MVVM AT SCALE:

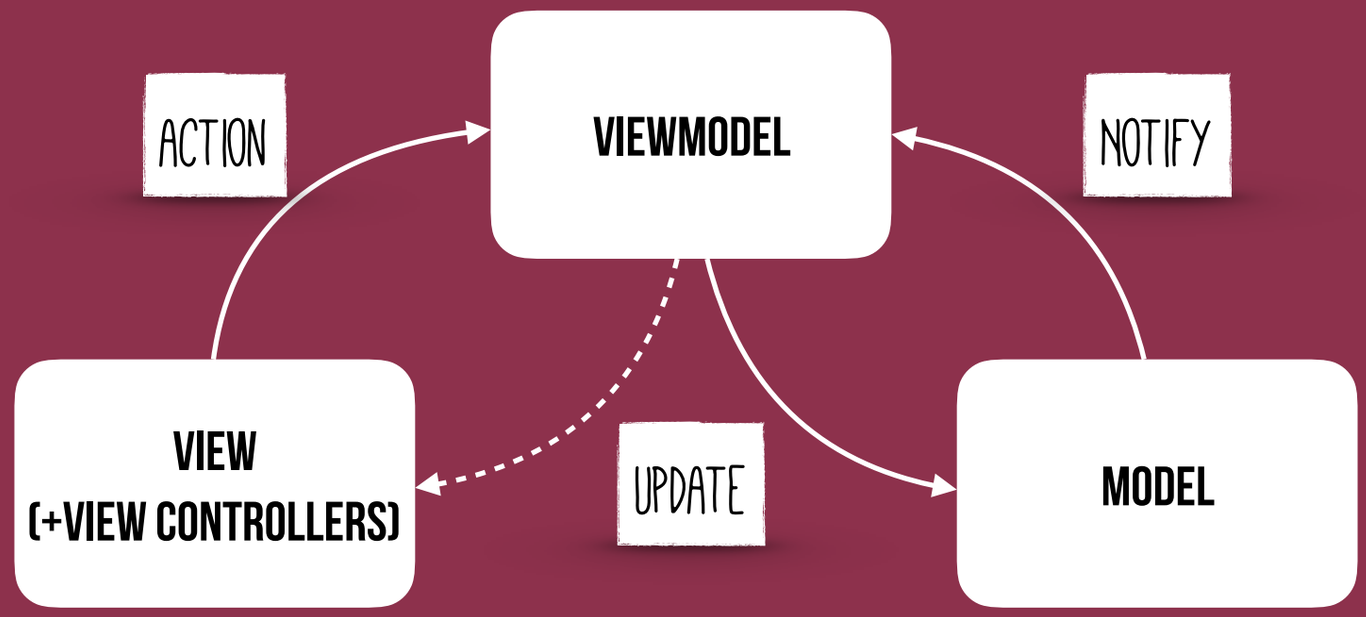
not so Simple

@NATALIYA_BG

“MVVM HAS BEEN A TRENDY TOPIC *lately*.”

—JEMERIAH MORRILL, 2009

QUICK INTRO TO MVVM



DESIGN PATTERNS SPECIFY

ROLES AND COMMUNICATION,

NOT LANGUAGE SPECIFIC TOOLS

Motivation?

MVC:

VIEW CONTROLLERS GETTING BIGGER, MESSY AND NOT TESTABLE

MVVM:

VIEW CONTROLLERS ARE LIGHT AND YOU CAN TEST YOUR BUSINESS LOGIC

Me like it!

1. TESTING AND REFACTORING ADDICT
2. I LIKE SIMPLE THINGS
3. I LOVE THE QUESTION "WHY?"

COMMONLY USED EXAMPLES

Example 1: presentation logic

```
var startDate: Date?
var endDate: Date?
var tripDurationLabel: UILabel

func updateTripDurationLabel() {

}
```

Example 1: presentation logic

```
var startDate: Date?
var endDate: Date?
var tripDurationLabel: UILabel

func updateTripDurationLabel() {
    var text = ""
    if let startDate = startDate, let endDate = endDate {
        text = "All checkins between \(dateFormatter.string(from: startDate))
        and \(dateFormatter.string(from: endDate))"
    }
```

```
}
```

Example 1: presentation logic

```
var startDate: Date?
var endDate: Date?
var tripDurationLabel: UILabel

func updateTripDurationLabel() {
    var text = ""
    if let startDate = startDate, let endDate = endDate {
        text = "All checkins between \(dateFormatter.string(from: startDate))
        and \(dateFormatter.string(from: endDate))"
    } else if let startDate = startDate {
        text = "All checkins after \(dateFormatter.string(from: startDate))"
    }


}
```

Example 1: presentation logic

```
var startDate: Date?
var endDate: Date?
var tripDurationLabel: UILabel!

func updateTripDurationLabel() {
    var text = “”
    if let startDate = startDate, let endDate = endDate {
        text = "All checkins between \(dateFormatter.string(from: startDate))
        and \(dateFormatter.string(from: endDate))"
    } else if let startDate = startDate {
        text = "All checkins after \(dateFormatter.string(from: startDate))"
    } else if let endDate = endDate {
        text = "All checkins before \(dateFormatter.string(from: endDate))"
    }
    tripDurationLabel.text = text
}
```

Example 1: presentation logic

```
var startDate: Date?
var endDate: Date?
var tripDurationLabel: UILabel!

func updateTripDurationLabel() {
    var text = “”
    if let startDate = startDate, let endDate = endDate {
        text = "All checkins between \(dateFormatter.string(from: startDate))
        and \(dateFormatter.string(from: endDate))"
    } else if let startDate = startDate {
        text = "All checkins after \(dateFormatter.string(from: startDate))"
    } else if let endDate = endDate {
        text = "All checkins before \(dateFormatter.string(from: endDate))"
    }
    tripDurationLabel.text = text
}
```

```
tripDurationLabel.text = viewModel.tripDurationString
```

Example 2: business logic

```
let startDatePicker: UIDatePicker
let endDatePicker: UIDatePicker
let startDateTextField: UITextField
let endDateTextField: UITextField

func endDatePickerValueChanged(_ datePicker:UIDatePicker) {

}
```

Example 2: business logic

```
let startDatePicker: UIDatePicker
let endDatePicker: UIDatePicker
let startDateTextField: UITextField
let endDateTextField: UITextField

func endDatePickerValueChanged(_ datePicker:UIDatePicker) {
    // validate selected end date
    // update the end date text label with a formatted date
    // check if current start date is still valid
    // update the start date text label if needed
    // calculate the maximum start date based on the new end date
    // set the maximum start date to the start date picker
}
```

Example 2: business logic

```
let startDatePicker: UIDatePicker
let endDatePicker: UIDatePicker
let startDateTextField: UITextField
let endDateTextField: UITextField

func endDatePickerValueChanged(_ datePicker:UIDatePicker) {
    // validate selected end date
    // update the end date text label with a formatted date
    // check if current start date is still valid
    // update the start date text label if needed
    // calculate the maximum start date based on the new end date
    // set the maximum start date to the start date picker
}
```

```
func endDatePickerValueChanged(_ datePicker:UIDatePicker) {
    viewModel.updateEndDate(datePicker.date)
}
```

Example 3: networking & parsing

```
func fetchTripCheckins() {




}
```

Example 3: networking & parsing

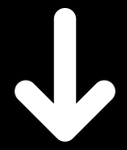
```
func fetchTripCheckins() {
    let url = // construct the url
    let task = URLSession.shared.dataTask(with: url) { (data, response, error) in




    }
}
```

Example 3: networking & parsing

```
func fetchTripCheckins() {
    let url = // construct the url
    let task = URLSession.shared.dataTask(with: url) { (data, response, error) in
        if !error {
            // parse data response
            // save current state
        }
        reloadTripView()
    }
}
```

Example 3: networking & parsing

```
func fetchTripCheckins() {
    let url = // construct the url
    let task = URLSession.shared.dataTask(with: url) { (data, response, error) in
        if !error {
            // parse data response
            // save current state
        }
        reloadTripView()
    }
}
```



```
private viewModel: TripViewModel {
    didSet {
        (…)
        reloadTripView()
        (…)
    }
}
```

LIGHTER VIEW CONTROLLERS, FAT VIEW MODELS

WAIT.. BUT WHY?

IF YOU HAVE BUSINESS AND DATA ACCESS LOGIC IN YOUR PRESENTATION LAYER

that's not really MVC's fault..

Abizer Nasir
@abizern

Follow

Talk about a straw man in blog.uptech.team/taming-great-c…. If you start off badly it's bound to seem better.

> The `RepositoryListViewController` is also a delegate and a data source for the table view. It handles the navigation, formats model data to display and performs network requests. Wow, a lot of responsibilities for just one View Controller!

10:36 AM - 4 Aug 2017 from Greenwich, London

“MVVM TO THE RESCUE!

“MVVM FOR BETTER DESIGNED CODE!

“MVVM FOR A BETTER WORLD!

<PATTERN> TO THE RESCUE!

<PATTERN> FOR BETTER DESIGNED CODE!

<PATTERN> FOR A BETTER WORLD!

I WAS DOING PROGRAMMING

wrong all the time

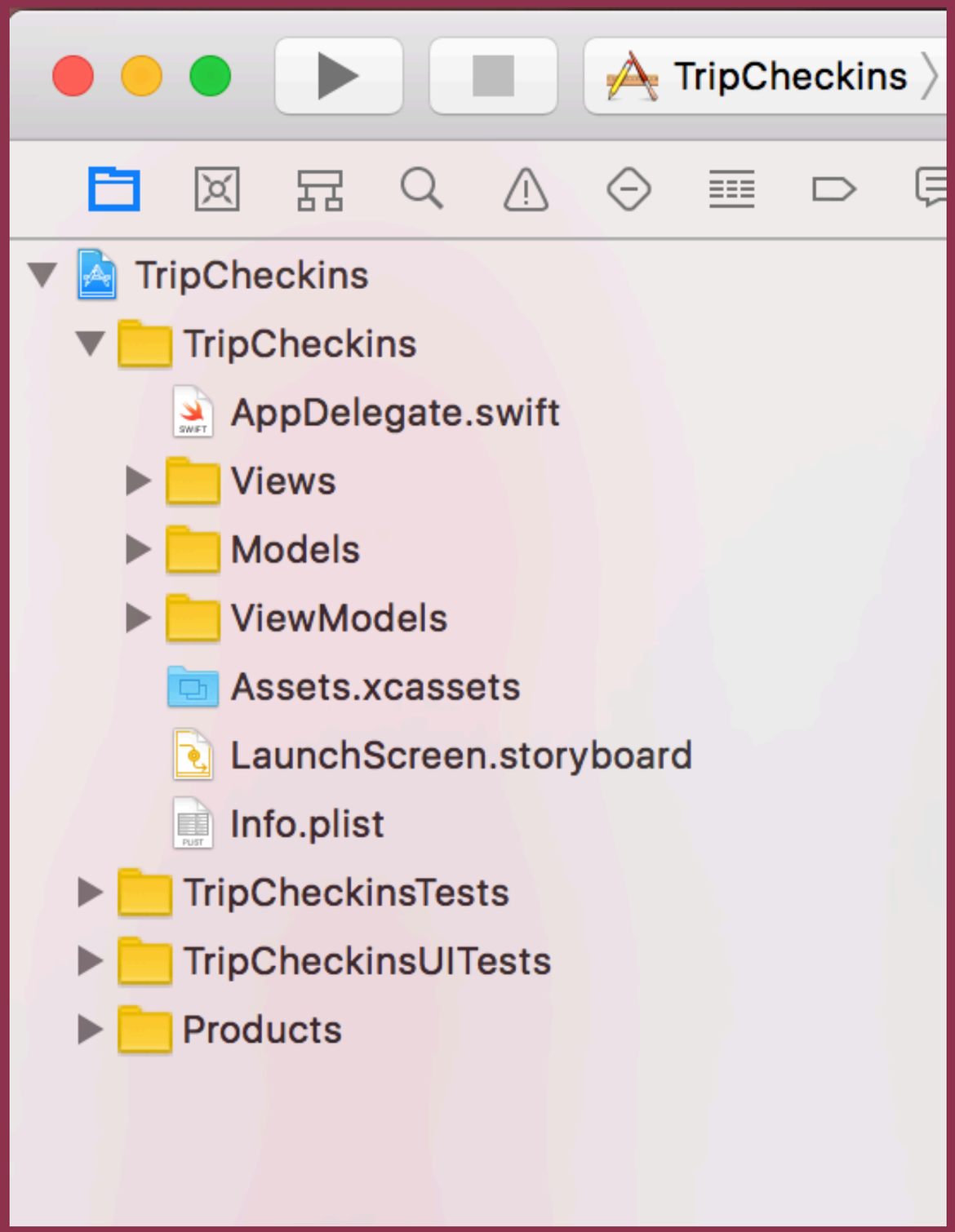
INFORMED DECISIONS

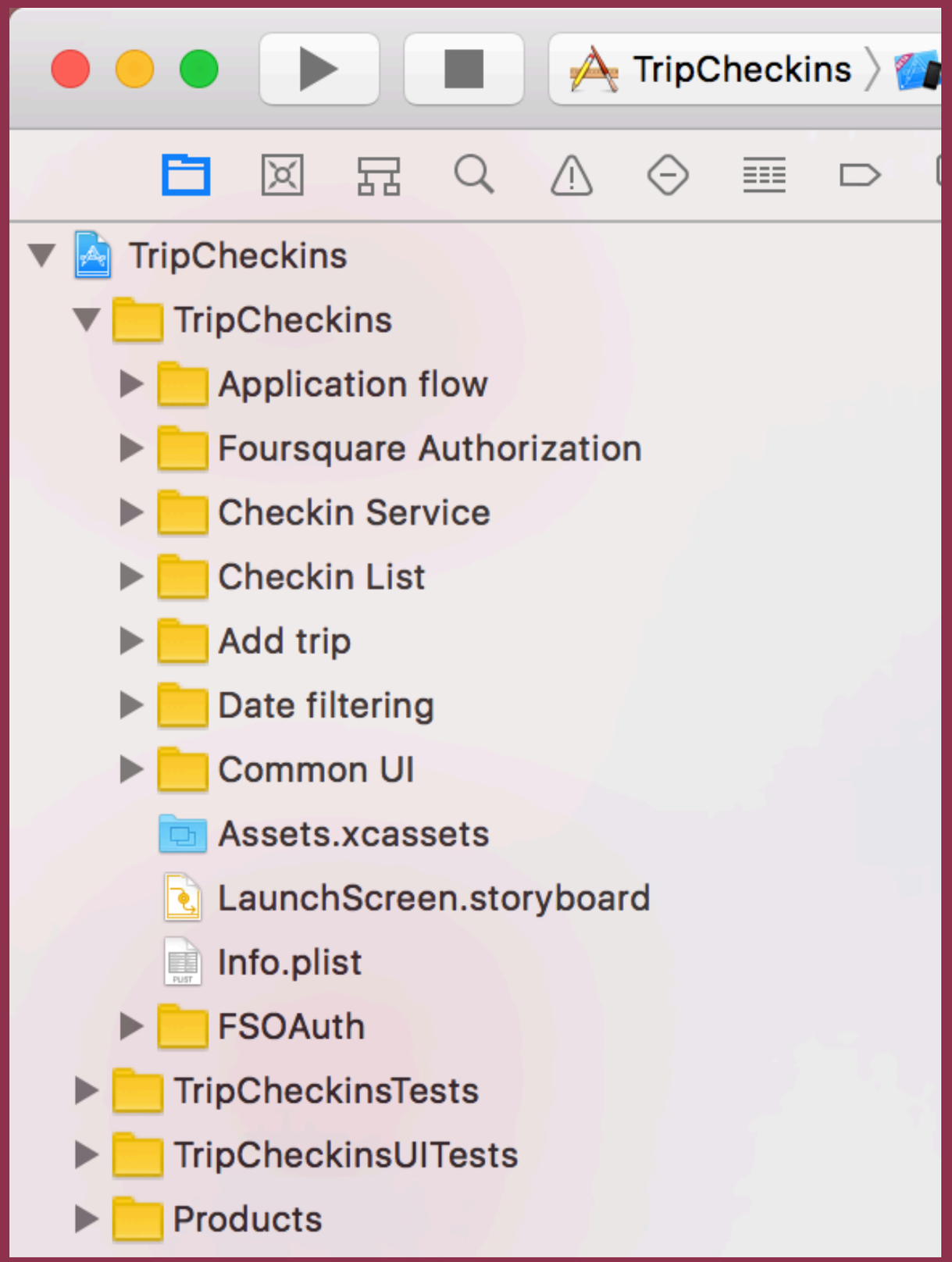
COMMUNITY PRESSURE DRIVEN DEVELOPMENT

EVEN *Hidden* PROMOTING..

- FUNCTIONAL PROGRAMMING
- PROTOCOL-ORIENTED PROGRAMMING
- TESTING
- VALUE TYPES
- ...

I DO USE MVVM


TripCheckins
TripCheckins
TripCheckins
AppDelegate.swift
Views
Models
ViewModels
Assets.xcassets
LaunchScreen.storboard
Info.plist
TripCheckinsTests
TripCheckinsUITests
Products


TripCheckins
TripCheckins
TripCheckins
Application flow
Foursquare Authorization
Checkin Service
Checkin List
Add trip
Date filtering
Common UI
Assets.xcassets
LaunchScreen.storboard
Info.plist
FSOAuth
TripCheckinsTests
TripCheckinsUITests
Products

WHEN I LOOK AT YOUR CODE STRUCTURE I

WANT TO UNDERSTAND

what the app does

NOT WHAT DESIGN PATTERN/FRAMEWORK YOU

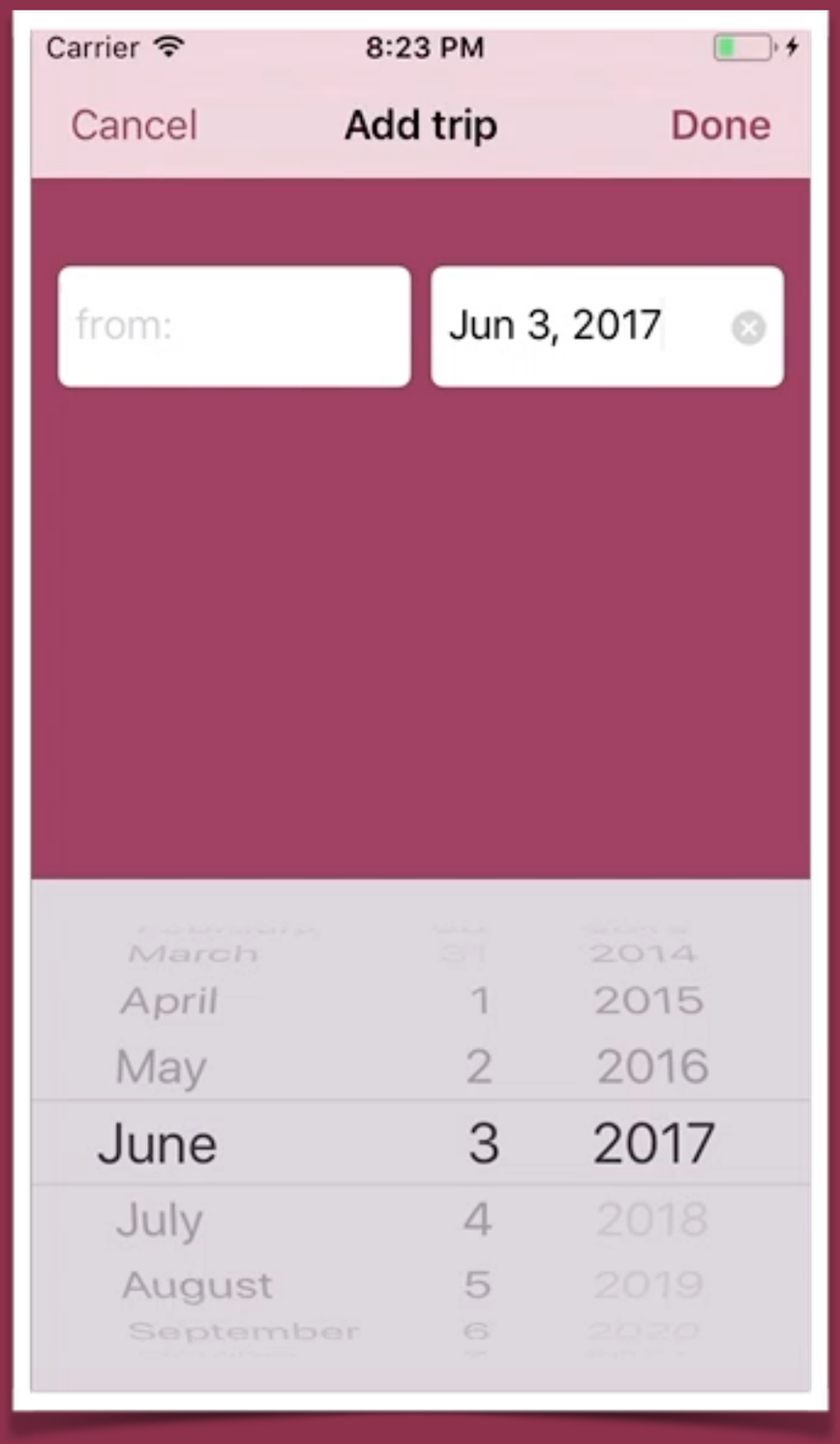
CHOSE

Date filter view

- RESET START DATE WHEN END DATE IS BEFORE START DATE

Date filter view

- UPDATE START DATE PICKER MAXIMUM DATE BASED ON THE END DATE



EXPOSE ONLY THE BITS THE VIEW NEEDS

```
protocol DateFilterCreationViewModel: DateFilterProvider {
    var maximumStartDate: Date { get }
    var maximumEndDate: Date { get }
    var startDateString: String? { get }
    var endDateString: String? { get }

    mutating func updateStartDate(_ startDate: Date?)
    mutating func updateEndDate(_ endDate: Date?)
}
```

- AddTripDateFilterViewModelTests
  - testThatMaximumEndDateIsAutomaticallySet()
  - testThatMaximumStartDateIsOneDayBeforeMaximumEndDate()
  - testThatDateFilterDatesAreInitiallyNil()
  - testThatDateFilterDatesAreEqualToTheOnesProvidedIfValid()
  - testThatDateFilterStartDateIsAtLeastOneDayBeforeEndDate()
  - testThatMaximumEndDateIsEqualToTheOneProvided()
  - testThatMaximumStartDateIsOneDayBeforeInitialDateFilterEndDate()
  - testThatMaximumStartDateIsOneDayBeforeUpdatedDateFilterEndDate()
  - testUpdatingEndDateWithValidDate()
  - testUpdatingEndDateWithNil()
  - testUpdatingStartDateWithValidDate()
  - testUpdatingStartDateWithNil()
  - testUpdatingEndDateWithNotValidDate()
  - testUpdatingStartDateWithNotValidDate()
  - testUpdatingStartAndEndDatesWithValidDates()
  - testThatUpdatingEndDateWithDateBeforeCurrentStartDateSetsStartDateToNil()

WHAT DOES THE VIEW CONTROLLER DO?

```
class AddTripViewController: UIViewController {

    weak var delegate: AddTripViewControllerDelegate?
    let dateFilterCreationView: UIView & DateFilterProvider

    init(dateFilterCreationView: UIView & DateFilterProvider) {
        self.dateFilterCreationView = dateFilterCreationView
        super.init(nibName: nil, bundle: nil)
    }
```

Initialization

WHAT DOES THE VIEW CONTROLLER DO?

```
class AddTripViewController: UIViewController {

    weak var delegate: AddTripViewControllerDelegate?
    let dateFilterCreationView: UIView & DateFilterProvider

    init(dateFilterCreationView: UIView & DateFilterProvider) {
        self.dateFilterCreationView = dateFilterCreationView
        super.init(nibName: nil, bundle: nil)
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = UIColor.tintColor()
        title = "Add trip"
        navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .done,
                                                            target: self,
                                                            action: #selector(doneButtonTapped(_:)))
        navigationItem.leftBarButtonItem = UIBarButtonItem(barButtonSystemItem: .cancel,
                                                           target: self,
                                                           action: #selector(cancelButtonTapped(_:)))
        view.addSubview(dateFilterCreationView)
    }
```

Initialization

Set subviews

WHAT DOES THE VIEW CONTROLLER DO?

```
class AddTripViewController: UIViewController {

    weak var delegate: AddTripViewControllerDelegate?
    let dateFilterCreationView: UIView & DateFilterProvider

    init(dateFilterCreationView: UIView & DateFilterProvider) {
        self.dateFilterCreationView = dateFilterCreationView
        super.init(nibName: nil, bundle: nil)
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = UIColor.tintColor()
        title = "Add trip"
        navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .done,
                                                            target: self,
                                                            action: #selector(doneButtonTapped(_:)))
        navigationItem.leftBarButtonItem = UIBarButtonItem(barButtonSystemItem: .cancel,
                                                           target: self,
                                                           action: #selector(cancelButtonTapped(_:)))
        view.addSubview(dateFilterCreationView)
    }

    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()
        dateFilterCreationView.frame = CGRect(x: 0, y: 100, width: view.frame.width, height: 50)
    }
```

Initialization

Set subviews

Layout

WHAT DOES THE VIEW CONTROLLER DO?

```
class AddTripViewController: UIViewController {

    weak var delegate: AddTripViewControllerDelegate?
    let dateFilterCreationView: UIView & DateFilterProvider

    init(dateFilterCreationView: UIView & DateFilterProvider) {
        self.dateFilterCreationView = dateFilterCreationView
        super.init(nibName: nil, bundle: nil)
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = UIColor.tintColor()
        title = "Add trip"
        navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .done,
                                                            target: self,
                                                            action: #selector(doneButtonTapped(_:)))
        navigationItem.leftBarButtonItem = UIBarButtonItem(barButtonSystemItem: .cancel,
                                                           target: self,
                                                           action: #selector(cancelButtonTapped(_:)))
        view.addSubview(dateFilterCreationView)
    }

    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()
        dateFilterCreationView.frame = CGRect(x: 0, y: 100, width: view.frame.width, height: 50)
    }

    // MARK: Actions
    @objc func doneButtonTapped(_ sender: Any) {
        let dateFilter = dateFilterCreationView.currentDateFilter
        delegate?.addTripControllerDidTriggerAddAction(self, dateFilter: dateFilter)
    }

    @objc func cancelButtonTapped(_ sender: Any) {
        delegate?.addTripControllerDidCancel(self)
    }
}
```

Initialization

Set subviews

Layout

Actions

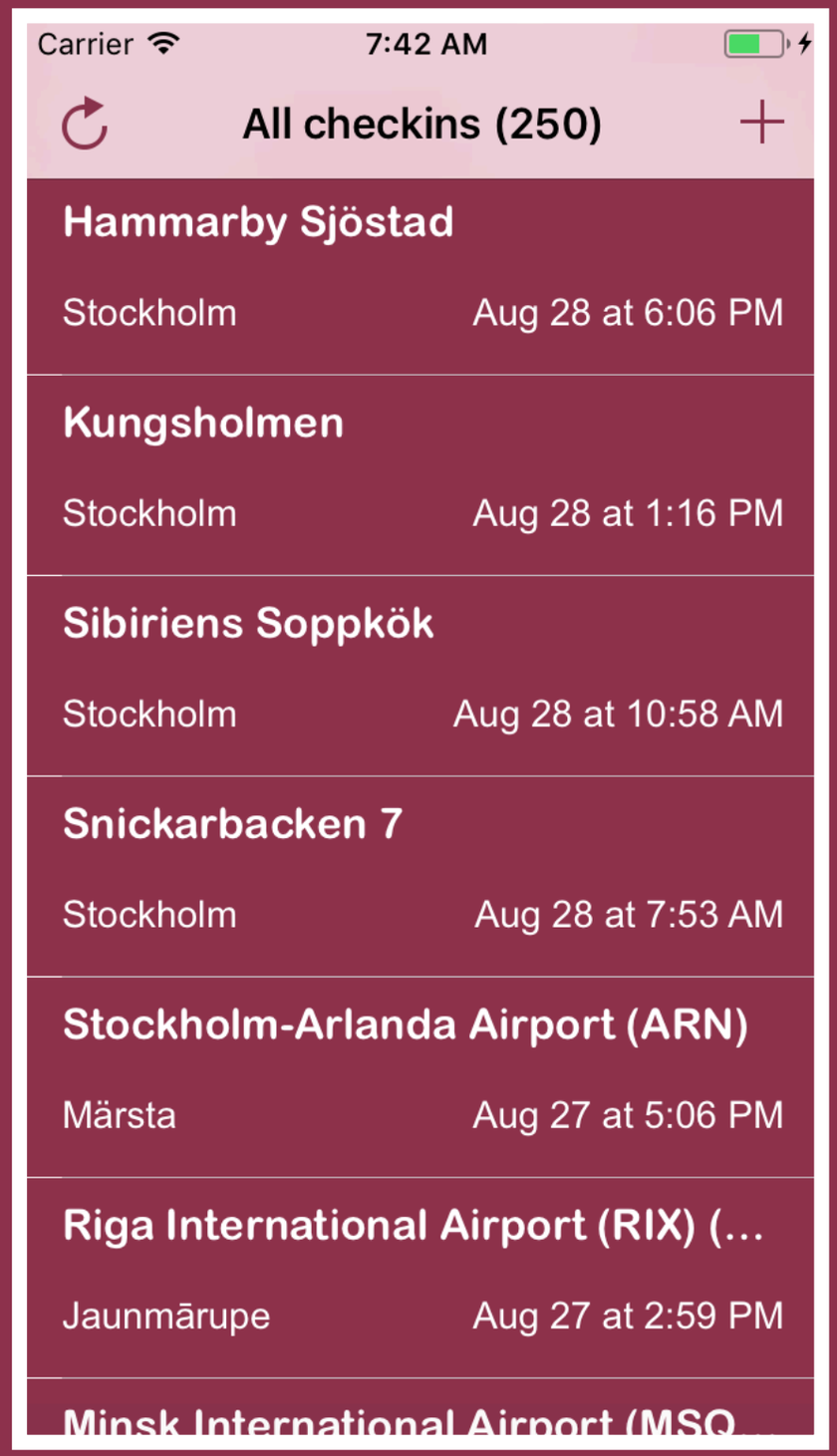
VIEW CONTROLLERS STILL EXIST AND

they behave the same

as in MVC

Checkin list

- SHOWS LIST OF CHECKINS
- HAS LOADING STATE
- HAS ERROR STATE
- SHOWS DYNAMIC TITLE
- HAS RELOAD AND ADD ACTIONS



THE SIMPLE MODEL

```
struct CheckinItem {
    let venueName: String
    let city: String?
    let country: String?
    let date: Date
    let dateTimeZoneOffset: Int
}
```

THE USER FRIENDLY CHECKIN ITEM

```
struct CheckinListItemViewModel {
    let venueName: String
    let locationName: String
    let dateString: String

    private static let dateFormatter: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateFormat = "MMM d 'at' h:mm a"
        return formatter
    }()

    init(checkinItem: CheckinItem) {
        self.venueName = checkinItem.venueName
        self.locationName = checkinItem.city ?? checkinItem.country ?? ""

        let dateFormatter = CheckinListItemViewModel.dateFormatter
        dateFormatter.timeZone = TimeZone(secondsFromGMT: checkinItem.dateTimeZoneOffset)
        self.dateString = dateFormatter.string(from: checkinItem.date)
    }
}
```

NESTED VIEW MODELS

```
struct CheckinListViewModel {
    let title: String
    let listItemViewsType: CheckinListItemViewsType
    let state: ListViewModelState
}

enum CheckinListItemViewsType {
    case compact
    case normal
}

enum ListViewModelState {
    case loadingItems
    case error(String)
    case loadedListItemViewModels([CheckinListItemViewModel])
}
```

VIEWS THAT CAN BE REUSED SHOULD HAVE

separate view models

WHO CREATES THE VIEW MODELS?

```
protocol CheckinListController {
    var currentListViewModel: CheckinListViewModel? { get }
    var onViewModelUpdate: (() -> ())? { set get }

    func reloadListItems()
}
```

WHO CREATES THE VIEW MODELS?

```
protocol CheckinListController {
    var currentListViewModel: CheckinListViewModel? { get }
    var onViewModelUpdate: (() -> ())? { set get }

    func reloadListItems()
}
```

```
class CheckinListViewController: UITableViewController {
    (…)
    init(controller: CheckinListController) {
        self.controller = controller
        super.init(nibName: nil, bundle: nil)

        self.controller.onViewModelUpdate = { [weak self] in
            guard let listViewModel = self?.controller.currentListViewModel else {
                return
            }
            self?.configureWithViewModel(listViewModel)
        }
    }
    (…)
}
```

VIEW MODELS ARE INJECTED
OR CREATED BY OTHER VIEW MODELS

For reusability and testability

WHAT'S WRONG WITH THIS CODE?

```
private var listViewModels: [CheckinListItemViewModel]? {
    didSet {
        self.tableView.reloadData()
    }
}
```

WHO MANAGES THE NAVIGATION?

```
class AppCoordinator {
    (…)
    init(navigationController: UINavigationController,
         authorizationTokenKeeper: AuthorizationTokenKeeper? = nil) {
        self.navigationController = navigationController
        self.authorizationTokenKeeper = authorizationTokenKeeper

        if let token = authorizationTokenKeeper?.authorizationToken() {
            showCheckinsList(authorizationToken: token)
        } else {
            showAuthorizationViewController()
        }
    }

    private func showCheckinsList(authorizationToken token:String) {
        let checkinsService = FoursquareCheckinService(authorizationToken: token)
        let controller = AllCheckinsListController(checkinsService: checkinsService)
        let viewController = CheckinListViewController(controller: controller)
        viewController.delegate = self
        pushViewController(viewController)
    }
    (…)
}
```

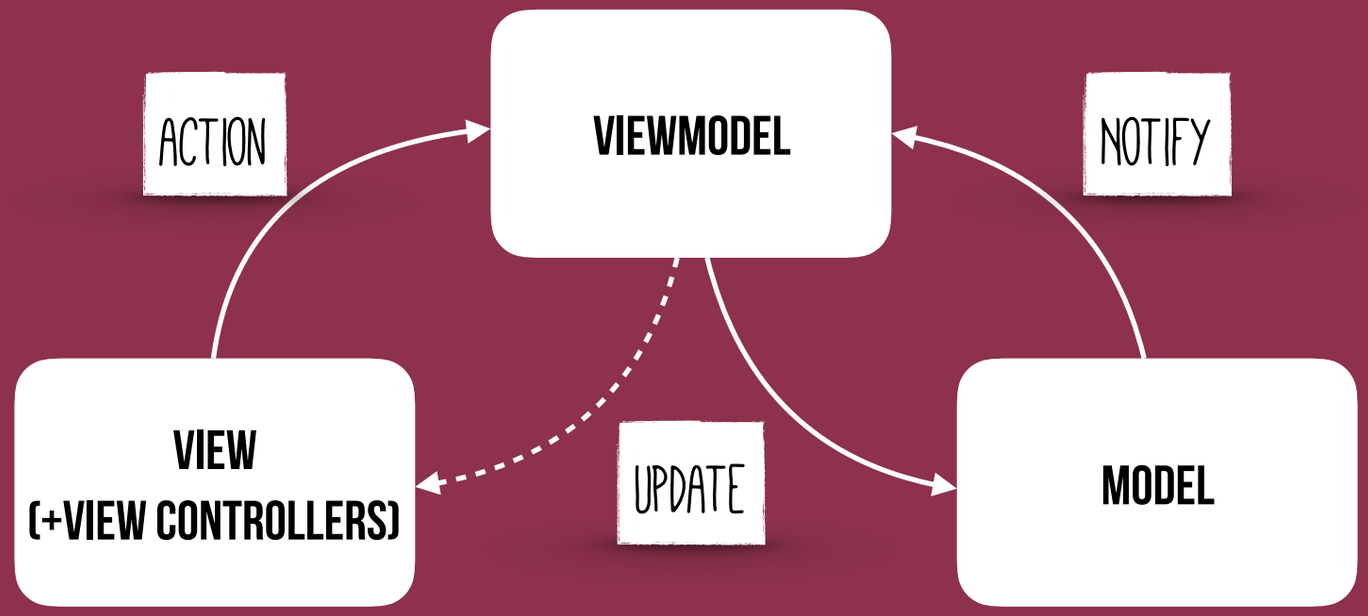
WHO MANAGES THE NAVIGATION?

```
extension AppCoordinator: CheckinListViewControllerDelegate {
    func listViewControllerDidTriggerAddAction(_ controller: CheckinListViewController) {
        let viewModel = AddTripDateFilterViewModel()
        let dateFilterCreationView = DateFilterCreationView(viewModel: viewModel)
        let viewController = AddTripViewController(dateFilterCreationView: dateFilterCreationView)
        let addTripNavigationController = UINavigationController(rootViewController: viewController)
        viewController.delegate = self

        navigationController.viewControllers.last?.present(addTripNavigationController,
                                                           animated: true, completion: nil)

    }
}
```

MVVM is not an architecture,

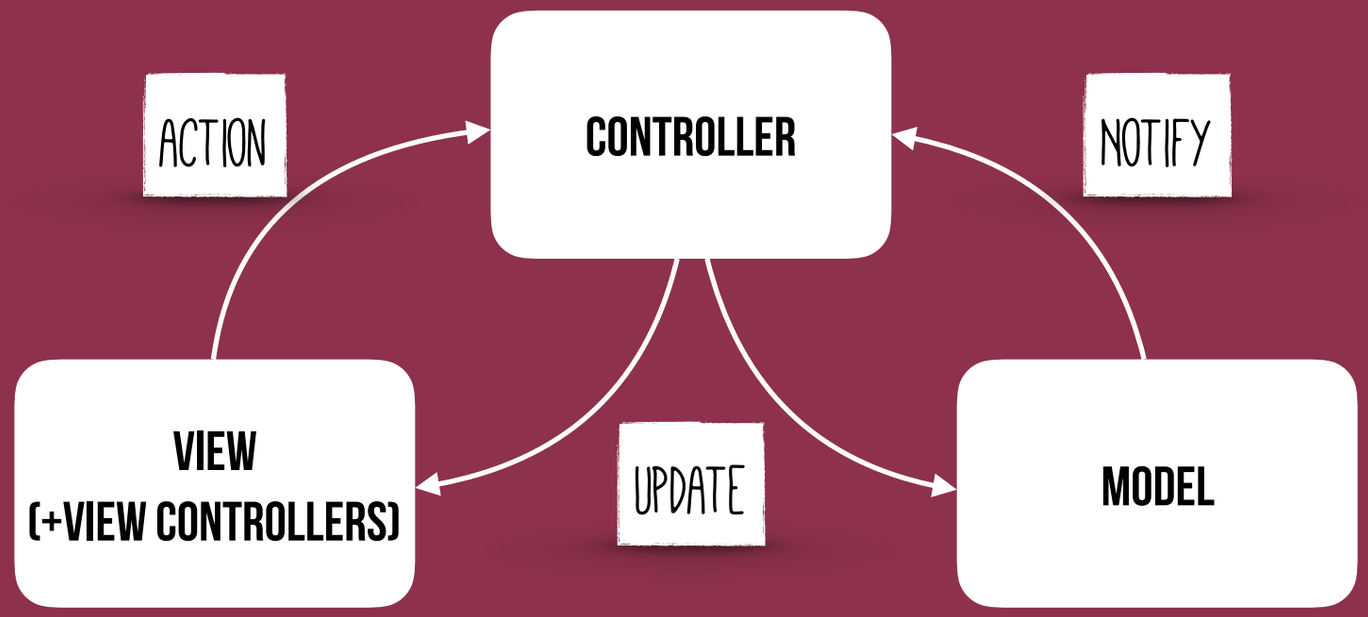
USE DIFFERENT PATTERNS WHERE APPROPRIATE

"VIEW MODEL" IS A LAYER OF OBJECTS



MVC IS NOT THAT DIFFERENT

🤔



THE BEST THING ABOUT MVVM IS ITS NAME DOESN'T CONTAIN THE WORD

"Controller"

Managing complexity

IS THE MOST IMPORTANT TECHNICAL TOPIC IN SOFTWARE DEVELOPMENT.

—CODE COMPLETE, STEVE MCCONNELL

TESTABILITY IS A
SIDE-EFFECT OF
managing complexity

APPLYING DESIGN PATTERNS DOESN'T MEAN

YOU CAN'T

Violate design principles

LET'S GIVE MORE REAL-WORLD EXAMPLES

*

LET'S BE MORE THOUGHTFUL

*

THANKS!

*