MVC

Model-View-Controller

모든 입력에 대한 부분은 Cotroller에서 처리하고 Model(데이터갱신)을 조작하고, Controller는 다시 조작된 데이터를 View를 선택하여 View가 보여줍니다.

그래서 Model은 데이터의 저장이나갱신, View는 처리된 데이터를 사용자에게 직접적으로 입력받고,보여주는,Controller는 View와 Model간에 처리나 제어를 담당한다고 볼 수 있습니다.

MVP패턴
Model-View-Presenter

MVP패턴은 1990년대 초기에 등장한 패턴으로 마이크로소트가 사용해서 알려진 패턴으로 MVC패턴을 기반을 둔 GUI를 만들기 위한 설계패턴입니다.

View : 실제적으로 보여줄 UI컴포넌트를 보여주는 역할을 합니다.

Presenter: View 와 Model간의 제어를 담당하며 ,View interface를 통해서 View 처리된부분을 보여줍니다. 또한 View에서 사용자가 동작한 것을 전달받아서 실제적으로 Model에서 데이터를 갱신하는 역할을 합니다.

View Interface : Presenter에서 View를 직접 참조하지 않고 View Interface를 참조해서 View와 의존성 을 감소시키고, View의 실제 UI 요소가 어떻게 구현되는지 몰라도 데이터를 올바르게 표현할 수 있도록 해줍니다.

이렇게 독립적으로 진행해서 View와 Model의 의존관계를 낮출 수 있습니다.

MVVM

Model-View-ViewModel

MVVM또한 MVC개념으로부터 탄생했다고 볼 수 있습니다.

Controller 에서 ViewModel로 바뀌었는데 View들어간것처럼
뷰를 위한 모델입니다. 그래서 뷰에 대한 이벤트처리와 Controller의 역할도 수행하게 되어서 점점 코드가 집중되는 구조가 될 수도 있습니다.

IOS에서 비슷한 패턴은 MVP패턴인 것 같습니다.

기본적으로 Model -View-Controller 지만 뷰에서 전달받은 사용자이벤트에 대한처리를 delegate를 통해서 Controller에서 해당 이벤트가 Controller에서 지정해줘야하는 것이 있다면 위임받아 처리해줘야합니다. MVP에서는 delegate를 ViewInterface가 역할을 대신하는 것 같습니다.