Углублённые темы

Занятие №12

Дмитрий Тараев

Отметиться

Организационные вопросы (защита) 1/2

- Защита проектов (экзамен)
  - 23 мая (среда) с 18:00 до 21:00
  - офис Mail.Ru Group, кинозал (2 этаж)
  - можно пригласить гостей
    - (заявки до 18:00 22.05.2018)

- 31 мая (четверг)
  - Обсуждение курса (с 17 до 18)
  - Вручение сертификатов (с 18 до 19)

Организационные вопросы (защита) 2/2

- Защита проектов (экзамен)
  - Слайды + демонстрация работы приложения
    - можно будет одновременно вывести рядом картинку с устройства и ноутбука (слайды)
  - <= 10-15 минут (включая ответы на вопросы)

- Начало в 18:00
  - Если не можете, то предупредите заранее (мы с Геннадием будем «до упора»)

Проекты (к защите)

- 11 проектов (выступали на предзащите):
  - Запись
  - AllForFitness
  - Healthy Food
  - МГТУ-навигатор
  - Docker
  - Приложение для тестов
  - Интересный домовой
  - On Stage
  - Каршеринг
  - PocketBro
  - Криптовалюты

На этой лекции

- Есть какие-то вопросы по теории, которые хотелось бы обсудить?

- Сегодня поговорим о:
  - Отладка программы
    - Time Profiler
    - UI (Core Animation)
  - Custom Transitions
  - Core Animation / Core Graphics
  - URL-схемы
  - Разные архитектуры
  - Публикация приложения в AppStore

- Ответы на вопросы по проектам

Отладка приложения

- Демо

Custom transitions

- При переходе (modal segue)
  - viewController.modalTransitionStyle = UIModalPresentationCustom;
  - viewController.transitioningDelegate = self;
  - presentViewController:animated
- Реализовать протокол (в месте перехода)
  UIViewControllerTransitioningDelegate
  - animationControllerForPresentedController:
    presentingController:
    sourceController:
  - animationControllerForDismissedController:
- Должен быть класс (animator), реализующий протокол
  UIViewControllerAnimatedTransitioning
  - в нём будет сама анимация

Core Animation

- Демо

Core Graphics: ресайз картинки


```
func image(with image: UIImage, newSize: CGSize) -> UIImage? {

    UIGraphicsBeginImageContext(newSize)

    image.draw(in: CGRect(x: 0, y: 0, width: newSize.width, height:
newSize.height))

    let newImage = UIGraphicsGetImageFromCurrentImageContext()

    UIGraphicsEndImageContext()

    return newImage
}
```

URL-схемы

- С iOS 9 нужно прописать параметр **LSApplicationQueriesSchemes** в info.plist
  - в нём одна или несколько URL-схем, которые будут вызываться в проекте

- Демо

Universal links

Архитектуры приложений

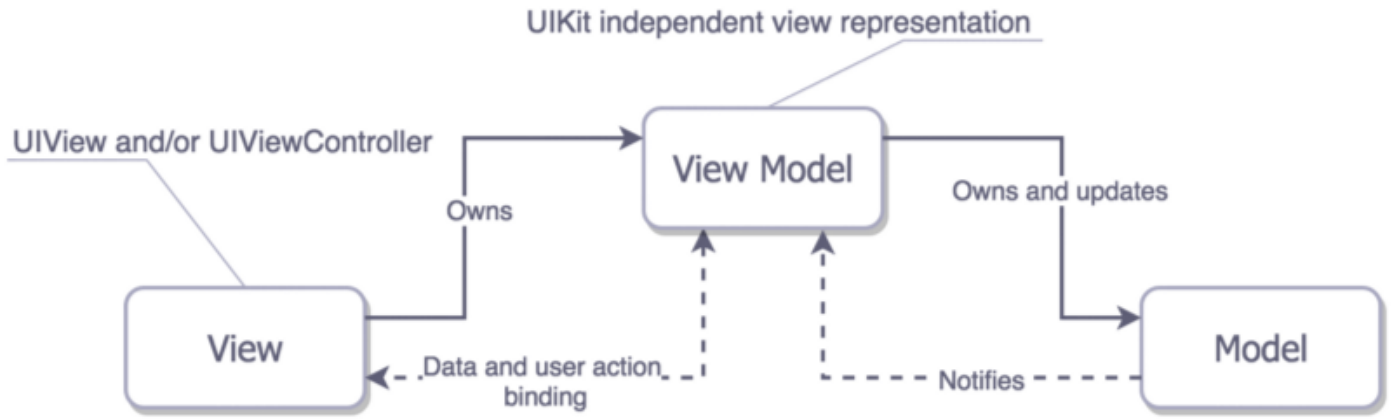
- Предпосылки к возникновению разных архитектур
  - MVC не до конца учитывает специфику мобильных приложений
  - MVC может трактоваться кучей способов, и это плохо
- Что решают другие архитектуры?
  - Унификация кода для того, чтобы можно было легче в нём разобраться
  - Уменьшение неопределённости при написании кода

Архитектуры приложений

- VIPER
  - Стильно, модно, молодёжно
  - V - view (представление)
  - I - interactor (бизнес-логика)
  - P - presenter (связь всего со всем)
  - E - entity (сущности)
  - R - router (маршрутизатор)

Архитектуры приложений

- MVVM (Model View ViewModel)
  - Отлично работает в связке с Rx* штуками
  - Отчасти решает проблему работы со сложными данными



Архитектуры приложений

- VIPER
  - Основная единица — модуль
    - Модули независимы друг от друга
    - Модуль это не обязательно == экран, на сложном экране может быть несколько модулей
    - Взаимодействие между модулями осуществляется через интерфейсы ModuleInput и ModuleOutput

Архитектуры приложений

- VIPER
  - Плюсы
    - Переиспользуемость модулей (хотя на практике это не часто применяется)
    - Тестируемость
  - Минусы
    - Многословность, на экран надо писать минимум 5 классов и кучу интерфейсов (отчасти решается кодогенерацией)

Архитектуры приложений

- UDF (Uni-Directional Data Flow)
  - Появился в вебе, flux / redux
  - Немного ортогонален MVC/MVVM/VIPER истории
  - Может использоваться как во всём приложении, так и в рамках одного модуля
  - Основная концепция — предсказуемое состояние в любой момент времени
    - Action — действие по изменению состояния
    - Reducer — обрабатывает действие и меняет состояние

Архитектуры приложений

- UDF (Uni-Directional Data Flow) / Actions
  - Несут в себе информацию о том, что нужно сделать (нажатие на элементы, запросы в сеть, всё, что поменяет состояние)
  - Обрабатываются последовательно, в порядке поступления
  - Обрабатываются цепочкой обработчиков (reducers)

Архитектуры приложений

- UDF (Uni-Directional Data Flow) / Reducers
  - func (state: State, action: Action) -> State
  - Строго чистые (pure) функции
  - Вызываются последовательно
  - Могут или изменять состояние сразу, или порождать actions для изменения его через время, и никак иначе

Архитектуры приложений

- UDF (Uni-Directional Data Flow)
  - Плюсы
    - В идеале полностью прогнозируемое состояние приложения в любой момент времени
    - Простота тестирования (надо тестировать только reducers)
  - Минусы
    - Всевозможные анимации и переходы достаточно сложно описываются в этой парадигме

FRP

- Functional Reactive Programming
  - Оно же “реактивное” программирование
  - Основано на потоках данных
  - Обработчики изменяют, объединяют, фильтруют эти потоки
  - Обработчики как правило это pure функции
  - Отлично работает для
    - ui bindings (mvvm)
    - сложная логика запросов к серверу

Кросс-платформенная разработка

- Предпосылки
  - Часто приложения (или их части) повторяются для нескольких платформ
  - Хочется экономить силы и вея разработки
  - Выход — писать подо всё сразу

- Какие бывают
  - На базе HTML
  - На базе нативных элементов, рисуемых какой-то средой
  - На базе своего набора виджетов

Кросс-платформенная разработка

- Средства на базе HTML
  - По сути это веб-сайты, которые завернуты в приложение
  - UIWebView + JavaScriptCore для взаимодействия с системными компонентами
- Это не очень
  - Медленно
  - Ограниченная функциональность
  - Некрасивые :(

Кросс-платформенная разработка

- ReactNative
  - Продукт на базе React — отличной веб-библиотеки
  - Поддерживается Facebook
  - JavaScript/TypeScript
  - JSX для рендеринга в нативные элементы
  - Куча библиотек на js
  - Очень неплох

Кросс-платформенная разработка

- Xamarin, Appcelerator etc
  - Тоже используют нативные ui элементы
  - Менее распространены
  - Платные

Кросс-платформенная разработка

- Flutter
  - Поддерживается и разрабатывается Google
  - Пока в alpha, но всё впереди, google уже активно использует его в своей разработке
  - На базе языка dart (тоже от google, вот сюрприз)
  - Своя библиотека виджетов, рисуется средствами фреймворка через gl
  - Ориентируется на FRP/UDF подход
  - Выглядит очень перспективно

Заключение

- **Сегодня**
  - Отладка программы
  - Core Animation
  - Core Graphics
  - URL-схемы
  - Разные архитектуры
  - Публикация приложения в AppStore
- **В среду (23.05.2017 с 18:00 до 21:00)**
  - Защита проектов в офисе Mail.Ru Group (кинозал, 2 этаж)
- **Отзыв** (важно!)

Полезные ссылки

- **Instruments Tutorial with Swift: Getting Started**
  https://www.raywenderlich.com/97886/instruments-tutorial-with-swift-getting-started
- **objc.io Issue 19: Debugging**
  https://www.objc.io/issues/19-debugging/

- **The Book of VIPER**
  https://github.com/strongself/The-Book-of-VIPER
- **MV(X) архитектуры**
  https://habrahabr.ru/company/badoo/blog/281162/
- **ReSwift**
  https://github.com/ReSwift/ReSwift
- **React Native**
  https://facebook.github.io/react-native/docs/getting-started.html
- **Flutter**
  https://flutter.io