Introducción al patrón de diseño MVC

Ing. Yakelin Quispe Valero

¿Qué es MVC?

- **Definición de MVC**
  MVC (Modelo-Vista-Controlador) es un patrón arquitectónico que divide una aplicación en tres componentes principales:
- **Modelo**: Representa los datos y la lógica de negocio.
- **Vista**: Representa la interfaz de usuario (UI).
- **Controlador**: Gestiona la interacción entre el modelo y la vista.

Propósito de MVC

- **¿Por qué usar MVC?**
- **Separación de preocupaciones**: Facilita el mantenimiento y desarrollo.
- **Mejora la escalabilidad**: Permite actualizar una parte sin afectar las demás.
- **Facilita pruebas**: Se puede probar cada componente por separado.

Componentes del patrón MVC

- **Modelo**:
  - Representa los datos.
  - Lógica de negocio y acceso a datos (base de datos, API).
- **Vista**:
  - Interfaz de usuario.
  - Representa la presentación de la información.
- **Controlador**:
  - Intermediario entre modelo y vista.
  - Maneja las interacciones del usuario.

Diagrama de flujo MVC

- **Diagrama básico de flujo MVC:**

1. El **usuario** interactúa con la **Vista**.
2. La **Vista** envía las acciones al **Controlador**.
3. El **Controlador** actualiza el **Modelo** si es necesario.
4. El **Modelo** actualiza los datos y notifica al **Controlador**.
5. El **Controlador** actualiza la **Vista** para reflejar los cambios.

Ejemplo práctico

- **Ejemplo en una aplicación web de blog**:
- **Modelo**: Base de datos de entradas de blog, comentarios.
- **Vista**: Página web con artículos y formularios de comentarios.
- **Controlador**: Gestiona la creación de nuevas entradas, la publicación de comentarios y la actualización de la vista.

Ventajas del patrón MVC

- **Facilidad de mantenimiento**: Los cambios en la interfaz no afectan la lógica de negocio.
- **Reusabilidad**: Componentes del modelo y la vista pueden ser reutilizados.
- **Escalabilidad**: Fácil de extender con nuevas funcionalidades.
- **Desarrollo paralelo**: Los desarrolladores pueden trabajar en diferentes componentes al mismo tiempo.

Desventajas del patrón MVC

- **Complejidad inicial**: Puede resultar difícil de implementar al principio.
- **Sobreabundancia de clases**: Se requiere crear múltiples clases y objetos.
- **Curva de aprendizaje**: Puede resultar difícil para principiantes en programación.

Variantes de MVC

- **MVVM (Modelo-Vista-ViewModel)**: Popular en aplicaciones de escritorio y móviles.
- **MVP (Modelo-Vista-Presentador)**: Utilizado en algunas aplicaciones web.
- **Flux / Redux**: Variantes de arquitecturas unidireccionales (común en aplicaciones frontend).

Conclusión

- El patrón **MVC** es fundamental para aplicaciones grandes y complejas.
- Se debe considerar la estructura del proyecto desde el principio.
- **Sigue aprendiendo** sobre patrones de diseño como **MVVM** y **MVP** para diferentes tipos de aplicaciones.