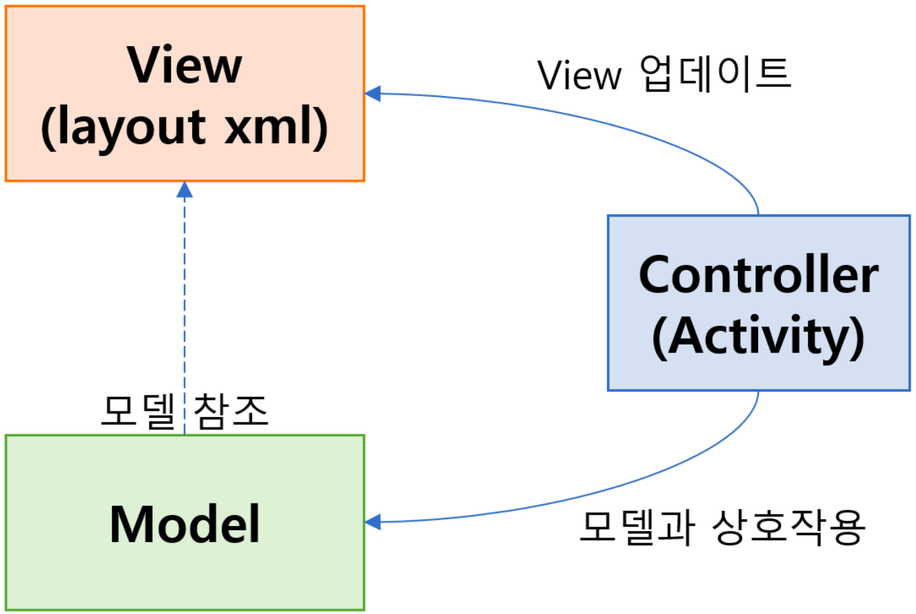
플러터 다트 아키텍처 패턴

3. MVVM 패턴

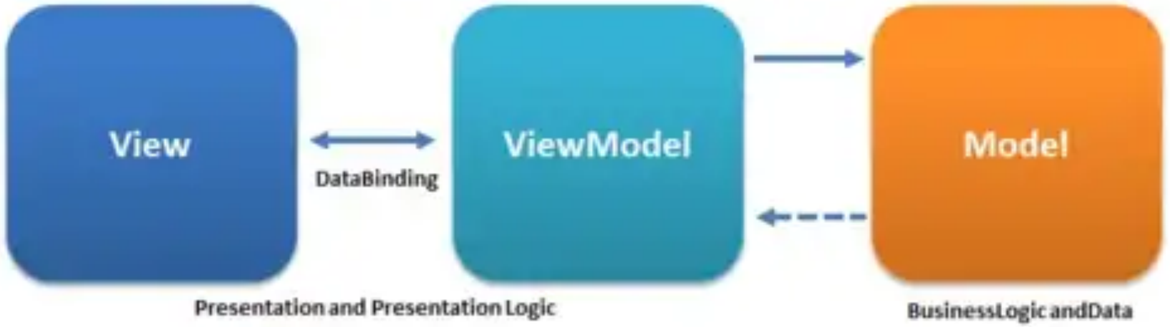
MVVM 패턴이란?

- MVC, MVP 패턴의 문제점을 개선하기 위해 나온 패턴
- Model은 데이터와 비즈니스 로직을 담당하며
- View는 사용자에게 보여지는 UI
- View Model은 Model과 View의 메신저 같은 역할을 함
- 단순히 요소들의 역할을 볼 때는 MVC, MVP와 큰 차이가 없어보임

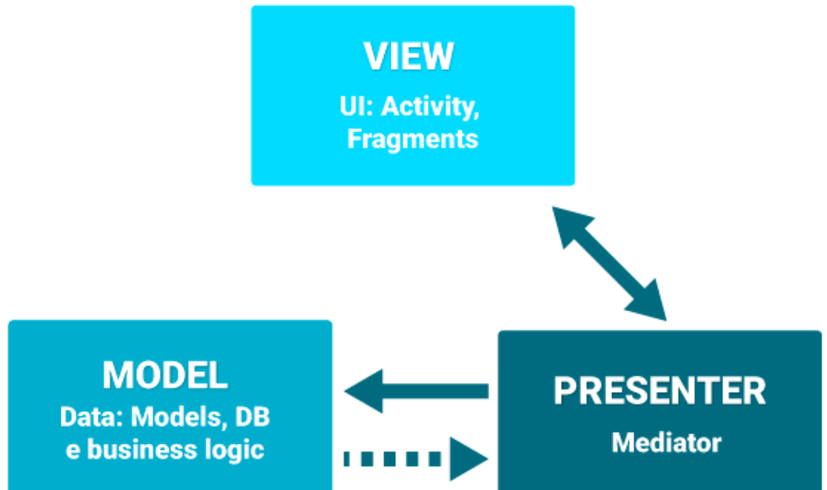
참조: 깃뚜님의 블로그
https://gitsu.tistory.com/38



참조: 미디엄 How To Use MVVM in Flutter - Aseem Wangoo
https://betterprogramming.pub/how-to-use-mvvm-in-flutter-4b28b63da2ca



참조: 봄석님의 블로그
https://beomseok95.tistory.com/212



기존 패턴과의 차이점(장점)과 단점

- 모델-뷰가 직접적으로 참조가 되기 때문에 필연적으로 종속성이 높아지는 MVC와 달리 MVVM은 뷰와 모델은 서로 의존성이 없음
- 뷰와 프리젠터가 1:1로 존재해야하는 MVP는 보일러플레이트 코드의 양이 많지만 MVVM의 경우 뷰 모델이 여러 곳의 뷰에서 쓰일 수 있기 때문에 불필요한 코드가 상대적으로 적음

- 데이터 바인딩, 커맨드 패턴 등이 기본이 되기 때문에 러닝커브가 높음
- 메모리 소모 이슈(데이터 바인딩)

플러터와 MVVM 패턴

- 실제로 플러터 기본 및 상태관리 관련 자료에 가장 자주 등장하는 패턴인 만큼 플러터로 구현하기가 MVC나 MVP보다 훨씬 수월함

- 데이터바인딩이란 쉽게 말해 눈으로 보이는 데이터와 현재 데이터 상태를 하나로 묶는, 즉 동기화를 시키는 것이라고 할 수 있는데 이것이 플러터의 상태관리와 겹치는 부분이 있음. (레이아웃을 잡고 그것을 코드로 접근해서 손수 바꿔야하는 기존 방식과 달리 레이아웃에 직접 어떤 값이 들어갈 것인지 선언적으로 UI를 작성하는 방식)

- 커맨드 패턴은 주체 혹은 상황에 따라 해야할 행동이 달라질 때 행동을 추상화함으로써 결합도를 낮춰 확장 시 기존 코드의 수정을 막을 수 있음(OCP)

플러터와 MVVM 패턴

- 데이터 바인딩과 상태관리가 비슷한 부분이 있다고 하지만 네이티브 데이터 바인딩과 플러터의 상태관리의 차이점 또한 존재한다고 함. MVVM에서는 데이터 바인딩이 가장 결정적인 요소이기 때문에(뷰모델이 뷰를 몰라도 되기 때문에) 엄격하게 말하면 MVVM 또한 플러터에서 100% 구현하긴 어렵다고 할 수 있음

코드 공유 시간

1. 상태관리는 GetX를 사용 - 데이터 바인딩이 꼭 rx로 이루어져야하는 것은 아니지만 rx를 통해 구현

2. 커맨드 패턴은 본 코드에서 사용하지 않음 - 규모가 너무 작아 커맨드 패턴을 적용할 수가 없음

결론

- 데이터 바인딩을 통해 뷰를 그리는 방식이 선언적으로 바뀌면서 플러터에 어느 정도 어울리는 패턴이라고 생각
- 실제로 BLoC 패턴이나 GetX 상태관리와 비슷한 부분이 아주 많음(GetX 공식 홈에서는 GetX의 방식이 MVVM이 아니라고 일축)
- 클린 아키텍처 등 다른 레이어드 아키텍처에서 또한 많이 차용되는 프레젠테이션 레이어 패턴