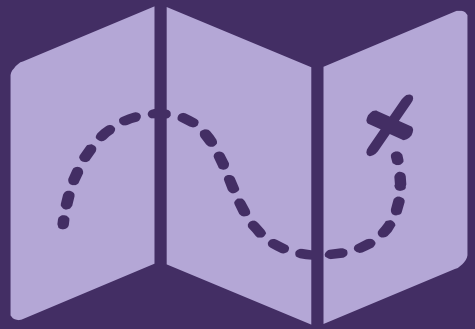
Modelo Vista Controlador (MVC)

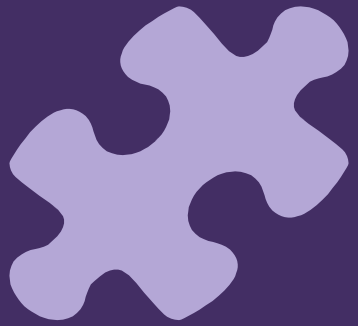
Arquitectura de software

Forma de estructurar un sistema que sirve como guía para que todos los desarrolladores de software trabajen bajo una misma guía de construcción.



Patrón

Modelo o tipo de tema que sirve para elaborar algo igual. Es de vital importancia en caso de haber objetos o sucesos recurrentes.



Patrón de diseño

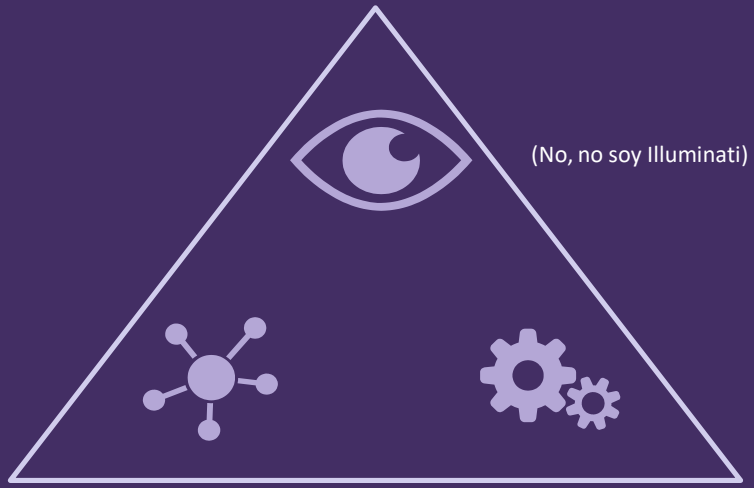
Solución reutilizable a problemas recurrentes que nos encontramos en el diseño de software.

Modelo Vista Controlador (MVC)

Patrón de arquitectura de software que separa los datos, la interfaz de usuario y la lógica de control en una aplicación.

Filosofía de MVC

“Debes dividir tu software en 3 capas distintas y separadas, donde cada capa tiene una única responsabilidad”



(No, no soy Illuminati)

Modelo Vista Controlador (MVC)

Modelo

(Persistencia)

Representación de los datos que administra el sistema, su lógica y sus mecanismos de persistencia.

Vista

(Presentación)

Interfaz de usuario, que compone la información que se envía al cliente y los mecanismos que interactúan con éste, al momento de presentar y obtener.

Controlador

(Reglas de negocio)

Actúa como intermediario entre el modelo y la vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

¿Cómo funcionan los componentes?

Modelo

- **Acceder a la capa de almacenamiento de datos**, donde actualiza, consulta y busca.
- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

Vista

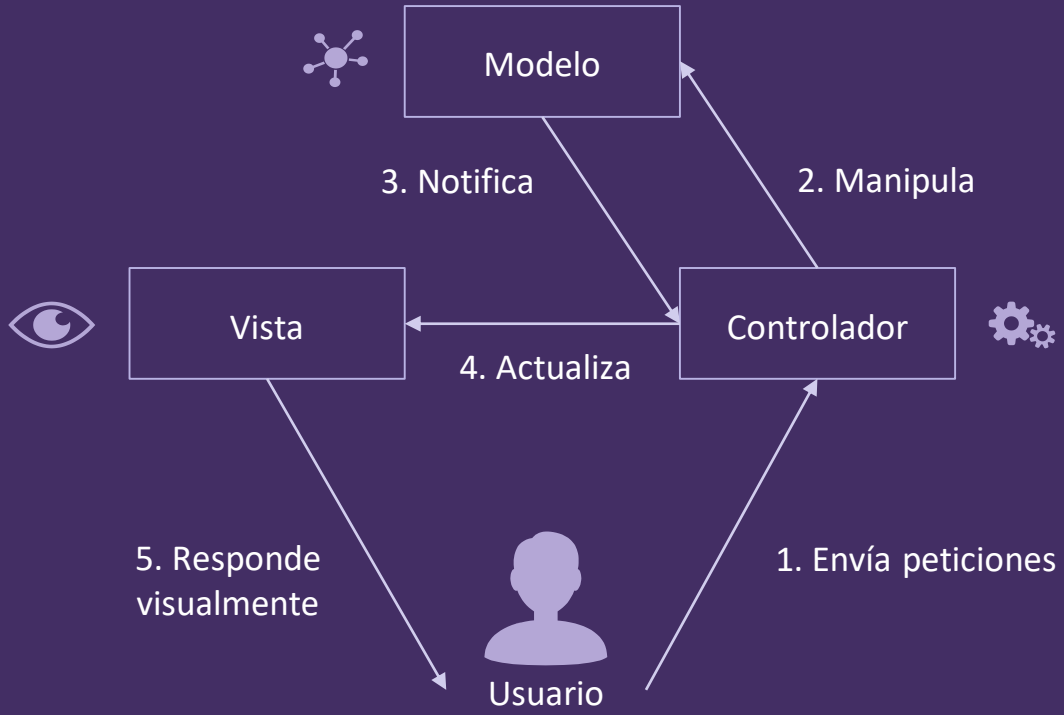
- Ni el modelo ni el controlador **se preocupan de cómo se verán los datos**, esa responsabilidad es únicamente de la vista.
- Recibe datos del modelo y los muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).

Controlador

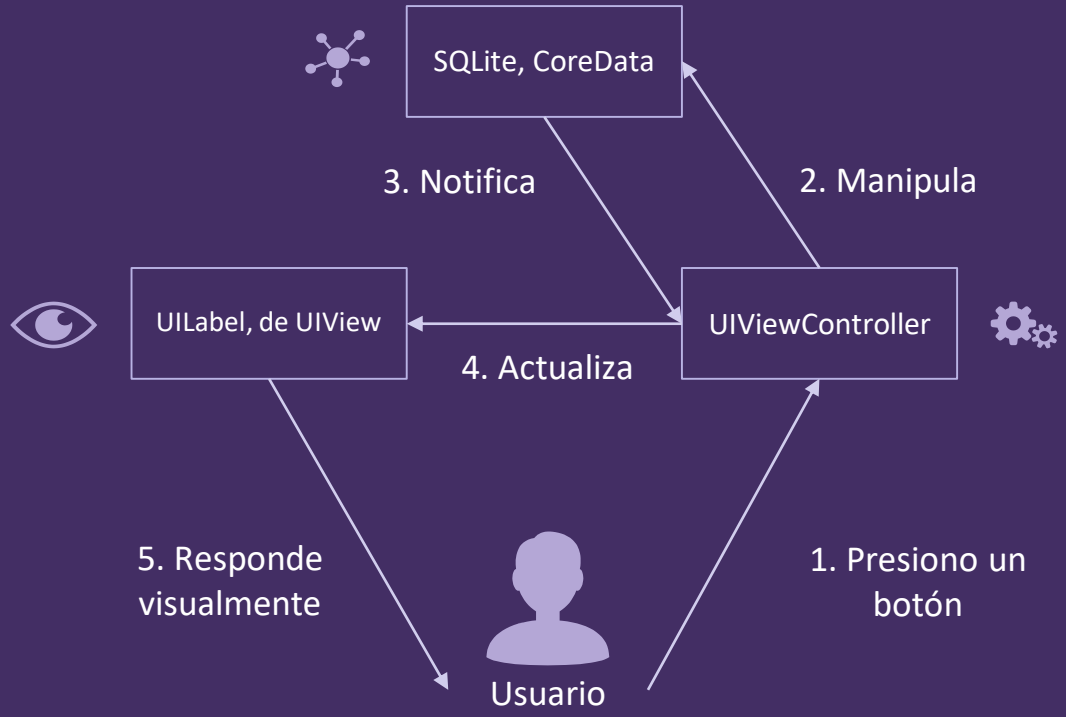
- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- **Sincroniza ambas capas**.

¿Para qué utilizar MVC?

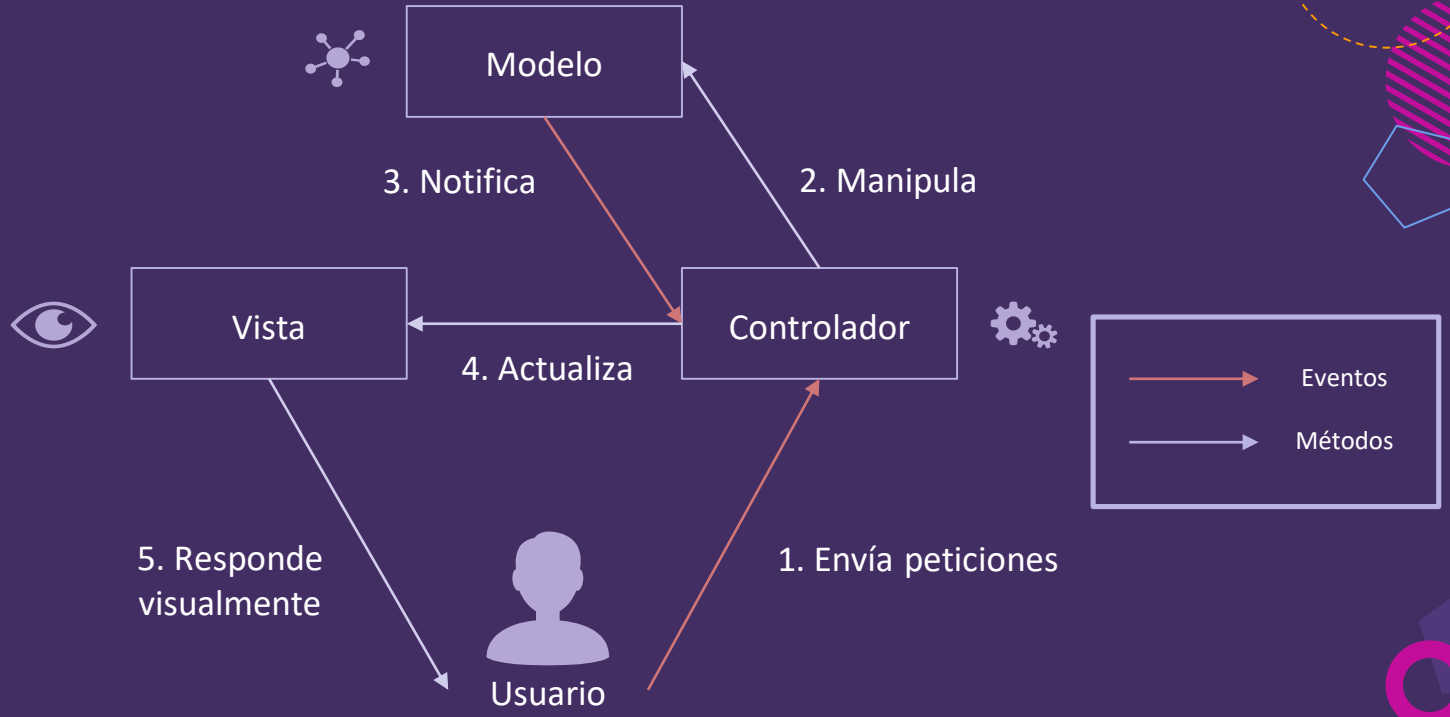
- Patrón maduro
- Puede ser utilizado en diferentes lenguajes y plataformas
- Divide en 3 componentes diferentes e independientes entre sí


Modelo
3. Notifica
2. Manipula
Vista
Controlador
4. Actualiza
5. Responde visualmente
1. Envía peticiones
Usuario

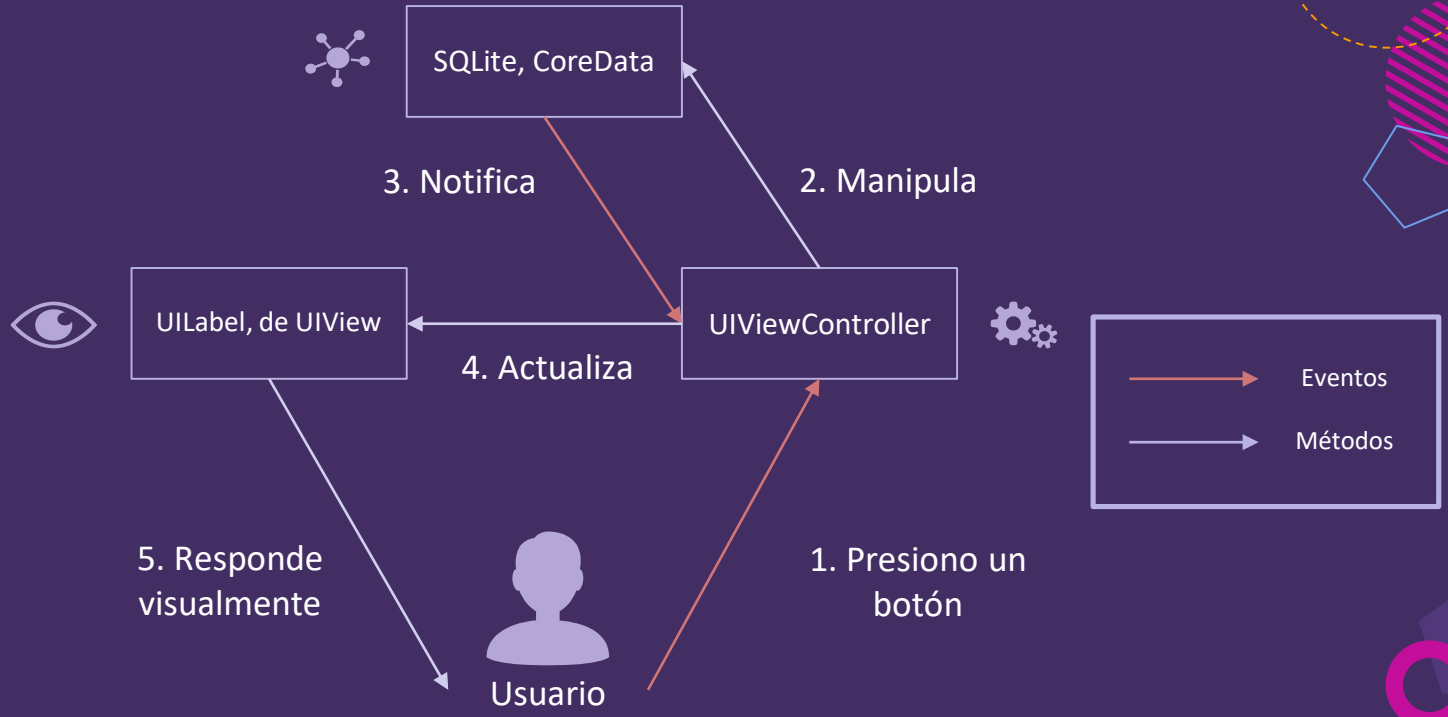
MVC en Swift


SQLite, CoreData
3. Notifica
2. Manipula
UILabel, de UIView
UIViewController
4. Actualiza
5. Responde visualmente
Usuario
1. Presiono un botón

Subamos un poco la dificultad



MVC en Swift


SQLite, CoreData
3. Notifica
2. Manipula
UILabel, de UIView
UIViewController
4. Actualiza
Eventos
Métodos
5. Responde visualmente
Usuario
1. Presiono un botón

Referencias

- Cervantes, H. (2010). *Arquitectura de Software.* Software Guru. Disponible en: https://sg.com.mx/revista/27/arquitectura-software
- Ecured. (2018). *Arquitectura de Software.* Disponible en: https://www.ecured.cu/Arquitectura_de_software
- Hernández, U. (2015). MVC (Model, View, Controller) Explicado. CodigoFacilito. Disponible en: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado
- R, U. (2017). *Patrón de Diseño Model View Controller: MVC*. Efecto Apple. Disponible en: https://www.efectoapple.com/patron-diseno-model-view-controller-mvc/
- Universidad de Alicante. (2018). *Modelo Vista Controlador*. Disponible en: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html