



MATEE


SwiftUI + MVI

Petr Chmelař
23. 3. 2022

Before SwiftUI

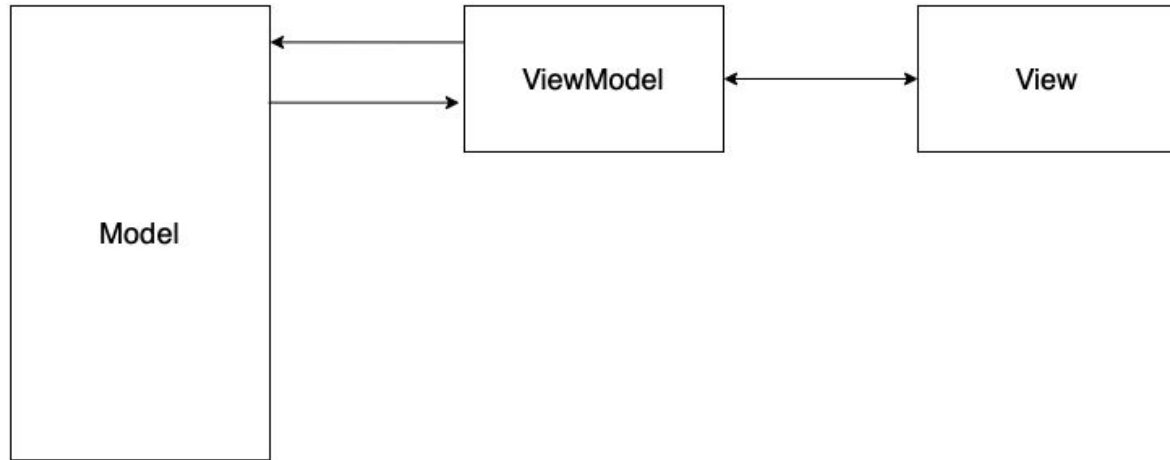
- UIKit 🧓
- Storyboards + XIBs 🎨 (or programmatically)
- Imperative UI updates

UIKit architectures

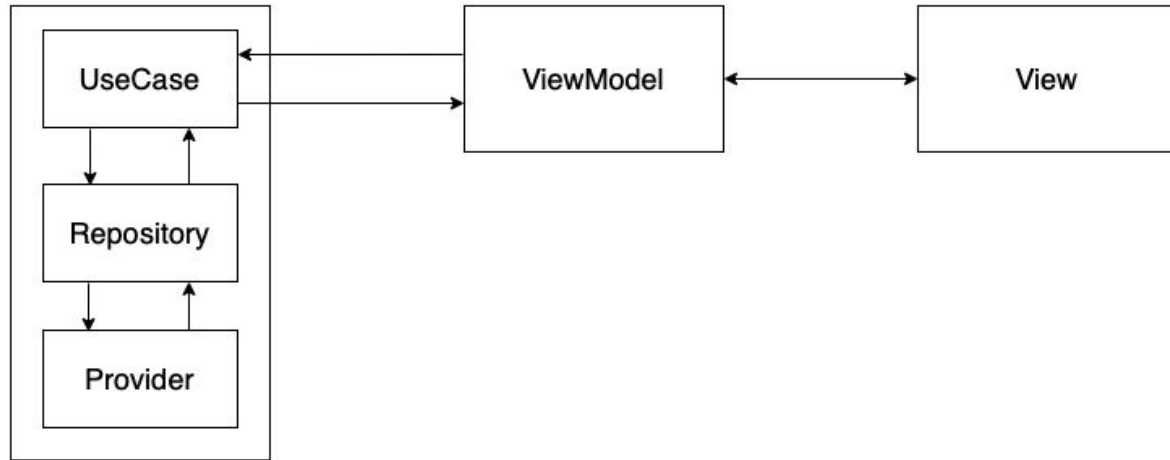
- MVC: ViewController ... 
- MVVM: ViewController + ViewModel
- VIPER, ReSwift, etc

- RxSwift, ReactiveSwift, etc

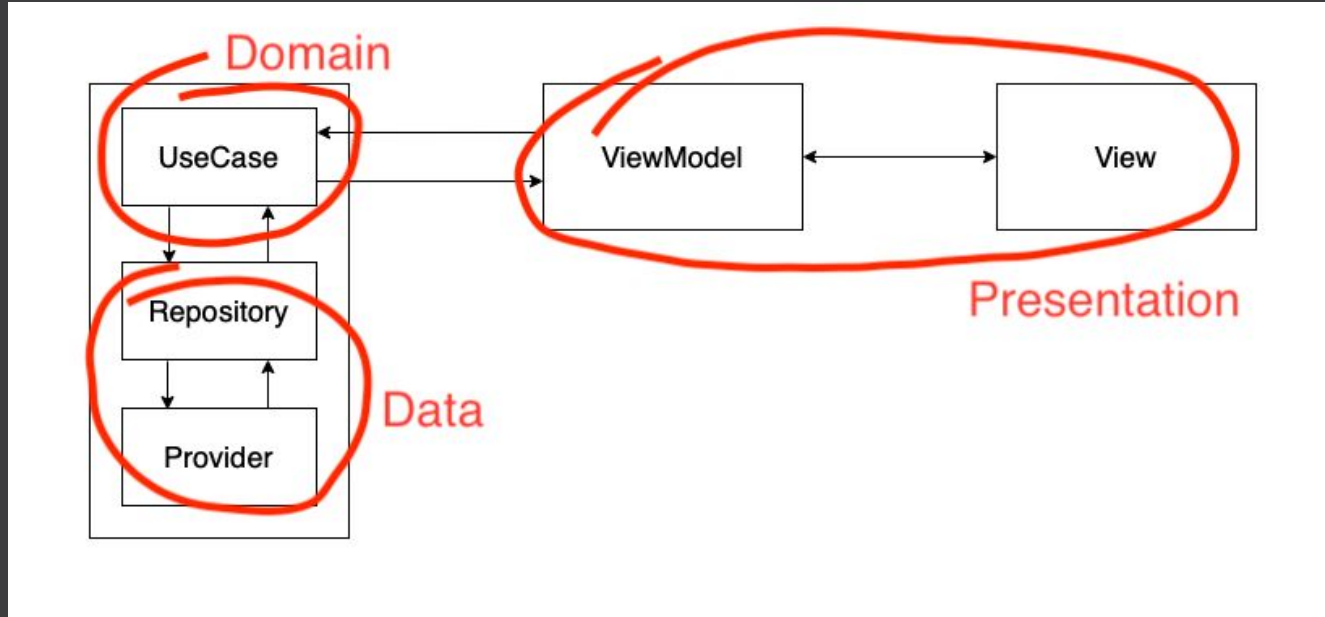
MVVM



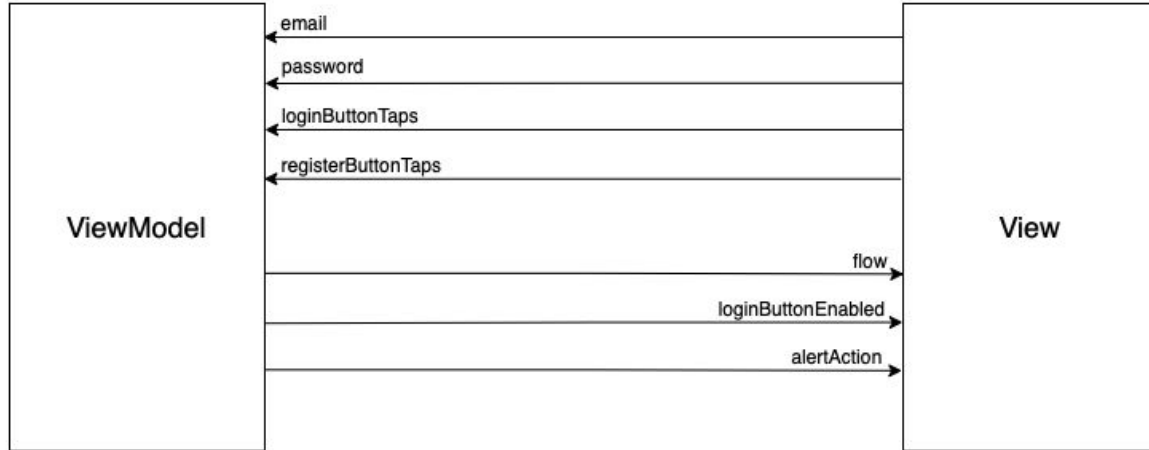
Clean Architecture ?



Clean Architecture ?



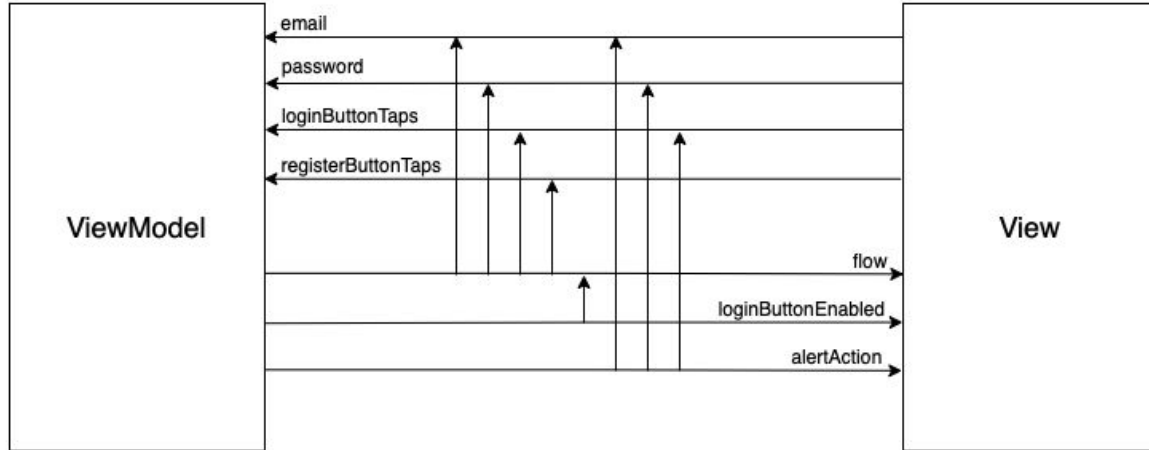
MVVM + RxSwift



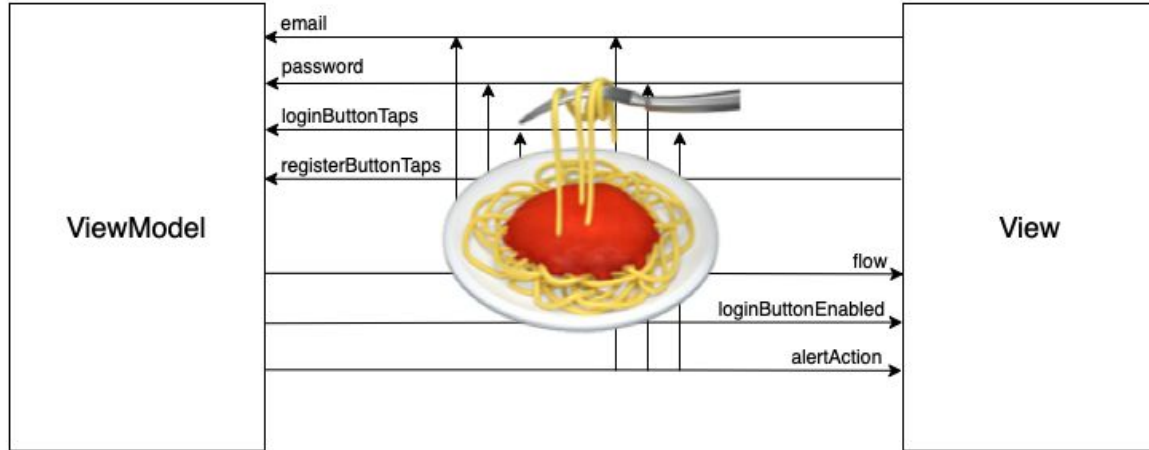
MVVM + RxSwift

```
final class LoginViewModel {  
  
    let input: Input  
    let output: Output  
  
    struct Input {  
        let email: AnyObserver<String>  
        let password: AnyObserver<String>  
        let loginButtonTaps: AnyObserver<Void>  
        let registerButtonTaps: AnyObserver<Void>  
    }  
  
    struct Output {  
        let flow: Driver<Flow.Login>  
        let loginButtonEnabled: Driver<Bool>  
        let alertAction: Driver<AlertAction>  
    }  
  
    init(...) {  
        // Transformations between inputs and outputs  
    }  
}
```

MVVM + RxSwift



MVVM + RxSwift



MVVM + RxSwift

```
let activity = ActivityIndicator()

let inputs = Observable.combineLatest(email, password) { (email: $0, password: $1) }

let login = loginButtonTaps.withLatestFrom(inputs).flatMapLatest { inputs -> Observable<Event<Void>>
    if inputs.email.isEmpty || inputs.password.isEmpty {
        return .just(.error(ValidationError(L10n.invalid_credentials)))
    } else {
        trackAnalyticsEventUseCase.execute(LoginEvent.loginButtonTap.analyticsEvent)
        let data = LoginData(email: inputs.email, password: inputs.password)
        return loginUseCase.executeRx(data).trackActivity(activity).materialize()
    }
}.share()

let flow = Observable<Flow.Login>.merge(
    login.compactMap { $0.element }.map { .dismiss },
    registerButtonTaps.map { .showRegistration }.do { _ in
        trackAnalyticsEventUseCase.execute(LoginEvent.registerButtonTap.analyticsEvent)
    }
)

let messages = ErrorMessages([.httpUnauthorized: L10n.invalid_credentials], defaultMessage: L10n.signing_failed)

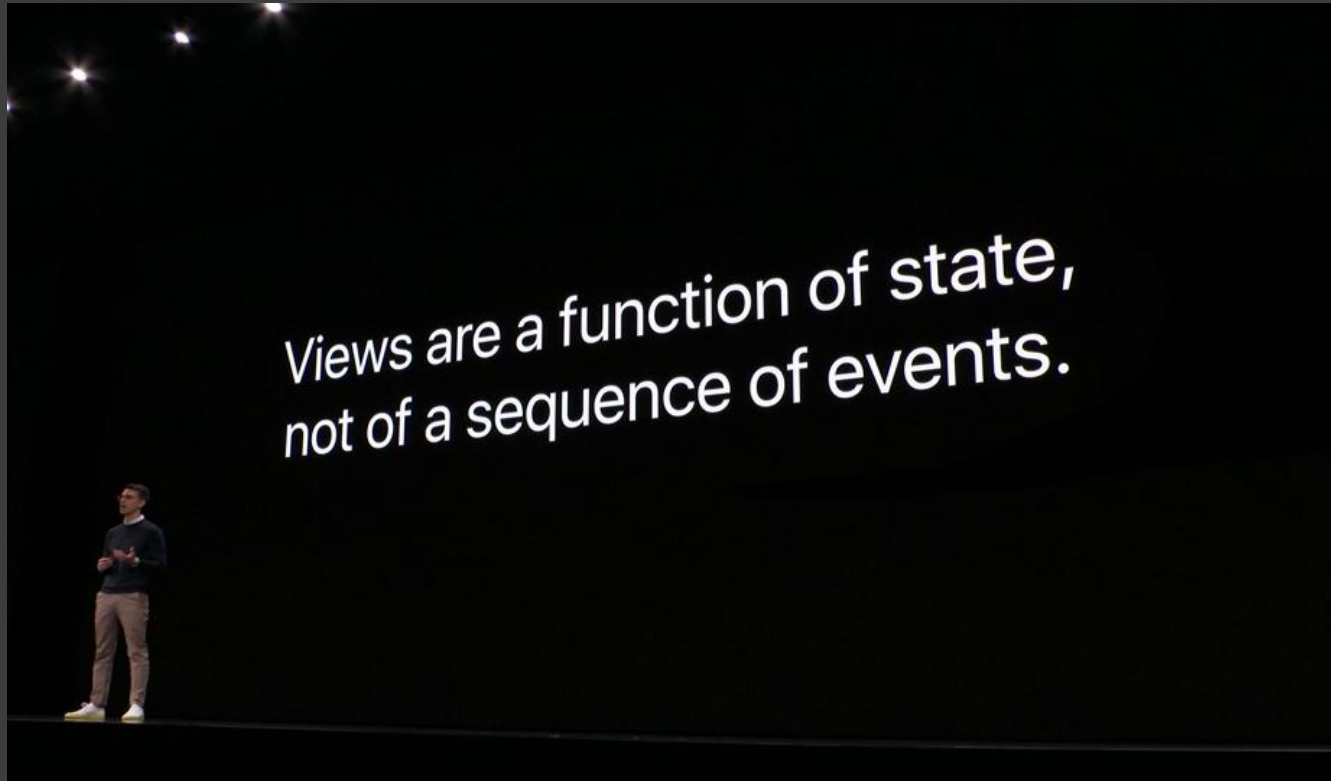
let alertAction = Observable<AlertAction>.merge(
    activity.toWhisper(L10n.signing_in),
    login.compactMap { $0.element }.map { .hideWhisper },
    login.compactMap { $0.error }.map {
        .showWhisper(WhisperData(error: $0.toString(messages)))
    }
)
```



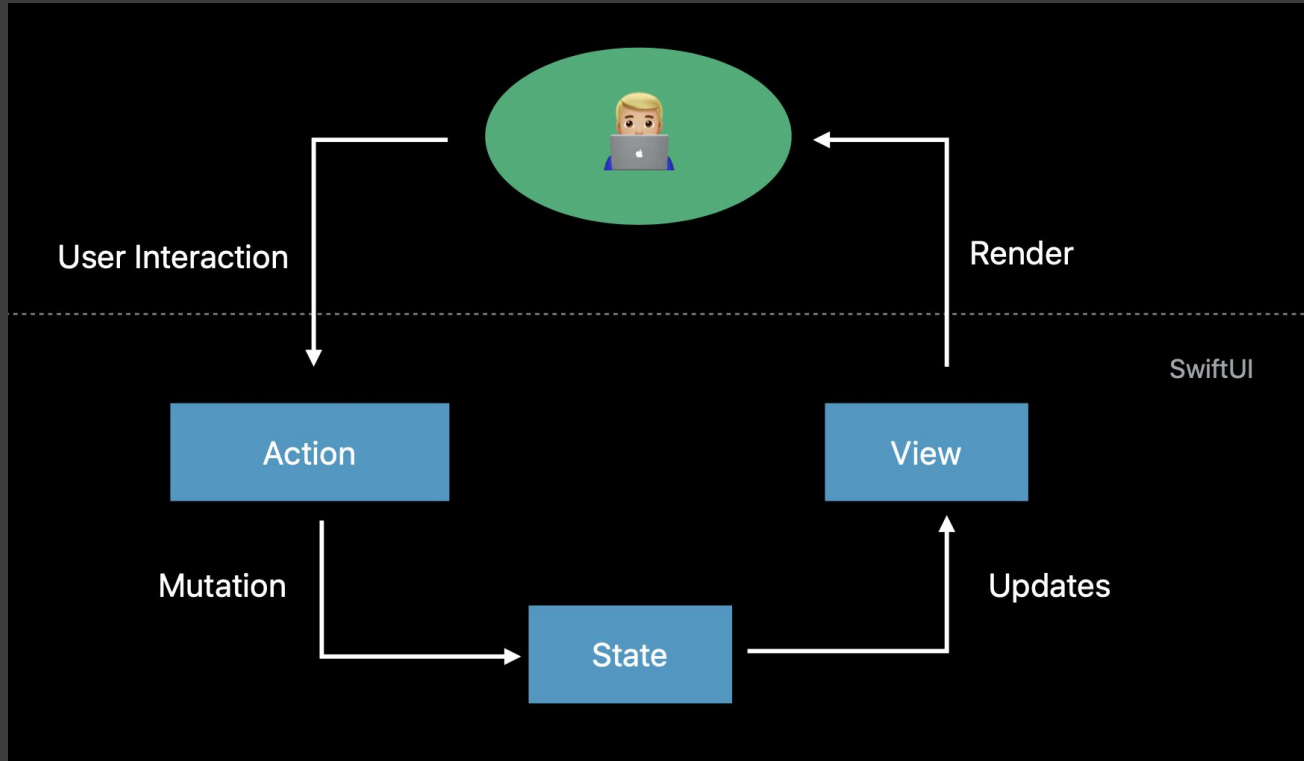
SwiftUI

- Since WWDC 2019 (iOS 13)
- Live previews
- Reactive UI updates 🥰

SwiftUI - Declarative framework



SwiftUI - Declarative framework



SwiftUI architectures

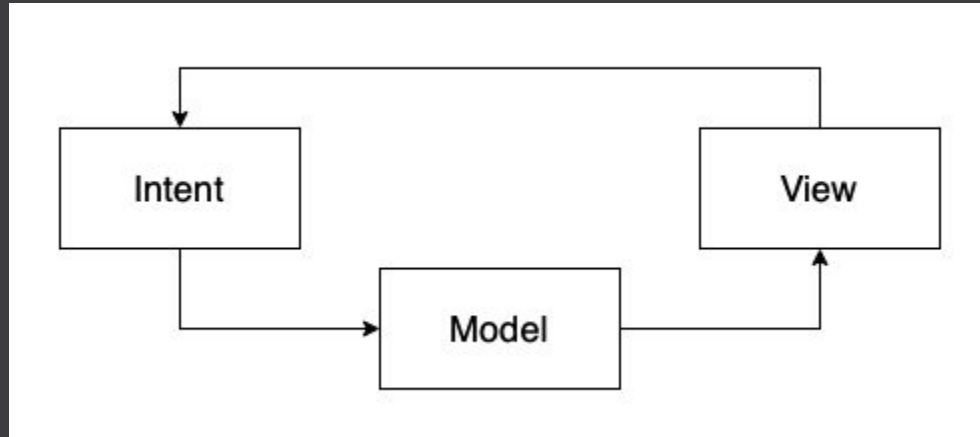
- None - everything in Views
- MVVM / VIPER / ReSwift / ...
- Maybe there is something else? 🤔

MVI: Model-View-Intent

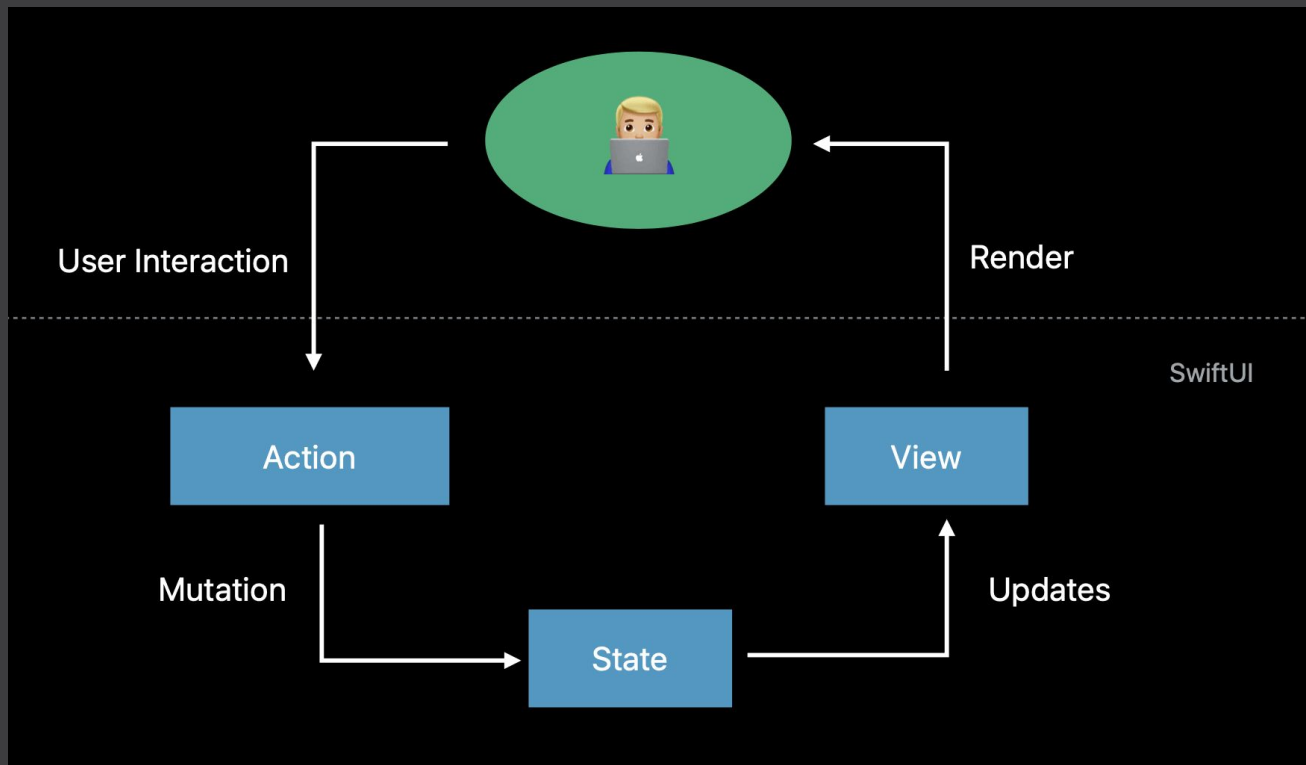
- Introduced in 2015 for Cycle.js
- 1) Reactive & Functional
- 2) Unidirectional Data Flow
- 3) Immutable State
- -> Ideal for declarative UI frameworks 🤝

MVI: Model-View-Intent

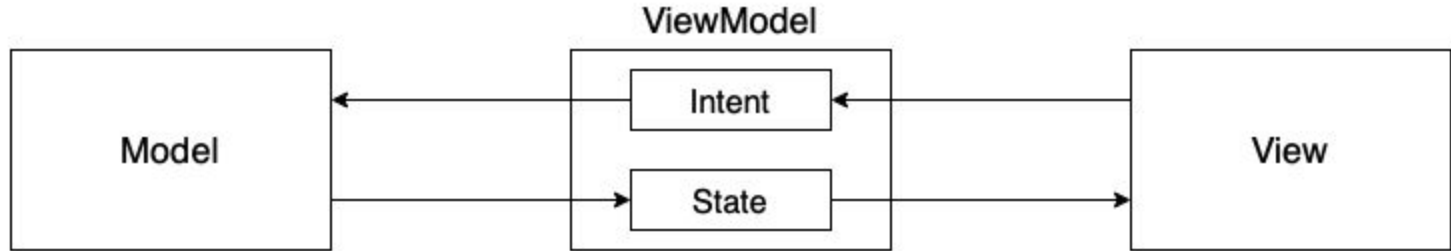
- **Model** represents a state
- **View** represents the UI (function of state)
- **Intent** represents an intention to perform an action



SwiftUI ❤️ MVI



MVI: Model-View-Intent



MVI: Model-View-Intent

```
final class LoginViewModel: ObservableObject {  
    @Published private(set) var state: State = State()  
  
    struct State {  
        var email: String = ""  
        var password: String = ""  
        var loginButtonLoading: Bool = false  
        var alert: AlertData?  
    }  
  
    enum Intent {  
        case changeEmail(String)  
        case changePassword(String)  
        case login  
        case register  
        case dismissAlert  
    }  
  
    func onIntent(_ intent: Intent) {  
        switch intent {  
            // Private intent functions  
        }  
    }  
}
```

Demo

<https://github.com/MateeDevs>



Thank you :)



MATEE