MVC

El famoso **Modelo-Vista-Controlador (MVC)** es un **patrón de arquitectura de software** que se usa para separar la **l**ógica de negocio, la interfaz de usuario y el control de flujo de una aplicación. Es muy utilizado en el desarrollo de software, especialmente en aplicaciones web y de escritorio.

Se caracteriza por estar formado por tres capas separadas:

- **Modelo**: Esta capa representa los **datos** y la **lógica de negocio**. Aquí es donde se gestiona cómo se accede, almacena y manipula la información de la aplicación, normalmente, una Base de Datos.

  Pero, esta capa no solamente interactúa con la BBDD. También gestiona otras fuentes de datos como, por ejemplo, conexiones con sistemas externos a nuestro sistema. También es responsabilidad preparar la información para ser utilizada por las demás capas.

- **Controlador**: Hace de intermediario entre la **Vista** y el **Modelo**. Recibe las acciones del usuario desde la Vista (clics, entradas de texto, etc.), las interpreta y actualiza el Modelo o la Vista según corresponda.

  Se puede decir que el controlador es el que 'decide' lo que se tiene que hacer en cada momento. Por ejemplo, convierte la pulsación de un botón en un acceso a BBDD, una inserción de un producto en un carrito, y en mostrar la información correspondiente en la vista.

- **Vista**: Es la **interfaz de usuario**, lo que el usuario ve y con lo que interactúa. Esta capa muestra y solicita información, pero nunca es la responsable de modificarla.

El objetivo final es separar responsabilidades para que:

- El código sea **más fácil de mantener y probar**.
- Se pueda **cambiar la interfaz** sin tocar la lógica.
- Se pueda **reutilizar el Modelo** con diferentes Vistas.

MCV en un proyecto real

MVC suele combinarse con otros patrones de diseño, no reemplaza otros patrones, sino que define la estructura general. Dentro de cada capa (Modelo, Vista o Controlador) puedes aplicar otros patrones más específicos.

Por ejemplo:

- Modelo:
  - Patrón **Data Access Object**, que separa los datos y la lógica de negocio de acceso a BBDD.
  - **Factory**, **Factory Method**
  - **Singleton**

- Controlador:
  - Strategy, para cambiar la lógica de control según el contexto

- Vista:
  - Observer, para actualizar automáticamente la vista cuando se cambian los datos del Modelo

Por tanto, MCV sirve como base para diseñar aplicaciones muy grandes, que van a estar funcionando durante muchos años, y van a sufrir un gran número de modificaciones a lo largo del tiempo. Si la aplicación está bien preparada, entonces será posible realizar cambios increíblemente complejos de forma razonablemente sencilla como, por ejemplo, cambiar por completo la interfaz del usuario sin tener que modificar el resto de la aplicación.

Spring Boot por ejemplo utiliza MCV, al igual que muchos otros frameworks.

Práctica - 1

Importa el proyecto **AgendaVFireBase** en tu IDE. Tendrás que generar una colección en FireBase y preparar el proyecto para que la utilice. Observa que el proyecto sigue el patrón MCV.

Siguiendo dicho patrón MCV, completa el ejercicio para todas las opciones de los botones.

¿Te has dado cuenta de que seguir el patrón te facilita mucho ampliar las funcionalidades del proyecto?