Clase 8

MVC

Agenda

- MVC
- Aplicaciones
- Flujo de Trabajo (MVC)
- Ventajas del MVC
- Aplicaciones

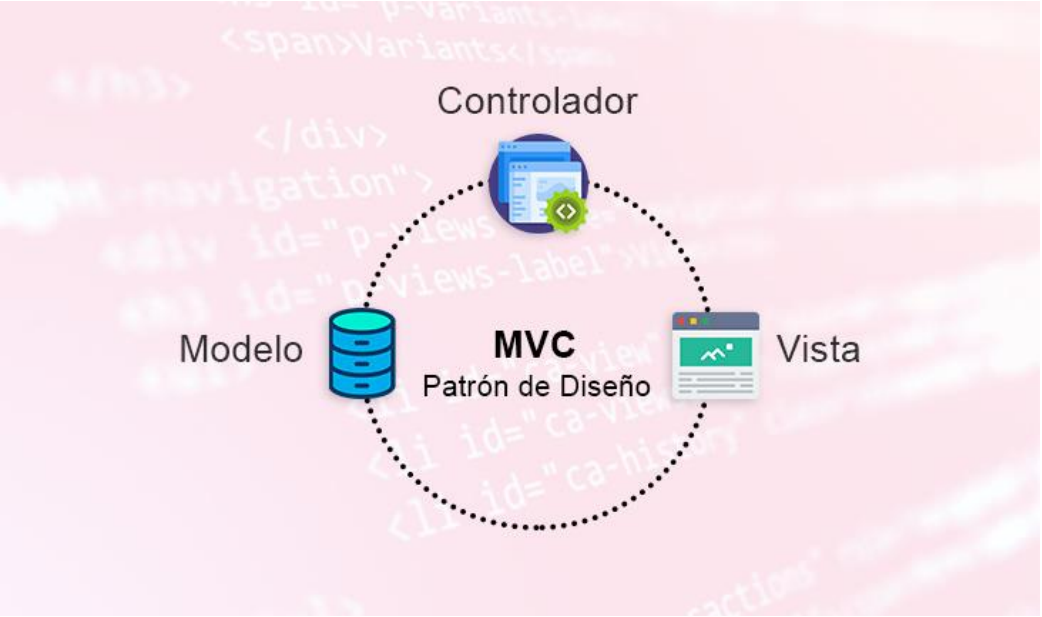
MVC

Modelo Vista Controlador

Definición, ¿Qué es?

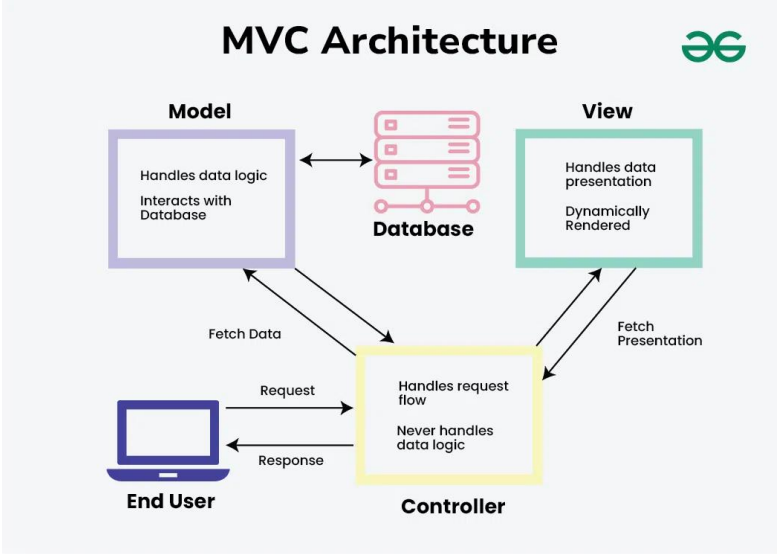
MVC

El patrón MVC (Model-View-Controller) es un patrón de diseño arquitectónico que organiza una aplicación en tres componentes principales: Modelo (Model), Vista (View) y Controlador (Controller). Su objetivo es separar las responsabilidades para facilitar el mantenimiento, la escalabilidad y la reutilización del código.



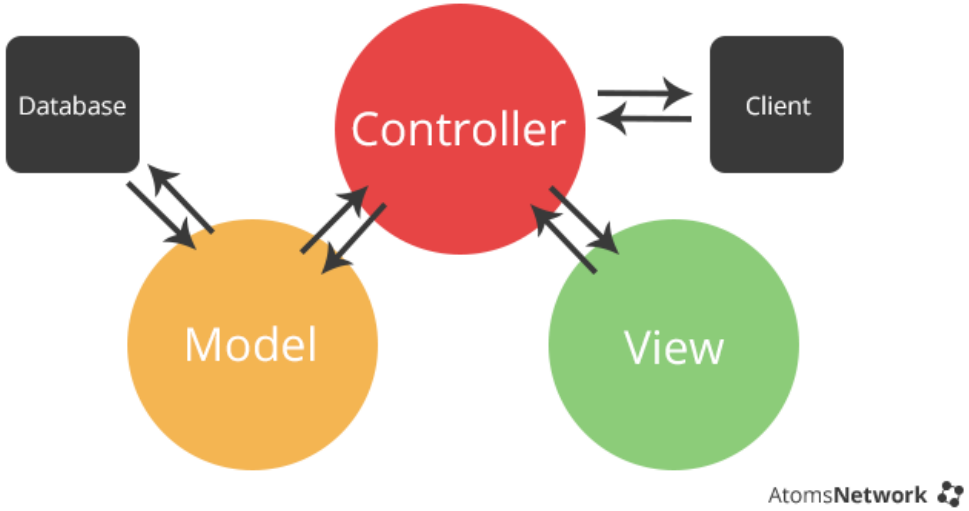
Componentes

- **Modelo:** Representa los datos y la lógica de negocio de la aplicación. Maneja las reglas de negocio, la lógica de la aplicación y la gestión de los datos persistentes (por ejemplo, bases de datos).
- **Vista:** Es responsable de la presentación de los datos al usuario. Es lo que el usuario ve e interactúa (la interfaz de usuario). La vista obtiene datos del modelo y los muestra de una forma que sea comprensible para el usuario.
- **Controlador:** Actúa como intermediario entre la vista y el modelo. Recibe las entradas del usuario desde la vista, las procesa y ejecuta las acciones necesarias, como interactuar con el modelo o actualizar la vista.



Como Aplicarlo

El enfoque MVC permite manejar de manera eficiente cómo los datos se muestran al usuario y cómo las interacciones del usuario afectan la aplicación.



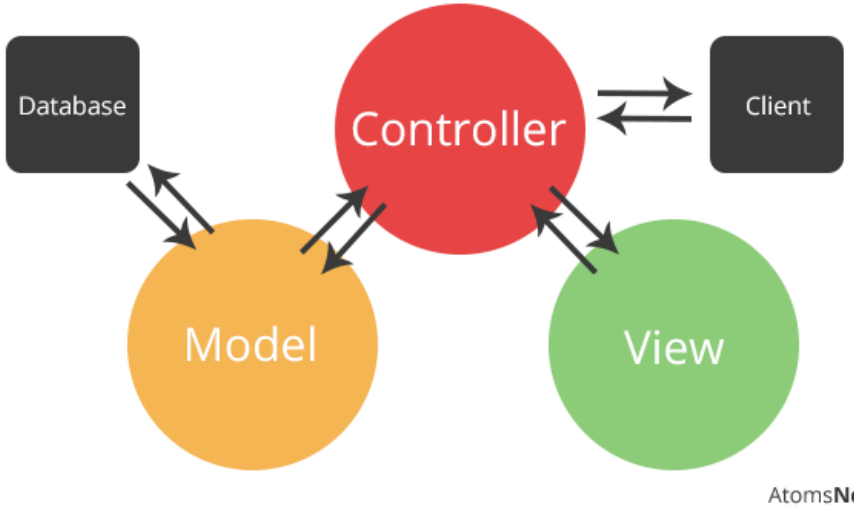
Modelo

- Es la capa lógica y de datos de la aplicación.
- Representa la estructura de los datos (tablas en bases de datos, estructuras de objetos, etc.).
- Gestiona la lógica de negocio y la interacción con la base de datos.
- Responde a solicitudes de datos realizadas por el controlador.
- Notifica a la vista cuando los datos cambian.

Ejemplo

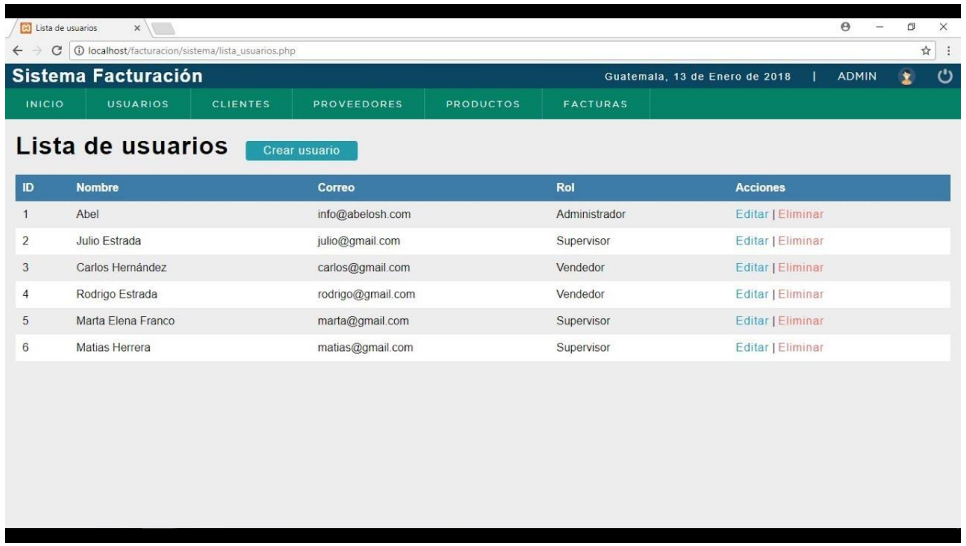
Modelo

Para un sistema de usuarios, el modelo puede incluir la definición de la entidad "Usuario" y métodos como crearUsuario, editarUsuario, o obtenerUsuarioPorID.



Vista

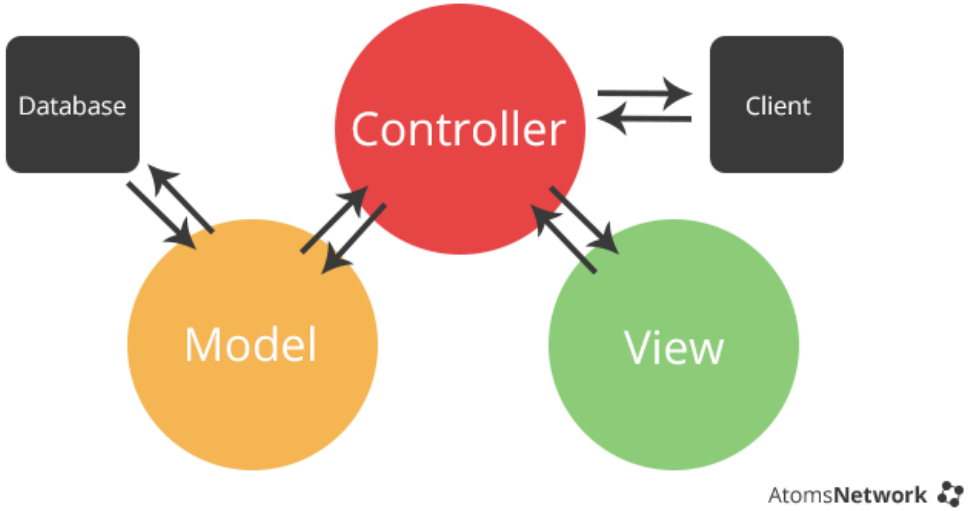
- Es la interfaz de usuario.
- Presenta los datos al usuario final (generalmente en forma visual).
- Depende del modelo, pero no lo modifica directamente.
- Se actualiza automáticamente cuando el modelo cambia, en implementaciones más avanzadas.



Un formulario de registro o una lista de usuarios renderizada en HTML.

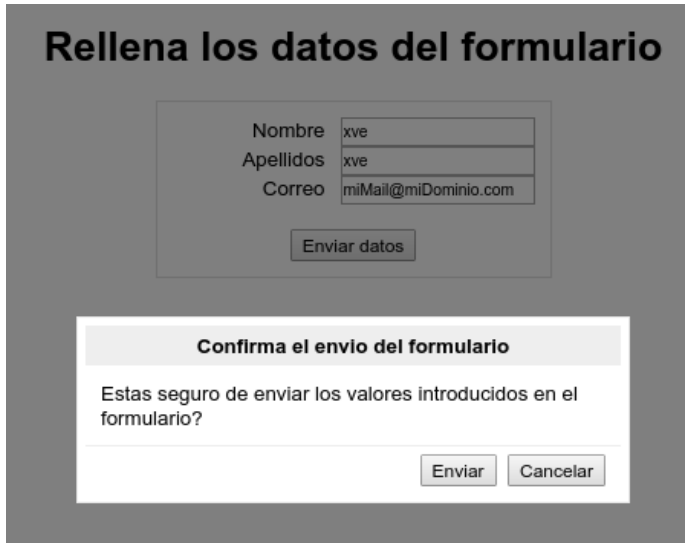
Ejemplo

Vista



Controlador

- Actúa como un intermediario entre el modelo y la vista.
- Recibe las entradas del usuario (a través de la vista) y las procesa.
- Llama a los métodos del modelo para realizar operaciones o actualizar los datos.
- Actualiza la vista en función de los resultados.

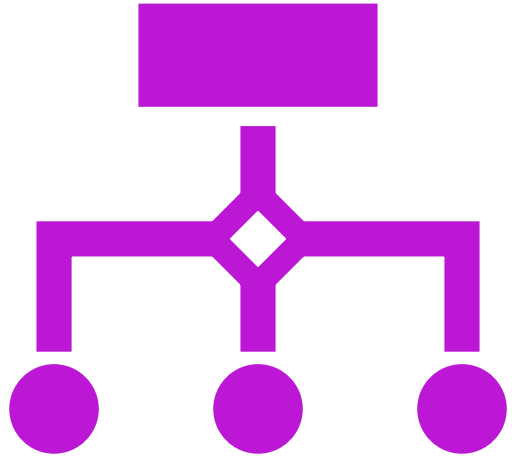


Ejemplo
Controlador

Si un usuario presiona el botón "Guardar", el controlador valida los datos, llama al método guardarUsuario del modelo, y decide si mostrar un mensaje de éxito o error.

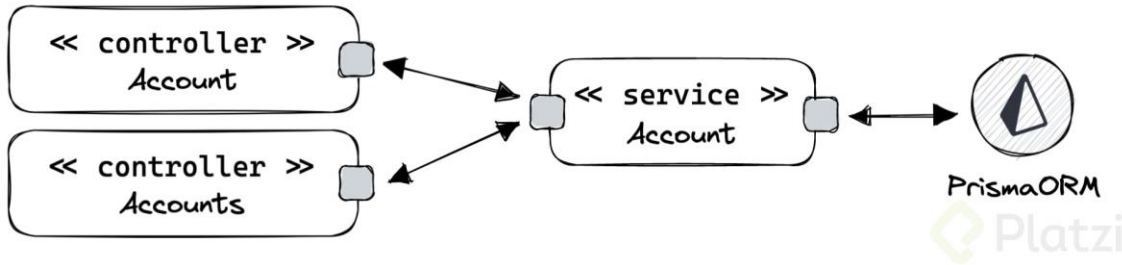
Flujo de Trabajo

- El usuario interactúa con la vista (por ejemplo, llena un formulario o hace clic en un botón).
- La vista envía la solicitud al controlador.
- El controlador procesa la solicitud, valida los datos, y realiza las operaciones necesarias en el modelo.
- El modelo actualiza los datos y notifica al controlador si hubo algún cambio.
- El controlador envía las actualizaciones a la vista, que presenta la información actualizada al usuario.



Ventajas (MVC)

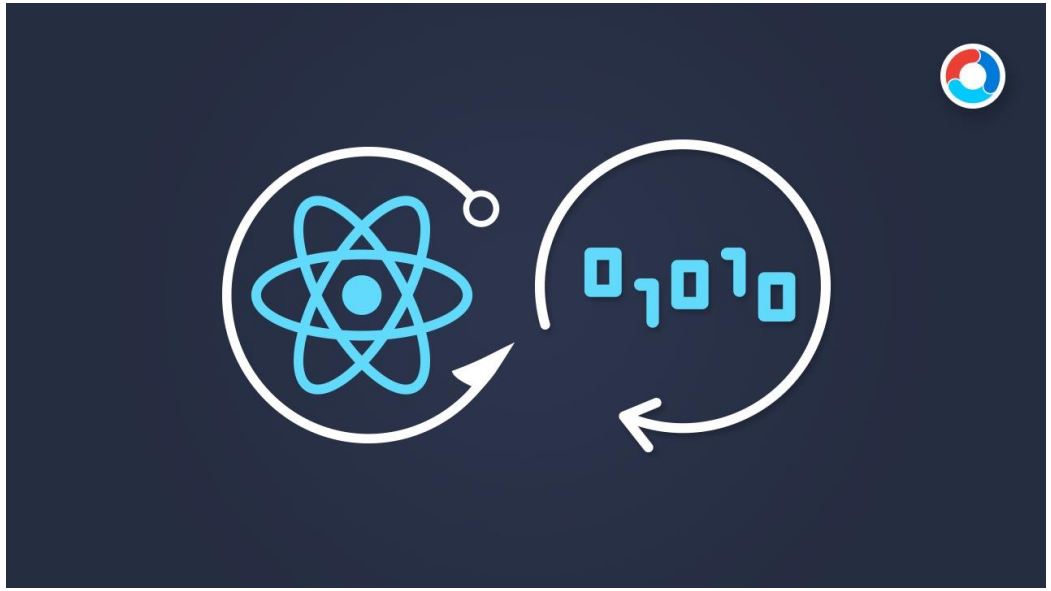
Separación de responsabilidades



- El modelo gestiona los datos.
- La vista gestiona la presentación.
- El controlador gestiona la lógica de aplicación.

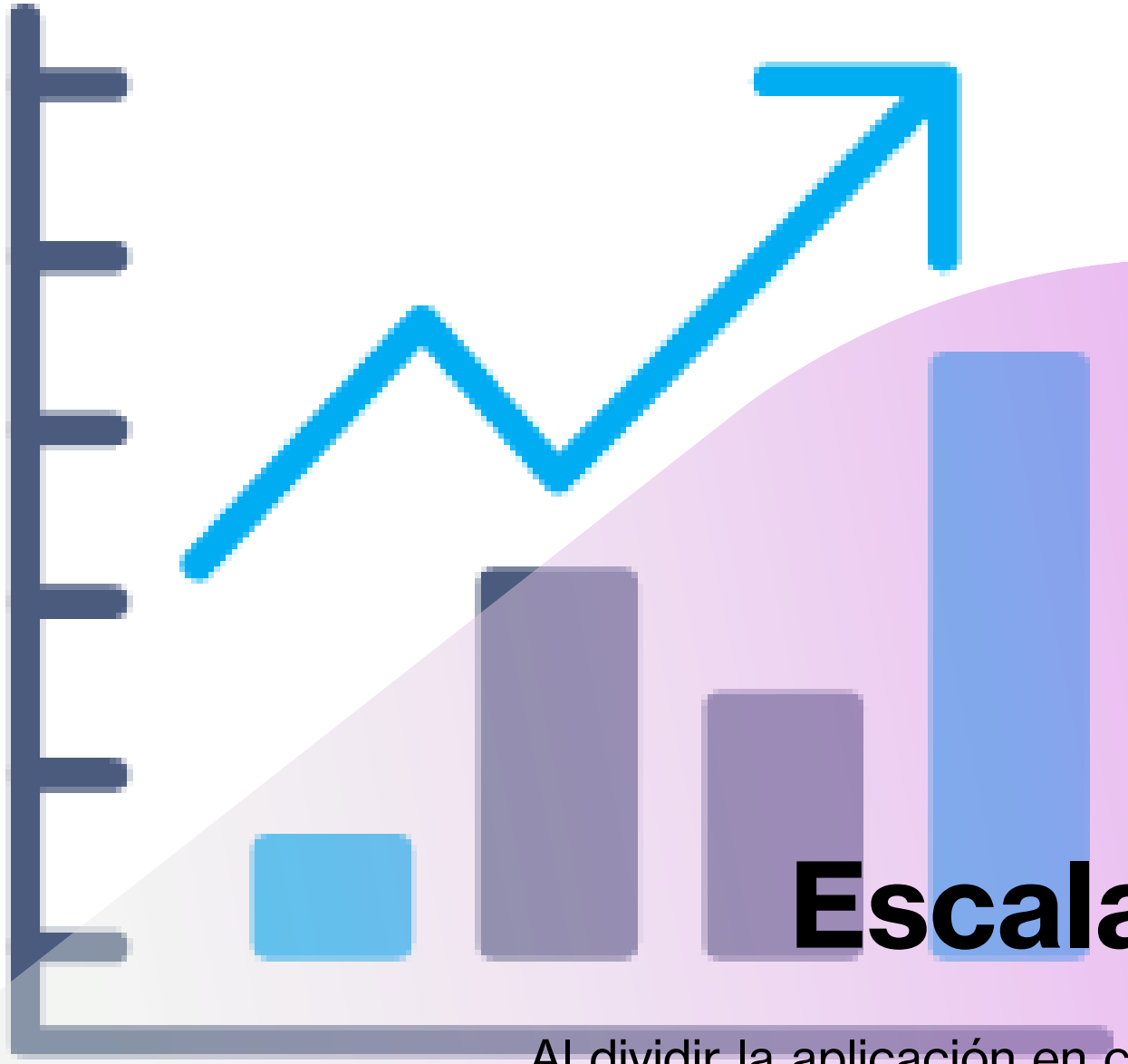
Mantenibilidad

El código está mejor organizado, lo que facilita corregir errores o implementar nuevas funcionalidades.



Reutilización del código

- Las vistas pueden reutilizar componentes.
- Los modelos pueden reutilizarse en diferentes controladores o vistas.



Escalabilidad

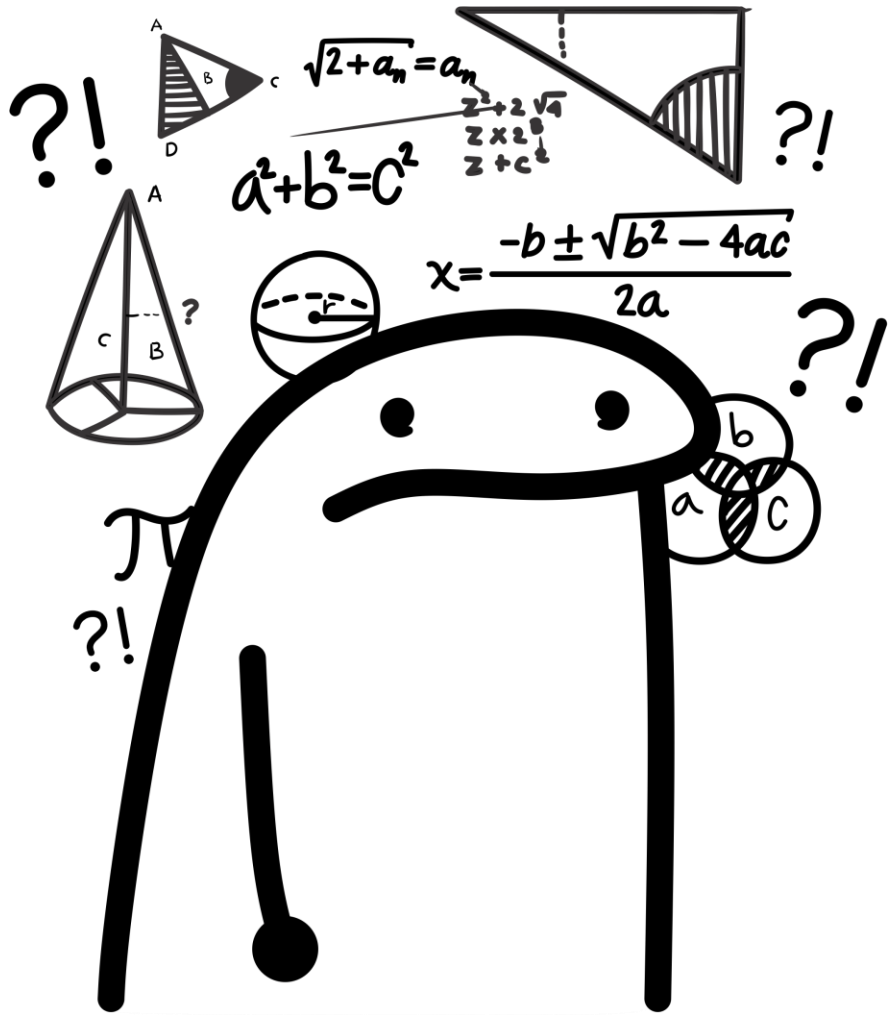
Al dividir la aplicación en capas, es más fácil añadir nuevas características o ampliarla.

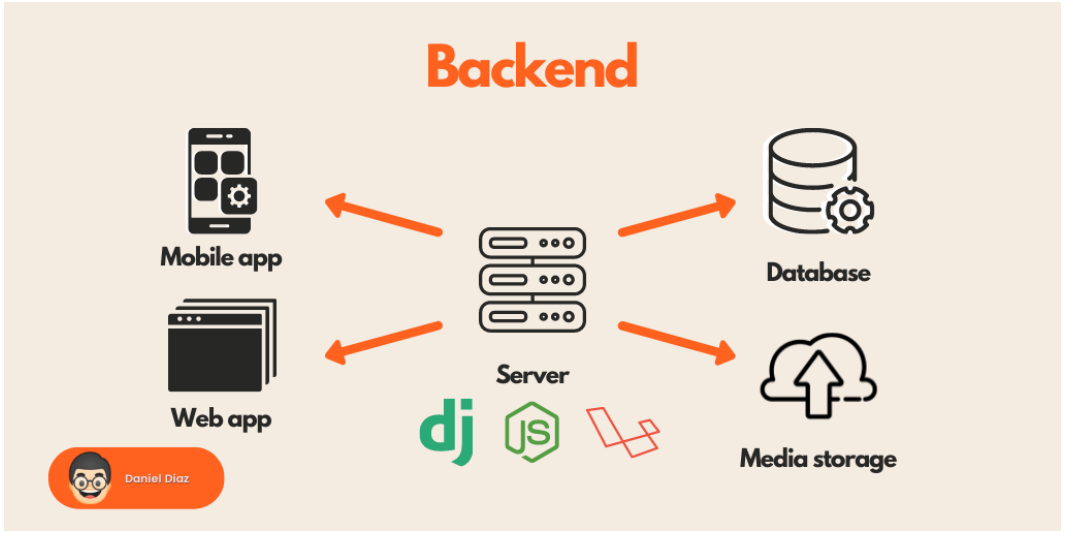
Facilidad de prueba

Las diferentes capas pueden probarse de forma independiente.

¿Dónde se aplica el patrón MVC?

El patrón MVC se usa ampliamente en el desarrollo de aplicaciones web, móviles y de escritorio.





Backend

- Laravel (PHP)
- Django (Python)
- Ruby on Rails (Ruby)
- ASP.NET (.NET)
- Spring Boot (Java)

Frontend

Aunque el frontend no implementaba MVC de manera directa, hay frameworks con conceptos inspirados en el mismo.

- Angular
- Vue
- Svelte

Dudas o Comentarios