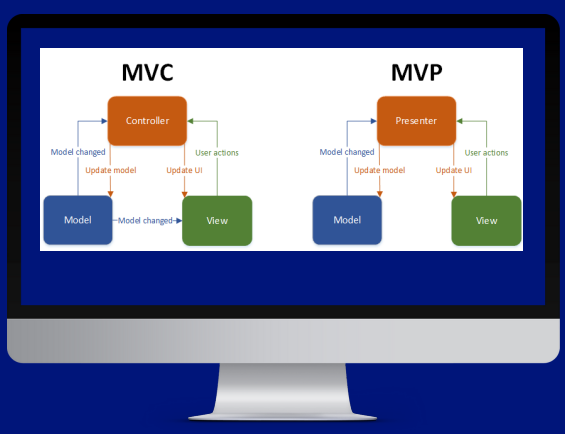
PATRONES MVVM Y MVC

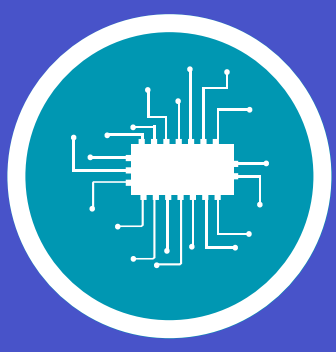


Guía paso a paso

1.

EL PATRÓN MVVM

MVVM (Model-View-ViewModel) Es un patrón de arquitectura de software. Se caracteriza por tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación.
Es muy popular en el desarrollo de aplicaciones modernas.



2.

EJEMPLOS DE APLICACIONES QUE UTILIZAN ESTE PATRÓN

- Microsoft WPF Applications - El patrón MVVM se utiliza comúnmente en el marco de la plataforma Windows Presentation Foundation (WPF) de Microsoft para construir aplicaciones de escritorio.
- Xamarin - MVVM es la arquitectura recomendada para la creación de aplicaciones móviles multiplataforma con Xamarin.

- AngularJS - El marco de trabajo de AngularJS utiliza el patrón MVVM en sus aplicaciones, basado en el enlace de datos bidireccional, la separación de preocupaciones y la modularidad.
- React Native - MVVM se ha convertido en una arquitectura popular en la creación de aplicaciones móviles en React Native.

3.

4.

MVC

- **El punto de entrada a la aplicación es el controlador.**
- **La vista no tiene referencia del Controlador.**
- **Presenta un acoplamiento entre los componentes muy bajo.**

MVVM

- **El punto de entrada se realiza desde la vista.**
- **La vista no tiene referencia del ViewModel.**
- **Los componentes están totalmente desacoplados.**

5.

PRINCIPALES COMANDOS Y PATRONES A UTILIZAR EN EL PATRÓN MVVM:

- **Comando Bind: Nos permite establecer una conexión de datos bidireccional entre la interfaz de usuario (View) y la lógica de negocio (ViewModel). Es decir, los cambios realizados en la vista se reflejarán automáticamente en el ViewModel y viceversa.**
- **Comando RelayCommand: se utiliza para definir una acción que se ejecutará cuando el usuario interactúe con un elemento de la interfaz de usuario, como un botón o un menú desplegable. El RelayCommand se vincula a un método del ViewModel que realiza la acción deseada.**

- **Comando INotifyPropertyChanged:** lo utilizamos para notificar al ViewModel cuando se producen cambios en las propiedades de la vista. El ViewModel puede entonces actualizar los datos según sea necesario y actualizar la vista en consecuencia.
- **comando Observer:** Este patrón se utiliza para mantener la sincronización entre la vista y el ViewModel. La vista actúa como un observador y el ViewModel actúa como un objeto observable. Cuando el ViewModel cambia de estado, notifica a la vista, que actualiza su contenido en consecuencia.
- **comando Singleton:** Este patrón se utiliza para garantizar que solo haya una instancia de una clase ViewModel en todo momento. Esto es útil cuando se desea compartir datos entre diferentes vistas en una aplicación.

BIBLIOGRAFIA

- https://www.abatic.es/arquitecturas-de-software-mvc-y-mvvm/
- https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93modelo_de_vista
- https://blog.ida.cl/diseno/que-son-los-patrones-de-diseno-en-interfaz-de-usuario/
- https://learn.microsoft.com/es-es/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

ALUMNOS

- Rodrigo Alejandro Estacuy Martinez EM152384
- Juan Pablo Salvador Garcia SG232051

ALGUNAS DE LAS PRINCIPALES CARACTERÍSTICAS DEL PATRÓN MVVM INCLUYEN:

- **Separación de responsabilidades:** El patrón MVVM permite separar claramente la responsabilidad de la interfaz de usuario, la lógica de negocio y la lógica de presentación. Esto hace que sea más fácil de mantener y actualizar la aplicación en el futuro.
- **Enlace de datos:** El patrón MVVM utiliza enlace de datos bidireccional, lo que significa que los cambios realizados en la vista se reflejarán automáticamente en el ViewModel y viceversa. Esto simplifica el proceso de actualización de la interfaz de usuario en respuesta a los cambios en los datos de la aplicación.
- **Pruebas unitarias:** Al separar claramente la lógica de negocio y la lógica de presentación, el patrón MVVM hace que sea más fácil escribir pruebas unitarias para la aplicación. Las pruebas unitarias pueden centrarse en la lógica de negocio y la lógica de presentación por separado, lo que aumenta la calidad y la fiabilidad de la aplicación.
- **Flexibilidad:** El patrón MVVM permite una gran flexibilidad en el desarrollo de la interfaz de usuario. Los diseñadores pueden trabajar en la interfaz de usuario sin preocuparse por la lógica de negocio o la lógica de presentación, y los desarrolladores pueden trabajar en la lógica de negocio y la lógica de presentación sin preocuparse por la interfaz de usuario.
- **Mantenimiento:** El patrón MVVM hace que sea más fácil mantener y actualizar la aplicación en el futuro. Como cada componente de la aplicación está claramente definido y separado, los cambios en uno de ellos no afectarán directamente a los otros componentes, lo que reduce el riesgo de errores y conflictos.