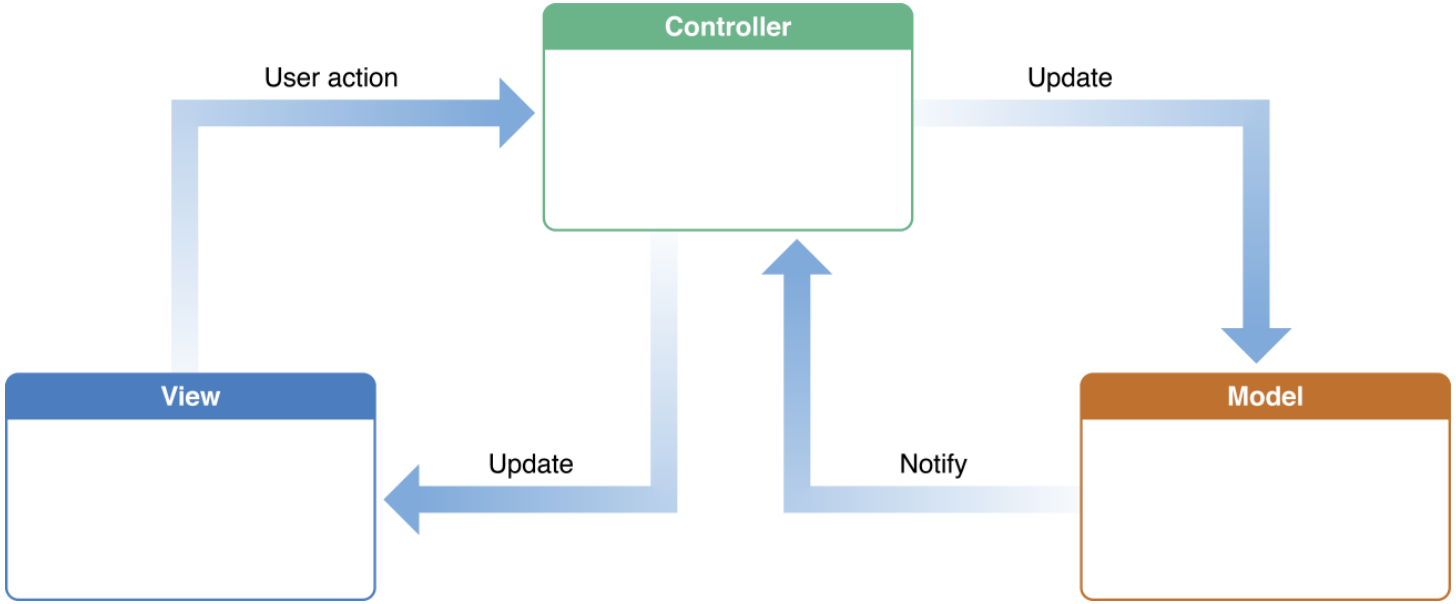
Patrones de diseño: MVC

KEEPCODING
Tech School

Model - View - Controller (MVC)



Model

- Encapsula los datos de la aplicación
- Los objetos modelo no tienen conexión con los objetos de la View que presentan los datos.
- Modelos de datos DTO (Data Transfer Objects).
- Gestionar el almacenamiento y recuperación de datos del dominio.
- Contiene la representación del dominio, lógica de negocio y la persistencia de datos

View

- Cualquier objeto de la aplicación que el usuario pueda ver.
- La interfaz de usuario UI.
- Un objeto View debe saber mostrarse al usuario y responder a acciones.
- Solo realiza funciones relacionadas con la interfaz de usuario.
- No implementa la lógica de negocio.
- No tiene contacto directo con la capa Model.

Controller

- Actúa de intermediario entre la capa de View y la de Model.
- Recibe los cambios del modelo y de la vista.
- Realiza tareas de configuración y coordinación entre los datos y las vistas.
- Se podría decir que es el cerebro del proyecto, decide lo que sucederá en cada momento.

Ventajas

El uso del patrón MVC ofrece ventajas sobre otras maneras de desarrollar aplicaciones con interfaz de usuario.

- Separación de responsabilidades impuesta por el uso del patrón MVC.
- Los componentes de las aplicaciones tienen sus misiones bien definidas.
- Proyectos más limpios, simples, más fácilmente mantenibles y más robustos.
- Mayor velocidad de desarrollo en equipo.
- Reutilizar los desarrollos y asegurando consistencia entre ellos.
- Facilidad para realización de pruebas unitarias.

Desventajas

No todo es perfecto, su uso presenta **también algunas desventajas**:

- La división impuesta por el patrón MVC obliga a mantener un mayor número de archivos, incluso para tareas simples.
- Curva de aprendizaje. Dependiendo del punto de partida, el salto a MVC puede resultar un cambio radical y su adopción requerirá cierto esfuerzo.
- Peligro de generar clases con gran número de líneas de código difíciles de mantener.

MVC

Massive View Controller: Peligros

- Se trata de comenzar a incrementar las funcionalidades de un ViewController acumulando responsabilidad en el mismo.
- Los ViewController pasan de tener menos de 300 líneas de código a más de 1.000.
- Funcionalidades imposibles de mantener, de reutilizar y de testear.

MVC

MVC

Massive View Controller: Consejos para evitarlo

- El ViewController solo será responsable de responder a las acciones que el usuario realiza en la View.
- Los ViewController **NO** deben ser responsables de actualizar el modelo de datos.
- No se deberían incluir UITableViewDataSource o UITableViewDelegate en el código del ViewController
- No realizar operaciones de comunicación con el API directamente.
- Nunca añadir métodos para parsear modelos de datos.

KEEPCODING

Tech School

Madrid | Barcelona | Bogotá

Alberto García Muñoz