https://github.com/LeeHongHwa/LeeHongHwa/blob/homework/homework/161019_ModifyLoginCode.md

MVC, MVP, MVVM

과거의 소프트웨어 개발 과정에서 발견된 설계의 노하우에 이름을 붙여 이후에 재사용하기 좋은 형태로 묶어서 정리한 것
유지 보수와 재사용하기에 좋다

MVC
Model: Data를 관리한다
View: 사용자에게 보이는 UI
Controller: 사용자가 입력에 따른 요구 사항을 Model에서 처리하고 View와 Model을 연결해줌

MVP
Model: Data를 관리한다
View: 사용자에게 보이는 UI, 사용자의 입력을 받아 Presenter에게 전달한다.
Presenter: View에게서 받은 사용자의 요구를 Model에게 전달 후 Model이 전달한 값을 View에게 전달한다.

MVVM
Model: Data를 관리한다
View: 사용자에게 보이는 UI, 사용자의 입력을 받아 ViewModel에게 전달한다.
ViewModel : View에게 받은 사용자의 입력에 따른 요구 사항을 Model에게 전달하고 Model이 수행한 값을 ViewModel이 받아 View의 속성을 변경시켜준다.
View와 ViewModel이 연결되어 있는 것 같은데 바인딩이라는 방식을 사용해서 View의 속성을 바꾸는 거라 연결이 안 돼있다고 하는데 잘 이해가 안 된다.

https://github.com/LeeHongHwa/LeeHongHwa/blob/homework/homework/161019_ModifyLoginCode.md

각 디자인 패턴의 장단점

MVC
장점: 데이터와 사용자의 View를 완전한 분리는 아니지만 어느 정도 분리할 수 있다.
단점: Model과 View의 완벽한 분리가 어렵다. 그렇기 때문에 속도가 낮을 수 있다 또한 재사용이 어려울 수 있다.

MVP
장점: Model과 View가 완벽하게 분리되어 있다. 유지/보수의 편리성을 가지고 있다.
단점: Presenter에 너무 의존하게 된다.

MVVM
장점: Model, View, ViewModel 이 완전하게 분리되어 있다.
단점: 구현하는 데에 있어서 복잡함이 있다.

나의선택
View에 대해서 의존성이 많아지는 어플을 만든다면 MVVM이 더 좋을 것 같지만 지금 딱히 그 정도로 View에 의존을 해서 무리가 간다는 느낌을 받지 않아 MVP라는 방식을 사용하는 것이 좋을 것 같습니다.

실제로 애플이 사용하고 있는 MVC 패턴은 어떤 패턴에 가장 가까울까?
제가 이해한 대로 보면 애플에서 사용하고 있는 MVC는 tableView를 만들때도 Data랑 Delegate를 구분하는 해서 ViewController가 수행해준다는 느낌이 들어서 MVP에 가까운것 같습니다.
또한 View에서 입력을 받는 면에서 MVP를 따라한 MVC가 아닐까 싶습니다.