fácil de modificar.