

The background features a series of overlapping, wavy, translucent shapes in yellow, orange, red, pink, and purple on the left side. On the right side, there is a large, semi-transparent purple circle. The overall composition is modern and abstract.

# Clase 8

MVC

# Agenda

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





# MVC

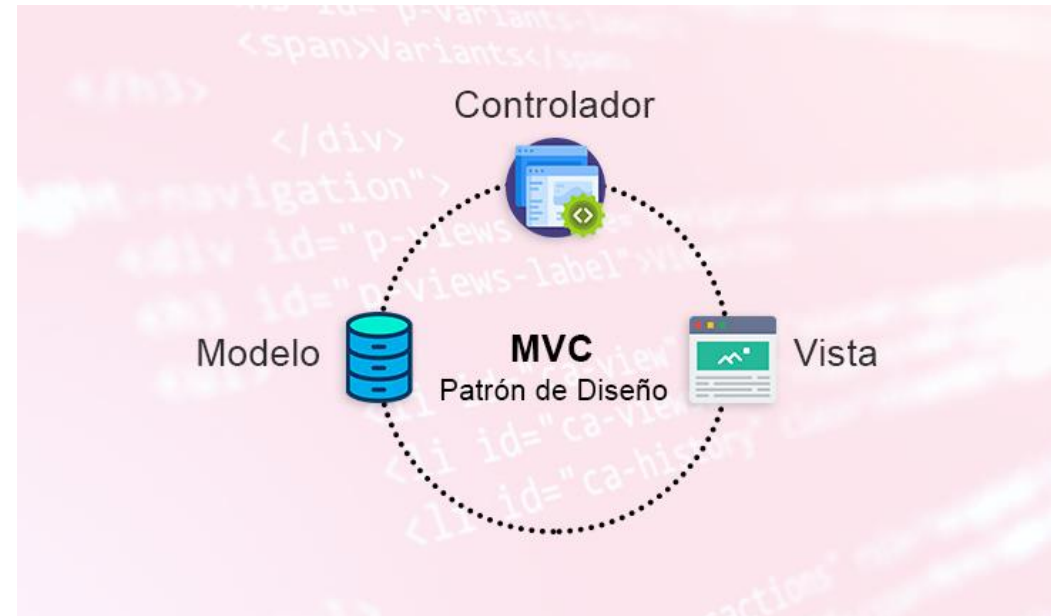
Modelo Vista Controlador

Definición, ¿Qué es?



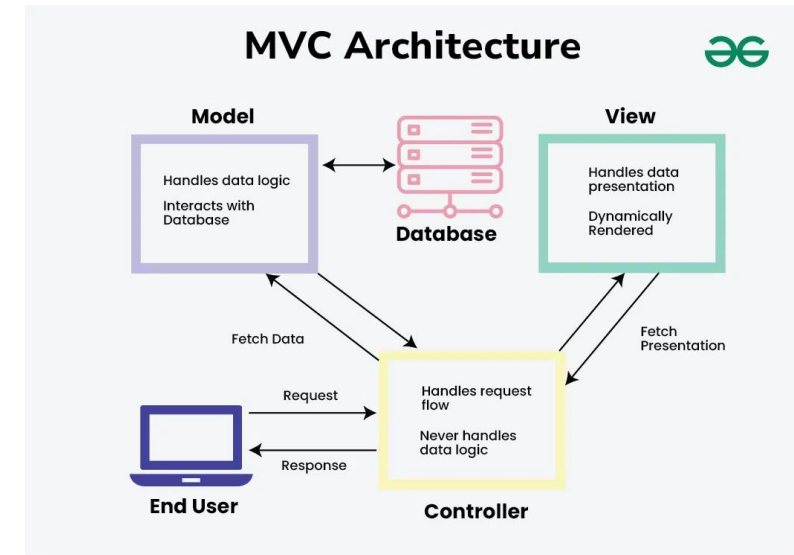
# MVC

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

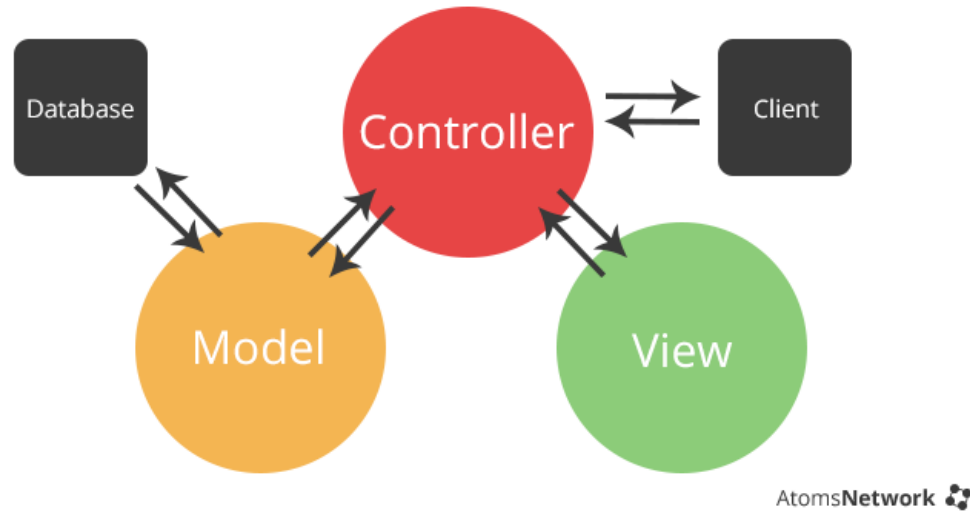
- **Modelo:** Representa los datos y la lógica de negocio de la aplicación. Maneja las reglas de negocio, la lógica de la aplicación y la gestión de los datos persistentes (por ejemplo, bases de datos).
- **Vista:** Es responsable de la presentación de los datos al usuario. Es lo que el usuario ve e 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.

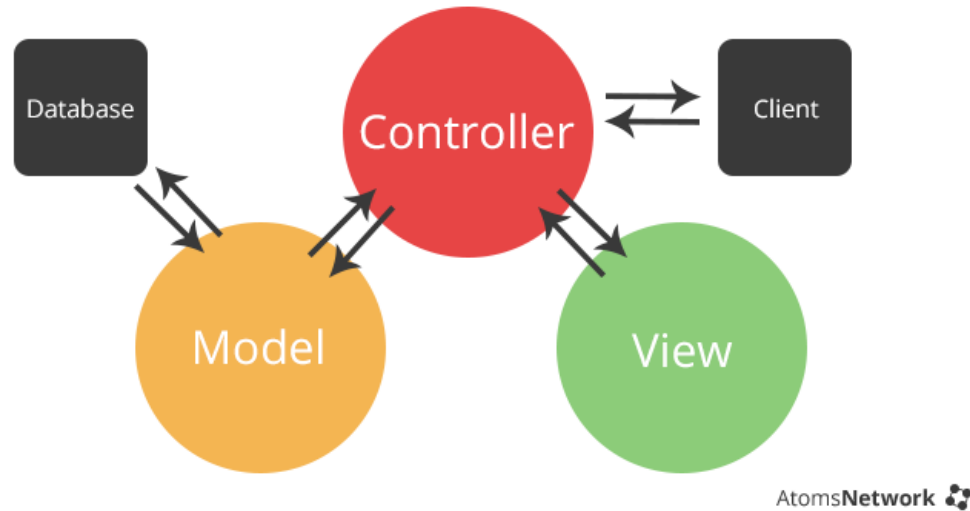


Para un sistema de usuarios, el modelo puede incluir la definición de la entidad "Usuario" y métodos como crearUsuario, editarUsuario, o obtenerUsuarioPorID.

# Ejemplo

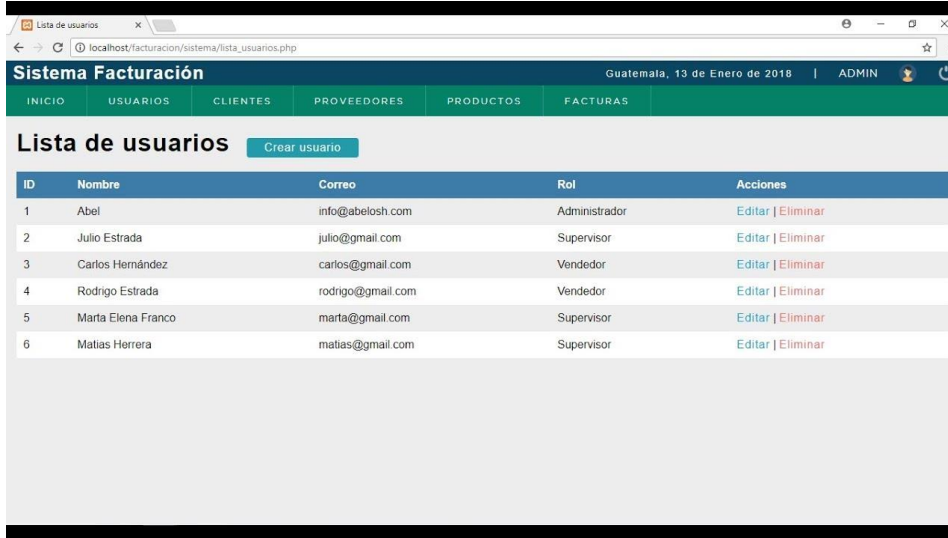
## Modelo





# 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.



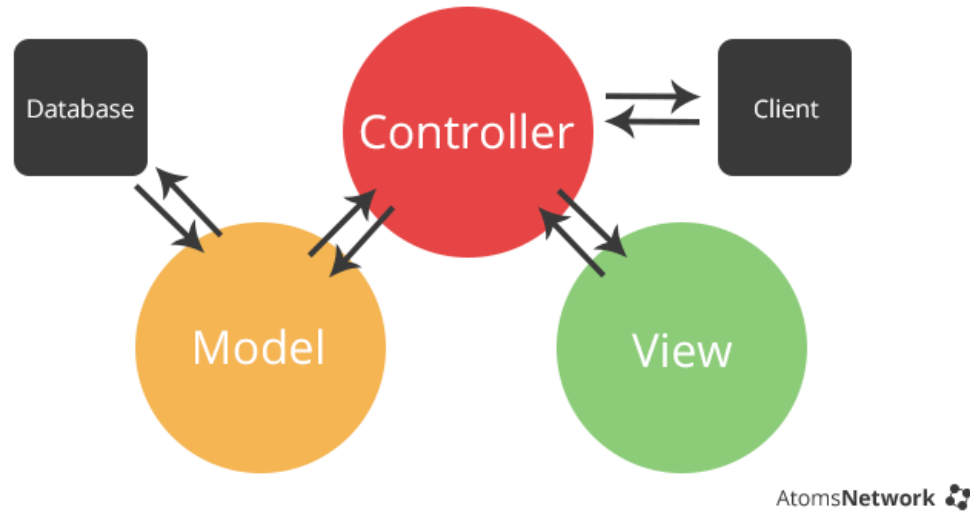
The screenshot shows a web browser window with the address bar displaying 'localhost/facturacion/sistema/lista\_usuarios.php'. The page title is 'Lista de usuarios'. Below the title is a green navigation bar with the text 'Sistema Facturación' and a date 'Guatemala, 13 de Enero de 2018'. The navigation bar includes tabs for 'INICIO', 'USUARIOS', 'CLIENTES', 'PROVEEDORES', 'PRODUCTOS', and 'FACTURAS'. The 'USUARIOS' tab is active. Below the navigation bar, there is a section titled 'Lista de usuarios' with a 'Crear usuario' button. A table with 5 columns (ID, Nombre, Correo, Rol, Acciones) displays a list of 6 users. Each user row has 'Editar' and 'Eliminar' links. The table is followed by a large empty space.

ID	Nombre	Correo	Rol	Acciones
1	Abel	info@abelosh.com	Administrador	<a href="#">Editar</a>   <a href="#">Eliminar</a>
2	Julio Estrada	julio@gmail.com	Supervisor	<a href="#">Editar</a>   <a href="#">Eliminar</a>
3	Carlos Hernández	carlos@gmail.com	Vendedor	<a href="#">Editar</a>   <a href="#">Eliminar</a>
4	Rodrigo Estrada	rodrigo@gmail.com	Vendedor	<a href="#">Editar</a>   <a href="#">Eliminar</a>
5	Marta Elena Franco	marta@gmail.com	Supervisor	<a href="#">Editar</a>   <a href="#">Eliminar</a>
6	Matias Herrera	matias@gmail.com	Supervisor	<a href="#">Editar</a>   <a href="#">Eliminar</a>

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.

**Rellena los datos del formulario**

Nombre	<input type="text" value="xve"/>
Apellidos	<input type="text" value="xve"/>
Correo	<input type="text" value="miMail@miDominio.com"/>

**Confirma el envío del formulario**

Estas seguro de enviar los valores introducidos en el formulario?

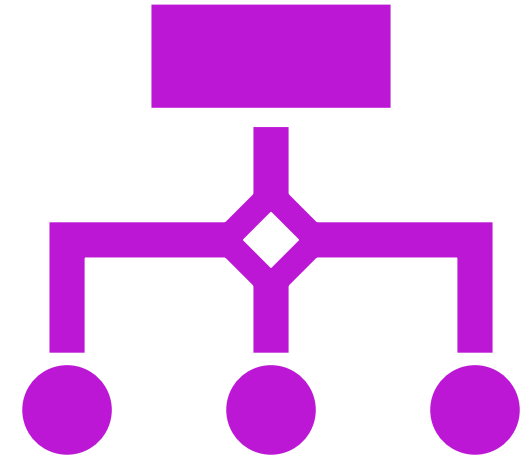
Si un usuario presiona el botón "Guardar", el controlador valida los datos, llama al método guardarUsuario del modelo, y decide si mostrar un mensaje de éxito o error.

# Ejemplo

## Controlador

# 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.

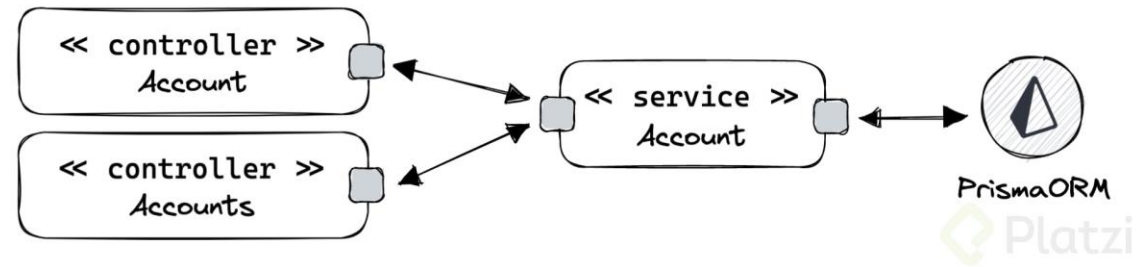


# Ventajas (MVC)

The background of the slide is a dense field of white umbrellas, viewed from a low angle looking up. The umbrellas are arranged in a way that creates a sense of depth and repetition. In the center of the image, slightly above the horizontal midline, one umbrella is a vibrant blue, making it the focal point of the visual. The lighting is soft and diffused, coming from the upper right, which creates gentle shadows and highlights on the fabric of the umbrellas.



# Separación de responsabilidades



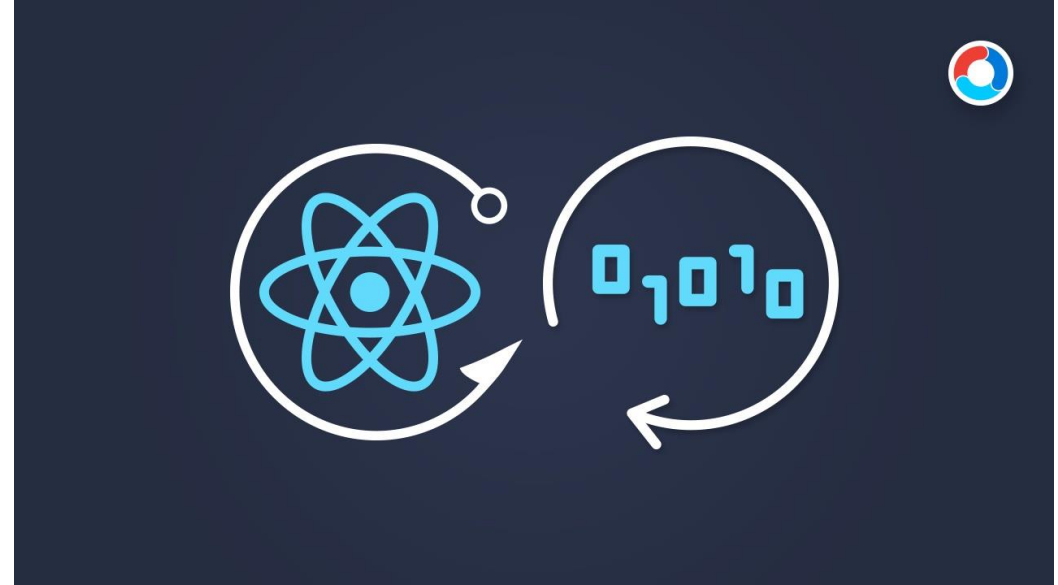
- El modelo gestiona los datos.
- La vista gestiona la presentación.
- El controlador gestiona la lógica de aplicación.

# Mantenibilidad

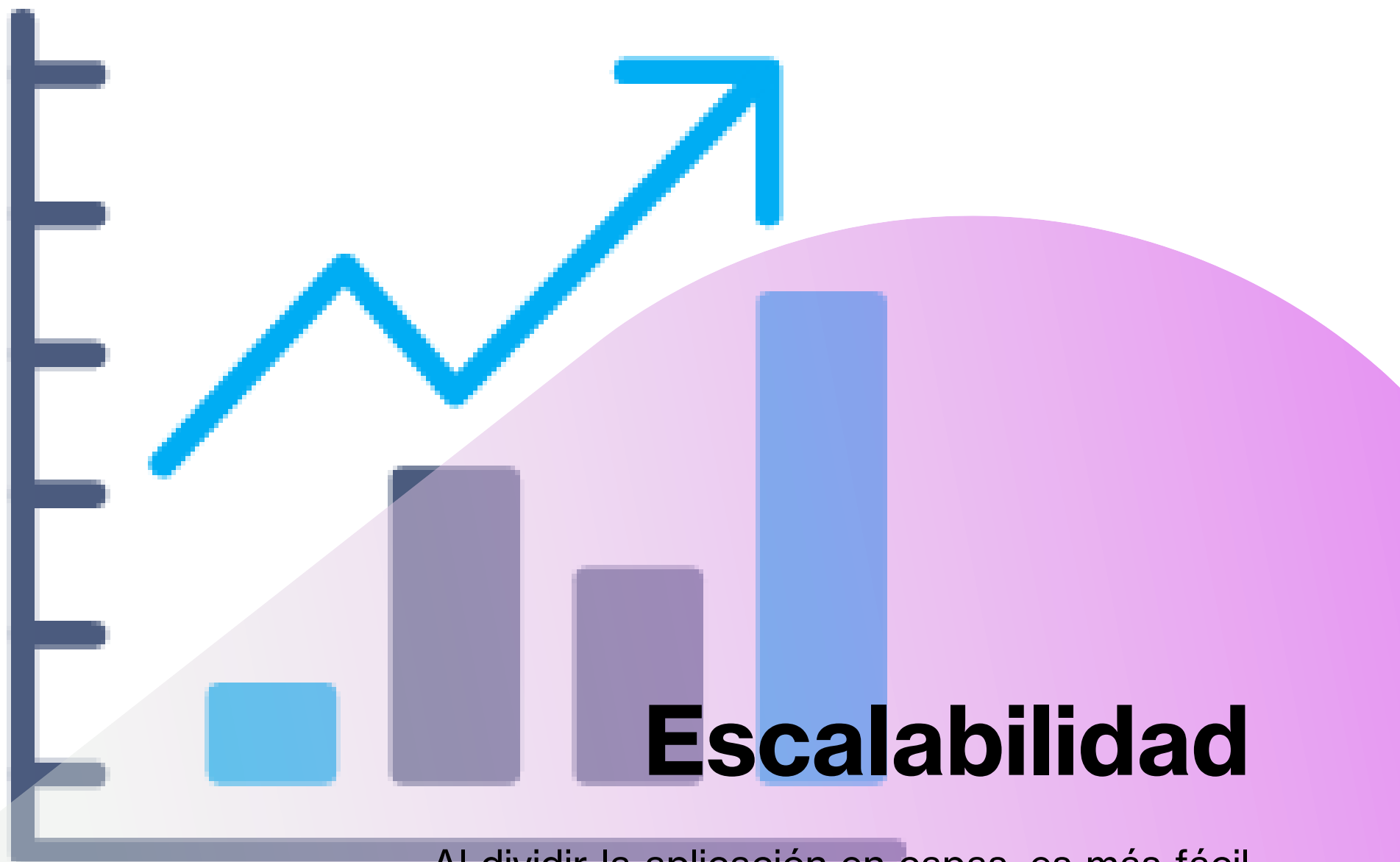
El código está mejor organizado, lo que facilita corregir errores o implementar nuevas funcionalidades.



# Reutilización del código



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

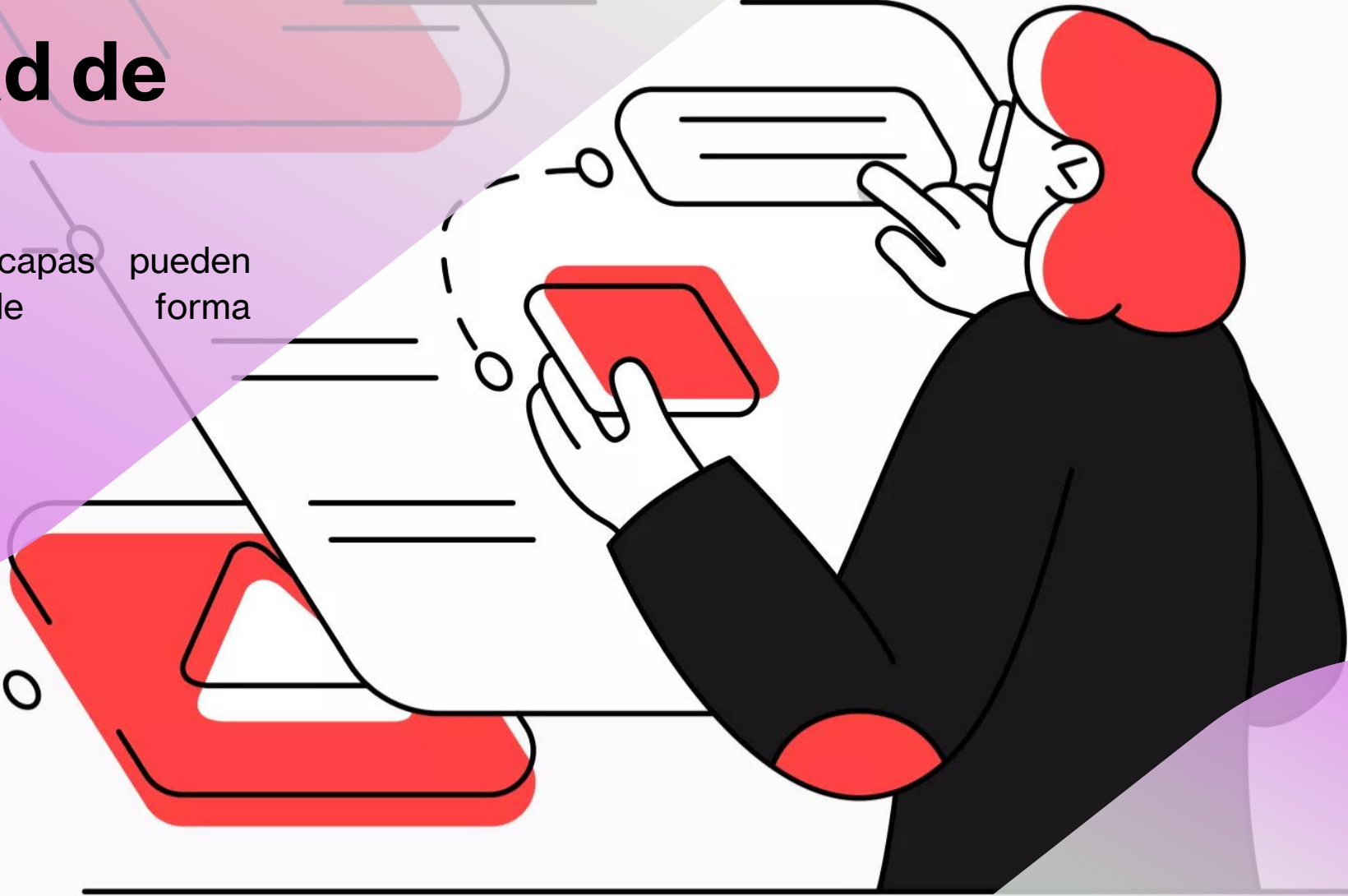


# Escalabilidad

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

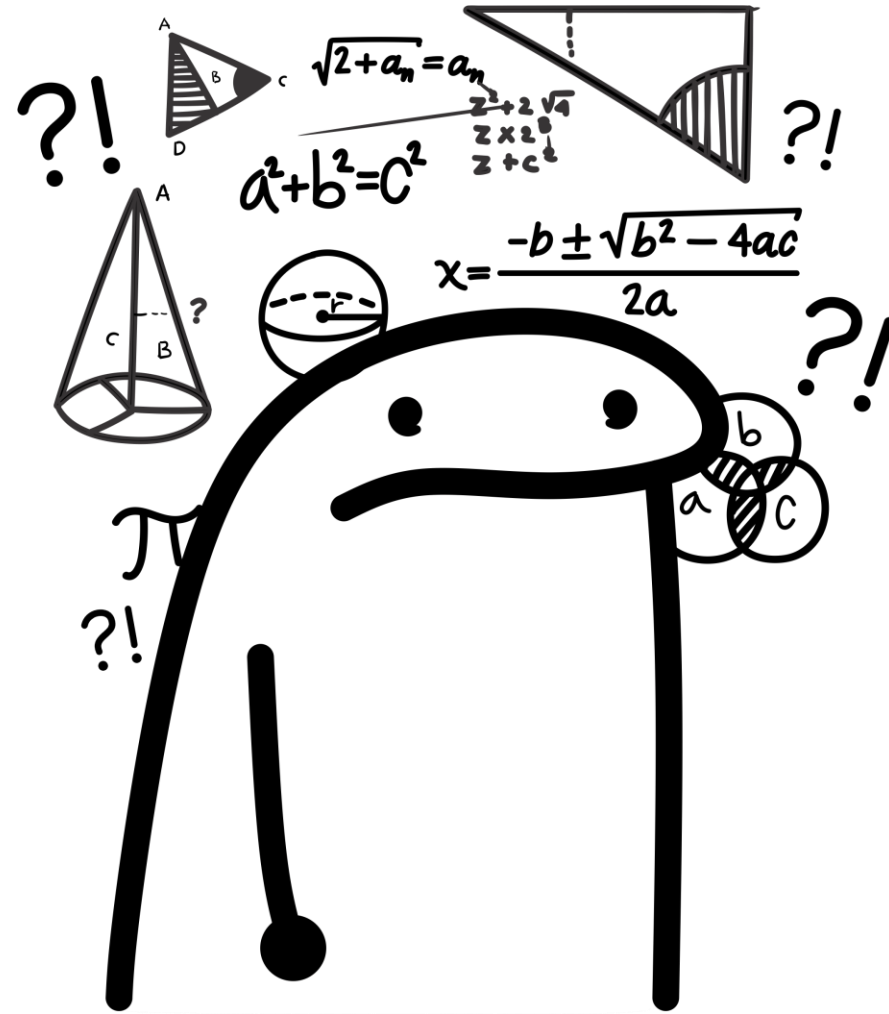
# Facilidad de prueba

Las diferentes capas pueden probarse de forma independiente.

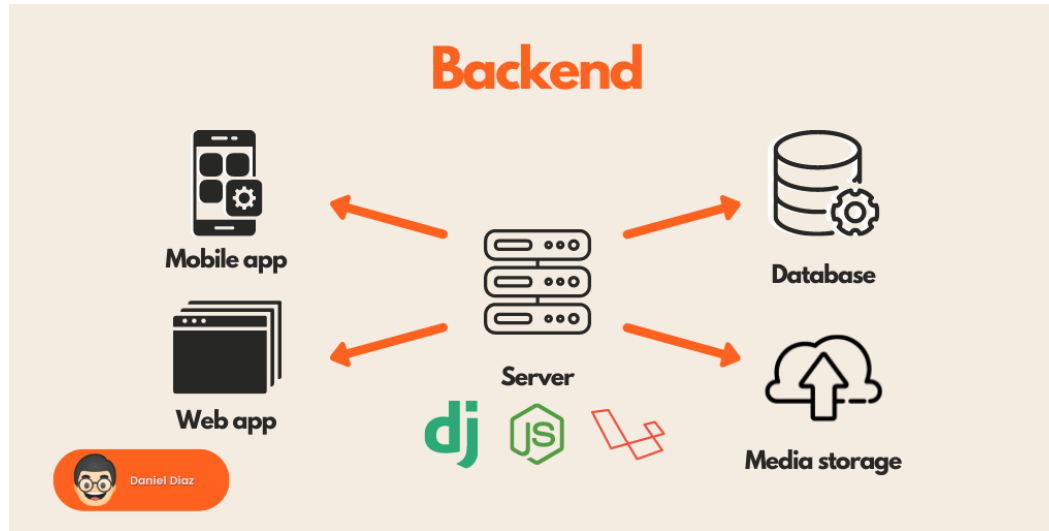


# ¿Dónde se aplica el patrón MVC?

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

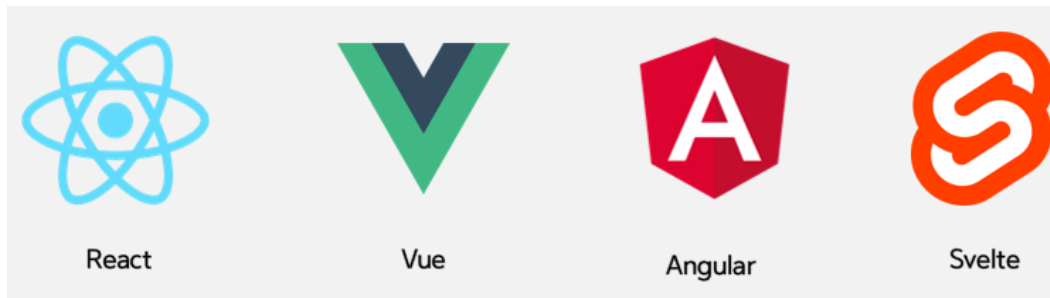






- Laravel (PHP)
- Django (Python)
- Ruby on Rails (Ruby)
- ASP.NET (.NET)
- Spring Boot (Java)

# Backend



# Frontend

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

- Angular
- Vue
- Svelte

# Dudas o Comentarios

