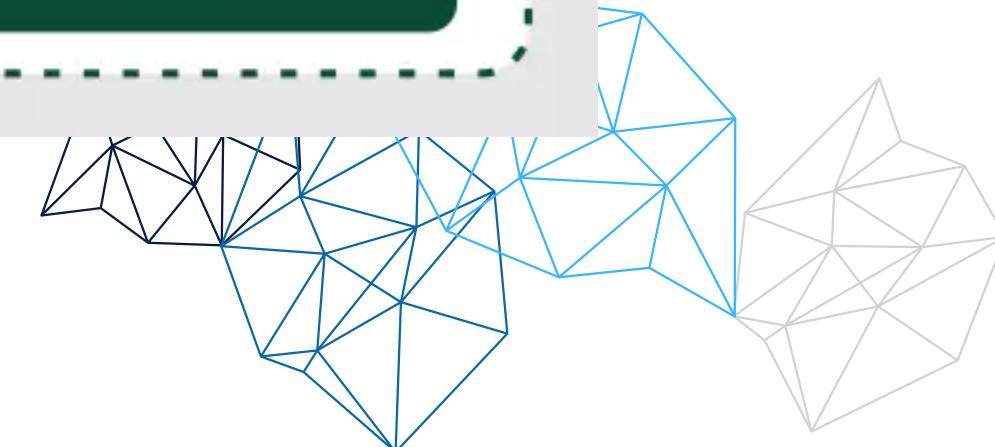
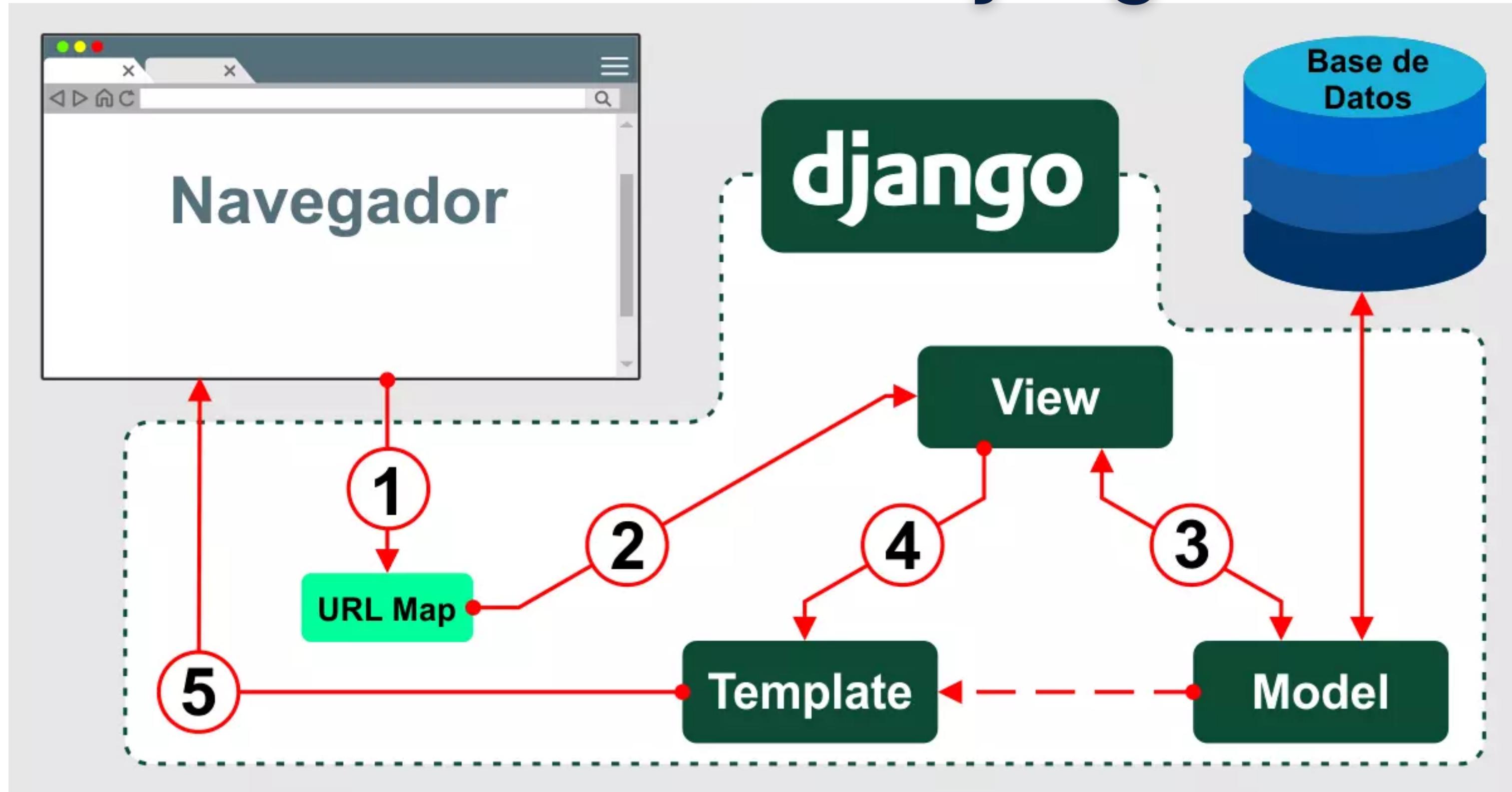


Introducción a Django

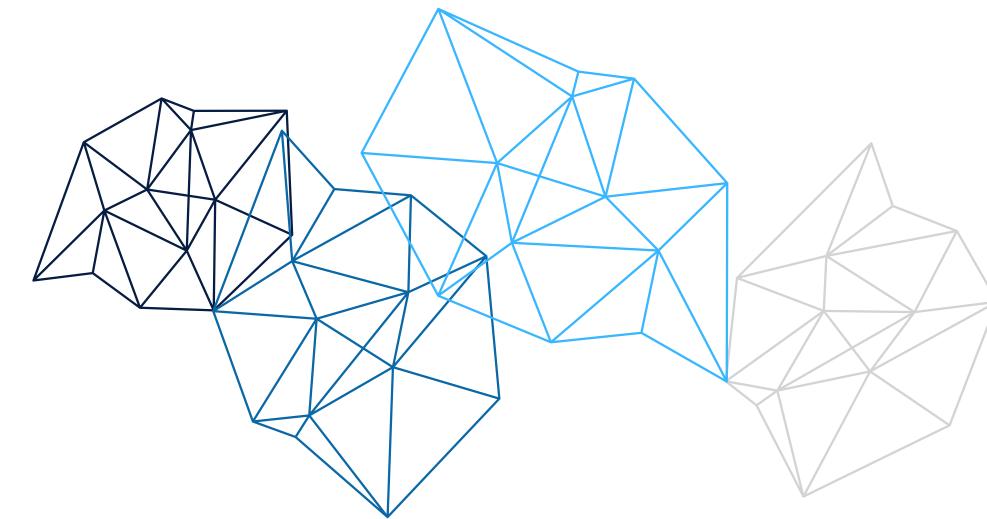


¿Qué es Django?

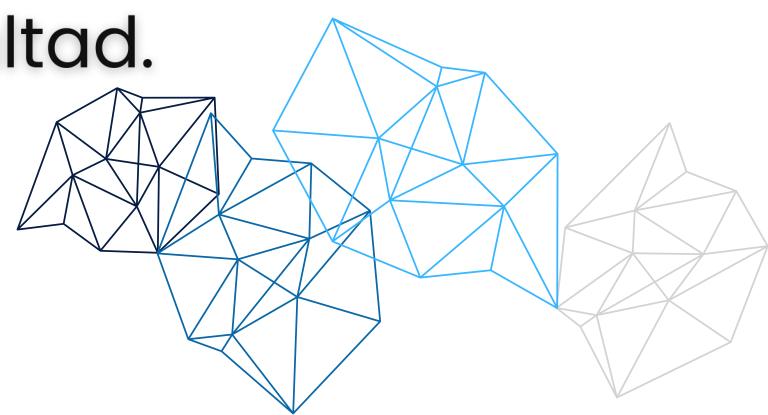
Django es un framework de desarrollo web de alto nivel y de código abierto, escrito en Python. Fue creado con el objetivo de simplificar y agilizar el desarrollo de aplicaciones web complejas al proporcionar una estructura y conjunto de herramientas predefinidas.

Algunas características importantes de Django son:

- Diseño basado en el patrón de arquitectura MVC (Modelo-Vista-Controlador): Django sigue el principio del patrón de diseño MVC, lo cual ayuda a separar la lógica de negocio, la presentación y el acceso a los datos en diferentes componentes.
- ORM (Object-Relational Mapping): Django incluye un potente ORM que permite interactuar con la base de datos utilizando objetos de Python en lugar de escribir consultas SQL directamente. Esto facilita la creación, consulta, modificación y eliminación de registros en la base de datos.



- Administrador de Django: Proporciona una interfaz de administración automática para las aplicaciones web. Con solo unas pocas líneas de código, se puede generar una interfaz de administración completa y personalizable que permite gestionar los datos del sitio de manera sencilla.
- Enrutamiento de URL: Django utiliza un enrutador de URL para mapear las URL de la aplicación a las funciones y vistas correspondientes. Esto facilita la definición de rutas y la gestión de las solicitudes entrantes.
- Plantillas: Django ofrece un sistema de plantillas que permite separar el diseño visual del código Python. Las plantillas son archivos HTML que pueden contener código Python incrustado y se utilizan para generar la interfaz de usuario de las aplicaciones web.
- Soporte para seguridad: Django incluye características de seguridad integradas, como protección contra ataques de inyección de código, protección contra ataques CSRF (Cross-Site Request Forgery), autenticación de usuarios y gestión de permisos.
- Internacionalización y localización: Django proporciona soporte integrado para la internacionalización y localización de aplicaciones web. Esto permite desarrollar aplicaciones que pueden adaptarse a diferentes idiomas y regiones sin dificultad.

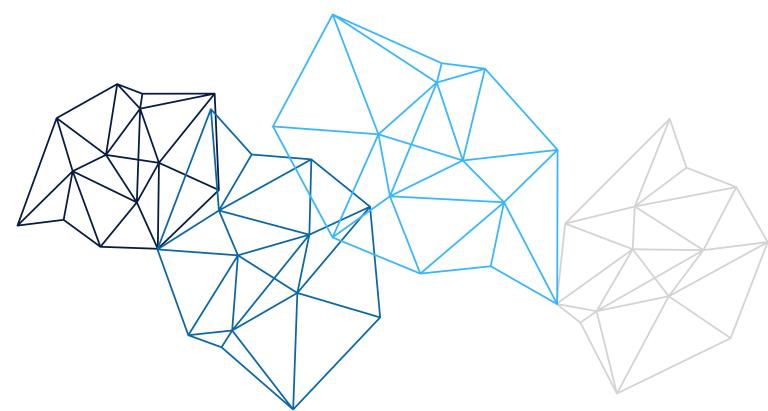


Django es utilizado ampliamente en la industria debido a su enfoque en la eficiencia, la reutilización del código y la escalabilidad. Es utilizado por empresas y desarrolladores para crear una amplia variedad de aplicaciones web, desde sitios web simples hasta plataformas complejas y de alto tráfico.

¿Qué es un framework?

Un framework es un conjunto de herramientas, bibliotecas y componentes predefinidos que proporcionan una estructura y abstracciones para facilitar el desarrollo de software. Es una infraestructura de software que define las reglas, convenciones y patrones de diseño para resolver problemas comunes en un dominio específico.

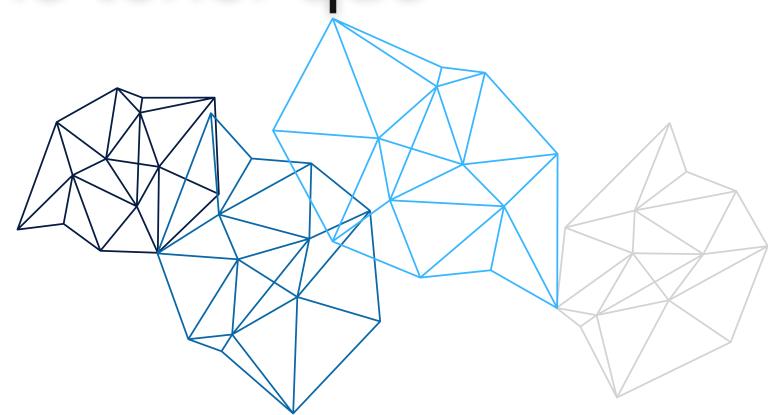
Los frameworks están diseñados para acelerar el proceso de desarrollo al proporcionar una base sólida y reutilizable sobre la cual construir aplicaciones. Al utilizar un framework, los desarrolladores pueden centrarse en la lógica específica de su aplicación sin tener que preocuparse por implementar aspectos básicos o repetitivos, como el manejo de solicitudes web, la interacción con bases de datos, la seguridad o la gestión de sesiones.



Los frameworks generalmente siguen un conjunto de principios y patrones de diseño establecidos, como el patrón de arquitectura Modelo-Vista-Controlador (MVC) o el patrón de Inversión de Control (IoC). Estos principios promueven la separación de preocupaciones y la modularidad del código, lo que resulta en una estructura más organizada y mantenible.

Algunos beneficios clave de utilizar un framework son:

- Productividad: Al proporcionar componentes y funcionalidades listas para usar, los frameworks permiten a los desarrolladores escribir menos código repetitivo y centrarse en la implementación de la lógica específica de la aplicación. Esto ayuda a acelerar el proceso de desarrollo y reduce la posibilidad de errores.
- Consistencia: Los frameworks establecen convenciones y patrones de diseño, lo que facilita la adopción de buenas prácticas y asegura una estructura coherente en el código. Esto es especialmente útil cuando varios desarrolladores trabajan en el mismo proyecto, ya que todos siguen las mismas reglas y pautas.
- Reutilización de código: Los frameworks suelen proporcionar bibliotecas y componentes que se pueden reutilizar en diferentes proyectos. Esto ahorra tiempo y esfuerzo al no tener que desarrollar funciones básicas desde cero cada vez.



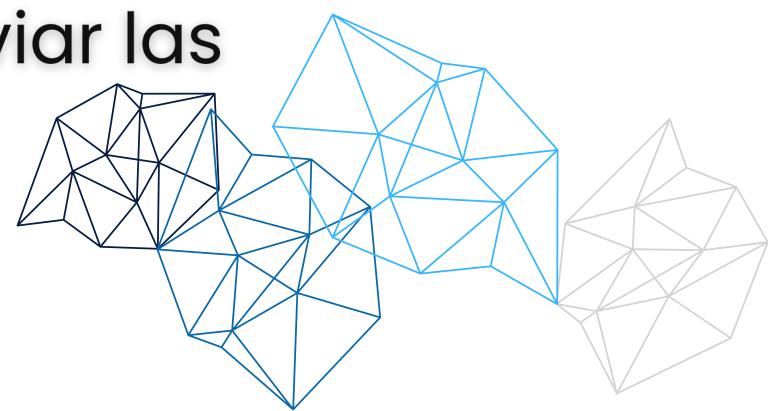
- Escalabilidad: Los frameworks suelen estar diseñados para manejar aplicaciones de gran escala y alta demanda. Proporcionan herramientas y prácticas recomendadas para optimizar el rendimiento, gestionar la concurrencia y escalar la aplicación de manera eficiente.

En resumen, un framework es una infraestructura de software que proporciona una estructura y herramientas para facilitar el desarrollo de aplicaciones, promoviendo la productividad, la consistencia y la reutilización de código.

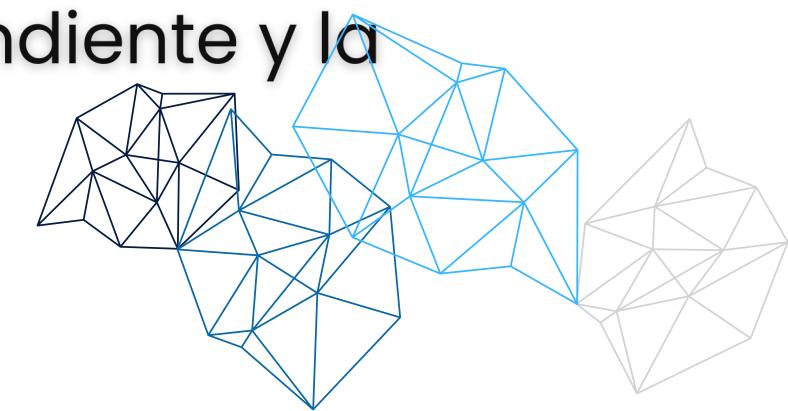
¿Qué es la arquitectura web?

La arquitectura web se refiere a la estructura y organización de una aplicación web, incluyendo la forma en que los componentes interactúan y se comunican entre sí. Hay varios enfoques arquitectónicos utilizados en el desarrollo de aplicaciones web, y los más comunes son:

- Arquitectura Cliente-Servidor: Es el enfoque más básico y ampliamente utilizado en el desarrollo web. En esta arquitectura, la aplicación se divide en dos partes principales: el cliente y el servidor. El cliente es generalmente un navegador web que solicita y muestra la interfaz de usuario al usuario final, mientras que el servidor es responsable de procesar las solicitudes, realizar operaciones de negocio, acceder a la base de datos y enviar las respuestas al cliente.



- Arquitectura de Capas: Este enfoque divide la aplicación en capas lógicas o niveles de abstracción, cada una con su responsabilidad específica. Por lo general, se utilizan tres capas principales:
 - Capa de Presentación (Interfaz de Usuario): Se encarga de la presentación y la interacción con el usuario final. Puede consistir en páginas HTML, hojas de estilo CSS, scripts de JavaScript, etc.
 - Capa de Lógica de Aplicación: Contiene la lógica de negocio y las reglas del sistema. Aquí se procesan las solicitudes, se realizan las operaciones necesarias y se coordinan las interacciones entre las diferentes partes del sistema.
 - Capa de Acceso a Datos: Gestiona el acceso y la manipulación de los datos almacenados en la base de datos u otros sistemas de almacenamiento. Se encarga de realizar consultas, actualizar datos y garantizar la integridad de la información.
- Arquitectura de Microservicios: En esta arquitectura, una aplicación se divide en un conjunto de servicios independientes y autocontenido, conocidos como microservicios. Cada microservicio tiene su propia lógica de negocio y se comunica con otros a través de interfaces bien definidas. Esto permite la escalabilidad, el despliegue independiente y la flexibilidad en el desarrollo de aplicaciones web complejas.

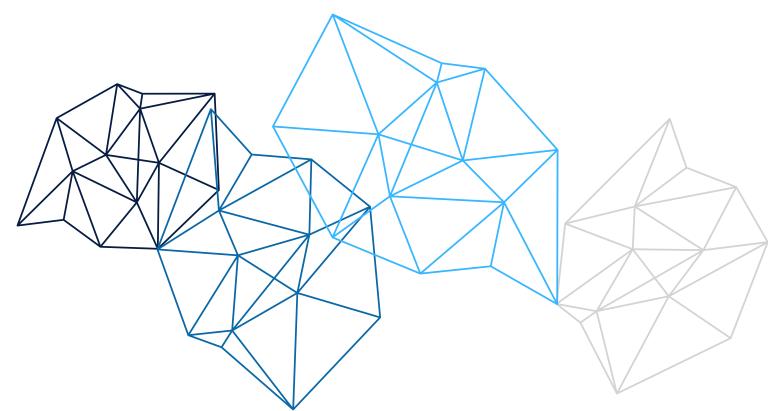


- Arquitectura Modelo-Vista-Controlador (MVC): Como se mencionó anteriormente, el patrón MVC se utiliza ampliamente en el desarrollo web y es un enfoque arquitectónico que separa la lógica de la aplicación en tres componentes principales: modelo, vista y controlador. El modelo representa los datos y la lógica de negocio, la vista es la interfaz de usuario y el controlador maneja la interacción entre el modelo y la vista.

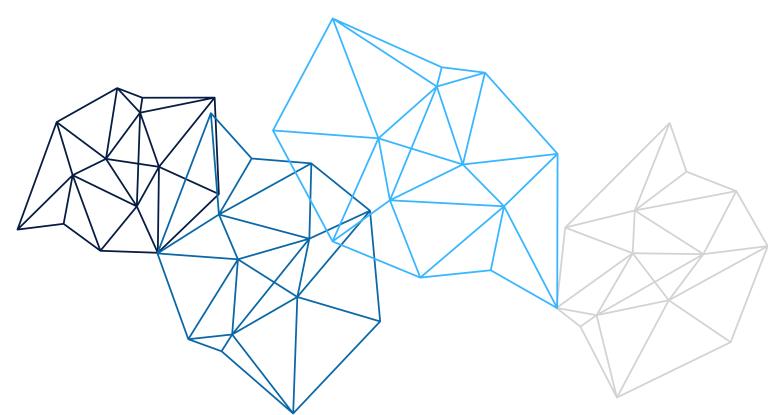
Cada arquitectura tiene sus propias ventajas y desafíos, y la elección depende de los requisitos y las necesidades específicas de la aplicación. Los desarrolladores deben considerar factores como la escalabilidad, el rendimiento, el mantenimiento y la facilidad de desarrollo al seleccionar la arquitectura adecuada para un proyecto web.

¿Qué es el patrón de arquitectura Modelo-Vista-Controlador(MVC)?

Es un enfoque de diseño comúnmente utilizado para desarrollar aplicaciones de software, especialmente aplicaciones web. Proporciona una forma de separar las responsabilidades y la lógica del programa en tres componentes principales: el modelo, la vista y el controlador.



- **Modelo:** El modelo representa los datos y la lógica de negocio de la aplicación. Es responsable de acceder, modificar y actualizar los datos, así como de aplicar las reglas y operaciones necesarias. En esencia, el modelo representa la estructura de datos subyacente y define cómo interactuar con ella.
- **Vista:** La vista es la capa de presentación de la aplicación. Se encarga de mostrar la información al usuario y de interactuar con él. La vista obtiene los datos necesarios del modelo y los muestra de una manera comprensible y adecuada para el usuario. También puede manejar eventos o acciones del usuario y comunicarlos al controlador.
- **Controlador:** El controlador actúa como intermediario entre la vista y el modelo. Recibe las interacciones del usuario a través de la vista y toma las acciones correspondientes. Se encarga de procesar las solicitudes, actualizar el modelo si es necesario y seleccionar la vista apropiada para mostrar los resultados. El controlador es el responsable de la lógica de coordinación y gestión del flujo de la aplicación.



El patrón MVC promueve la separación de preocupaciones y la modularidad, lo que facilita el desarrollo, la prueba y el mantenimiento de las aplicaciones. Cada componente tiene una responsabilidad clara y limitada, lo que permite cambios independientes en cada uno sin afectar a los demás. Además, la reutilización de componentes se facilita al mantener la lógica de negocio aislada en el modelo y la presentación en la vista.

En un flujo típico de una aplicación MVC, el usuario interactúa con la vista, que envía una solicitud al controlador. El controlador procesa la solicitud, realiza las operaciones necesarias en el modelo y selecciona la vista apropiada para mostrar los resultados actualizados al usuario. Este patrón se utiliza ampliamente en el desarrollo de aplicaciones web, pero también se puede aplicar en otros contextos, como aplicaciones de escritorio o móviles, siempre y cuando haya una separación clara entre los datos, la presentación y la lógica del programa.

En las siguientes clases seguiremos desarrollando más a fondo estos temas y llevándolos a código !

