**MVVM**

**(Modelo-View-ViweModel)**

El patrón modelo–vista–modelo de vista, es un patrón de [arquitectura de software](https://es.wikipedia.org/wiki/Arquitectura_de_software). Se caracteriza por tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación. Proporciona una separación entre la interfaz gráfica y el desarrollo de la lógica de negocio. El ViewModel del patrón MVVM es el responsable de exponer los objetos del modelo que podrá consumir la vista de manera que estos objetos serán fácilmente manejables y consumidos.

MVVM usa las funciones de DataBinding que nos proporciona XAML, para separar estas dos capas consiguiendo eliminar el code-behind de las vistas y facilitar el testing del código de los eventos que pude lanzar una vista, ya que también se encarga de gestionarlos. Por tanto, el ViewModel se puede entender como un contenedor de datos y comandos que no sabe de dónde vienen los datos (aspecto del que se encargará el modelo) y cómo estos serán mostrados (aspecto del que se encargará la vista).

Los elementos que tiene:

El modelo: Representa la capa de datos y/o la lógica de negocio, también denominado como el objeto del dominio. Contiene la información, pero nunca las acciones o servicios que la manipulan. Jamás depende de la vista.

La vista: Representa la información a través de los elementos visuales que la componen. La vista en MVVM son activas, contienen comportamientos, eventos y enlaces a datos que necesitan conocimiento del modelo subyacente.

Modelo de Vista: Es un intermediario entre el modelo y la vista, contiene toda la lógica de presentación y se comporta como una atracción de la interfaz. La comunicación entre la vista y el modelo de vista se realiza por medio de los enlaces de datos.

Beneficios:

- Este patrón permite tener two-way data binding entre el View y el ViewModel.

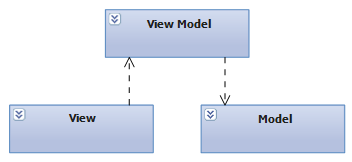
- El usuario sólo va a interactuar con el View.

- El View va a tener relación con el ViewModel.

- El ViewModel no obtiene información sobre el View.

- Al separar el código de la UI, pueden hacerse pruebas automatizadas sobre el resto del código

- Se pueden aislar los cambios y, así, minimizar el impacto de los cambios.



Entra en la categoría de patrones de diseño estructural.