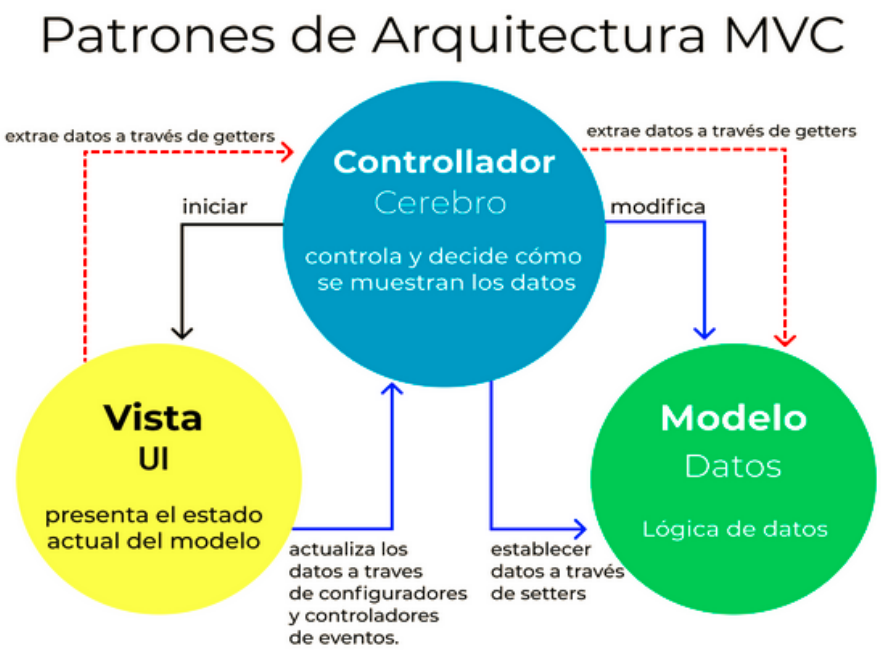
El Modelo-Vista-Controlador (MVC): Una Estructura Modular para el Desarrollo de Software

El patrón Modelo-Vista-Controlador (MVC) es un patrón arquitectónico que se utiliza comúnmente en el diseño de aplicaciones, especialmente en el desarrollo de software orientado a objetos. Esta estructura, reconocida por su capacidad para separar preocupaciones y mejorar la modularidad, ha sido ampliamente adoptada en el desarrollo de aplicaciones. Este patrón divide una aplicación en tres componentes principales: el Modelo, la Vista y el Controlador



Modelo: *Gestionando la Lógica y los Datos*

En el corazón del patrón MVC se encuentra el Modelo, que asume la responsabilidad de gestionar la lógica de la aplicación y los datos asociados. Este componente es crucial para la integridad y la consistencia de la aplicación, ya que se encarga de las operaciones de manipulación de datos y de comunicar cualquier cambio a la Vista.

Responsabilidades

1. Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
2. Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
3. Lleva un registro de las vistas y controladores del sistema.

4. Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).

Vista: *La Interfaz de Usuario Intuitiva*

La Vista se encarga de la presentación de la información al usuario. Este componente refleja visualmente los datos proporcionados por el Modelo y, al mismo tiempo, captura las interacciones del usuario. La separación clara entre la Vista y el Modelo permite una fácil modificación de la interfaz de usuario sin afectar la lógica de la aplicación.

Responsabilidades

1. Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
2. Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

Controlador: *La Unión entre Vista y Modelo*

El Controlador actúa como un intermediario vital entre la Vista y el Modelo. Este componente recibe las interacciones del usuario desde la Vista, procesa la lógica de la aplicación y actualiza el Modelo en consecuencia. Además, el Controlador recibe notificaciones del Modelo y actualiza la Vista para reflejar cualquier cambio.

Responsabilidades

1. Recibir datos del modelo y los muestra al usuario.
2. Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
3. Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Beneficios de MVC: *Separación de Preocupaciones y Mantenibilidad*

El uso del patrón MVC ofrece una serie de beneficios. La clara separación de preocupaciones permite a los desarrolladores trabajar en módulos específicos sin afectar otros componentes. Esta modularidad mejora la mantenibilidad del código a lo largo del tiempo, facilitando actualizaciones y modificaciones.

Adaptabilidad y Reutilización de Código

La estructura modular del MVC facilita la adaptabilidad de las aplicaciones a medida que los requisitos evolucionan. La capacidad de reutilizar componentes individuales, como modelos y controladores, en diferentes partes de una aplicación o incluso en proyectos nuevos, es un factor clave que contribuye a la eficiencia del desarrollo.