MODEL VIEW CONTROLLER (MVC)

Presentación – Objetos

CFP Nº8 FORMACION PROFESIONAL SMATA

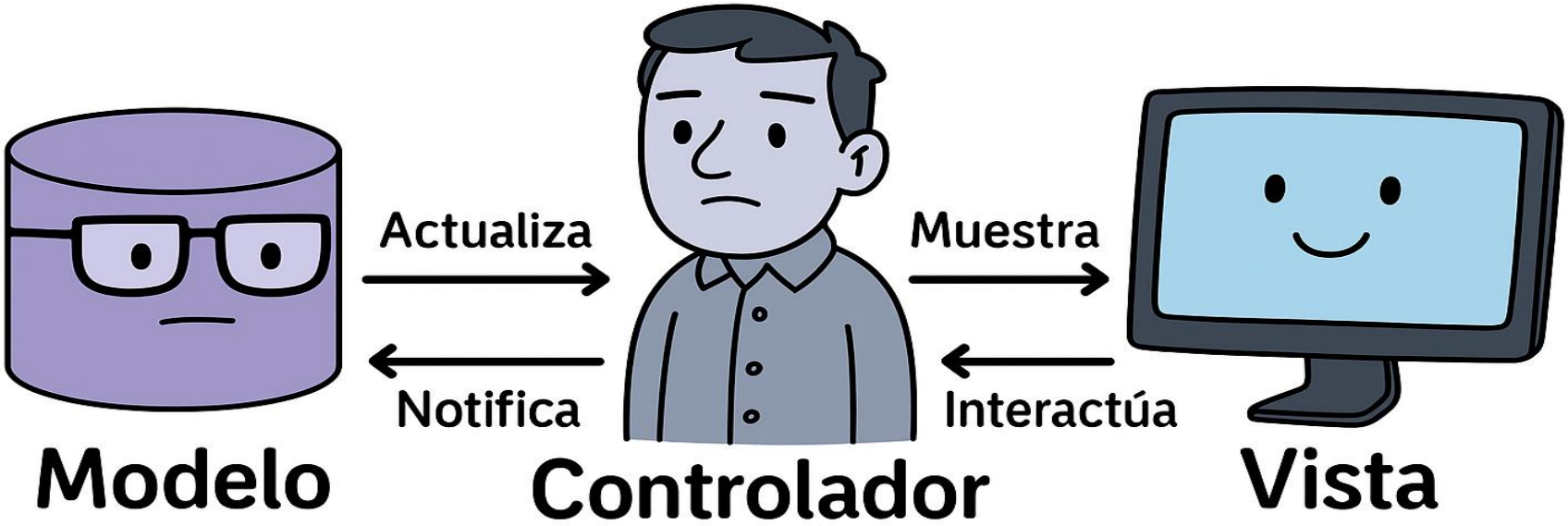
Patrón MVC

MVC significa **Modelo - Vista – Controlador (o Model – View – Controller).**

Es un patrón de arquitectura de diseño de software que se utiliza para organizar el código de una aplicación, separando la lógica de negocio, la interfaz de usuario y el control de flujo de la información.

MVC resuelve esto separando la aplicación en tres capas lógicas distintas:

- **Modelo (Model):** Representa los datos y la lógica de negocio de la aplicación.
- **Vista (View):** Es la interfaz de usuario, lo que el usuario ve y con lo que interactúa.
- **Controlador (Controller):** Actúa como un intermediario entre el Modelo y la Vista, manejando las interacciones del usuario y actualizando el Modelo y la Vista según sea necesario.


Actualiza
Muestra
Notifica
Interactúa
Modelo
Controlador
Vista

Modelo (Model)

El Modelo **es el corazón de la aplicación, en términos de datos y reglas de negocio**.

¿Qué contiene?

- **Datos:** Representa la información que la aplicación maneja (por ejemplo, una lista de usuarios, productos, publicaciones de blog). Esto incluye la estructura de esos datos (cómo se guardan, como en una base de datos) y cómo se acceden a ellos.
- **Lógica de Negocio:** Contiene las reglas y la lógica para manipular esos datos. Por ejemplo, si tienes un modelo de "Producto", el Modelo podría tener métodos para calcular el precio final con impuestos, validar si hay suficiente stock antes de vender, o guardar el producto en la base de datos.
- **Independencia de la UI:** El Modelo es completamente independiente de cómo se muestran los datos al usuario. No sabe nada de la interfaz de usuario.

Modelo (Model)

¿Cómo interactúa?

- Puede ser consultado por el **Controlador** para obtener datos.
- Puede ser actualizado por el **Controlador** con nueva información.
- Notifica a la **Vista** (o al **Controlador**, que a su vez actualiza la **Vista**) cuando sus datos cambian, para que la interfaz de usuario se actualice en consecuencia.

Ejemplo: En una aplicación de e-commerce, el **Modelo** sería responsable de manejar *productos, usuarios, pedidos*. Tendría métodos como *getProductos(), guardarUsuario(), calcularTotalPedido().*

Vista (View)

La **Vista** es la **capa de presentación**, lo que el usuario final ve y con lo que interactúa directamente.

¿Qué contiene?

- **Elementos visuales:** Botones, campos de texto, tablas, imágenes, etc.
- **Plantillas:** Código que define cómo se deben mostrar los datos recibidos del Modelo.

¿Qué hace?

- **Muestra los datos:** Recibe datos del Controlador (que a su vez los obtuvo del Modelo) y los presenta de una forma legible para el usuario.
- **Recoge interacciones del usuario:** Detecta eventos como clics de botón, entradas de texto, selección de opciones.

Vista (View)

¿Qué hace?

- **No contiene lógica de negocio:** La Vista no sabe cómo se procesan los datos ni por qué. Simplemente los muestra. Su única responsabilidad es la presentación.

¿Cómo interactúa?

- **El Controlador** le envía los datos que debe mostrar.
- Envía las interacciones del usuario al **Controlador**.

Ejemplo: En una aplicación de e-commerce, la **Vista** sería la *página de inicio*, la *página de detalles del producto*, el *carrito de compras*. Mostraría la información de los *productos, precios, descripciones*.

Controlador (Controller)

El **Controlador** actúa como **el cerebro de la aplicación**, coordinando las interacciones entre el Modelo y la Vista.

¿Qué contiene?

- **Lógica de control:** Reglas sobre cómo responder a las interacciones del usuario.

¿Qué hace?

- **Maneja las entradas del usuario:** Recibe eventos de la **Vista** (un clic en un botón, una entrada de formulario, etc.).

Controlador (Controller)

¿Qué hace?

- **Procesa la solicitud:** Decide qué hacer con esa entrada. Esto a menudo implica:
  - **Actualizar el Modelo:** Si la acción del usuario requiere un cambio en los datos (por ejemplo, guardar un nuevo usuario, añadir un producto al carrito), el **Controlador** le dice al **Modelo** que realice esa operación.
  - **Consultar el Modelo:** Si la acción requiere obtener datos (por ejemplo, mostrar una lista de productos), el **Controlador** le pide esos datos al **Modelo**.
- **Actualiza la Vista:** Después de que el **Modelo** ha sido actualizado o se han obtenido los datos, el **Controlador** decide qué **Vista** debe mostrarse y le pasa los datos necesarios para su presentación.

Controlador (Controller)

¿Cómo interactúa?

- Recibe eventos de la **Vista**.
- Envía comandos al **Modelo**.
- Recibe datos del **Modelo**.
- Envía datos a la **Vista**.

Ejemplo: En una aplicación de e-commerce, un **Controlador** manejaría una solicitud de "*añadir producto al carrito*". Recibiría la información del producto de la **Vista**, le indicaría al **Modelo** que agregue el producto al carrito, y luego le diría a la **Vista** del carrito que se actualice para mostrar el nuevo artículo.

Ventajas del modelo MVC

- **Separación de preocupaciones:** Cada componente tiene una responsabilidad clara, lo que facilita el desarrollo y el mantenimiento.
- **Reusabilidad del código:** Los Modelos y Controladores pueden ser reutilizados en diferentes Vistas (por ejemplo, los mismos datos y lógica de negocio para una aplicación web y una aplicación móvil).
- **Desarrollo en paralelo:** Diferentes equipos pueden trabajar en el Modelo, la Vista y el Controlador simultáneamente.
- **Mayor facilidad para pruebas:** Al estar los componentes separados, es más fácil probar cada parte de forma aislada.
- **Escalabilidad:** Las aplicaciones bien estructuradas con MVC son más fáciles de escalar y adaptar a nuevas funcionalidades.