MVC
(Model - View - Controller)

Занятие №3

Дмитрий Тараев

Организационная часть

- Отметиться!

- На прошлом занятии
  - База для написания приложений
- **Сегодня**
  - MVC и другие архитектурные паттерны

- Оставить отзыв (после занятия)

Вопросы к аудитории

- Кто скачал из репозитория код примера с прошлого занятия?
- Кто смотрел пример, переписанный на Objective-C?

- Кто начал работу над своим проектом?

- У кого нет темы?
- Кто ещё не разбился на группы?

- Вопросы?

Домашние задания - ДЗ 0

1. **ДЗ 0** (срок **21.03.2017** 00:01) - максимум **5 баллов**
   - Разбиться на группы, придумать тему, создать репозиторий

2. ***ДЗ 1*** *(срок **28.03.2017** 00:01) - максимум **5 баллов***
   - *Базовая функциональность*

3. ***ДЗ 2*** *(срок **11.04.2017** 00:01) - максимум **10 баллов***
   - *Использование списков элементов*

4. ***ДЗ 3*** *(срок **09.05.2017** 00:01) - максимум **15 баллов***
   - *Работа с сетью и локальной базой данных*

5. ***ДЗ 4*** *(срок **23.05.2017** 00:01) - максимум **10 баллов***
   - *Работа с дополнительными API*
     - *карта*
     - *камера / фотогалерея*

Текущая ситуация с проектами

- Заявлено:
  - 15 проектов
- По факту
  - 8 GitHub репозиториев

- Заявлено:
  - 29 участников
- По факту
  - 15 GitHub аккаунтов пользователей

- Сегодня последняя возможность согласовать тему/команду

План лекции

- Повторение (мать учения)
- MVC
  - Model
  - View
  - Controller
- Другие архитектурные паттерны

Повторение

- Жизненный цикл приложения (состояния приложения)
- Жизненный цикл контроллера
- Контейнеры
  - UITabBarController
  - UINavigationController
- UIView + наследники (label, button…)
- UIScrollView + наследники (table view, collection view) - на следующей лекции
- Auto layout
- Создание UI с помощью
  - Interface Builder
  - кодом
- Segue vs. работа с Navigation Controller напрямую

MVC

Model-View-Controller

- Разделение зон ответственности
- Даёт возможность переиспользования кода

Model

- сами данные
- обеспечение доступа к данным
- логика для изменения данных

- Ещё называют реализацией бизнес-логики приложения

- Не зависит от UI
- Чаще всего потомок NSObject (или базового класса базы данных)

View

- Стандартные элементы (view, label и т.п.)
- Показывают данные
- Общаются с пользователем
- Ничего не знают о модели

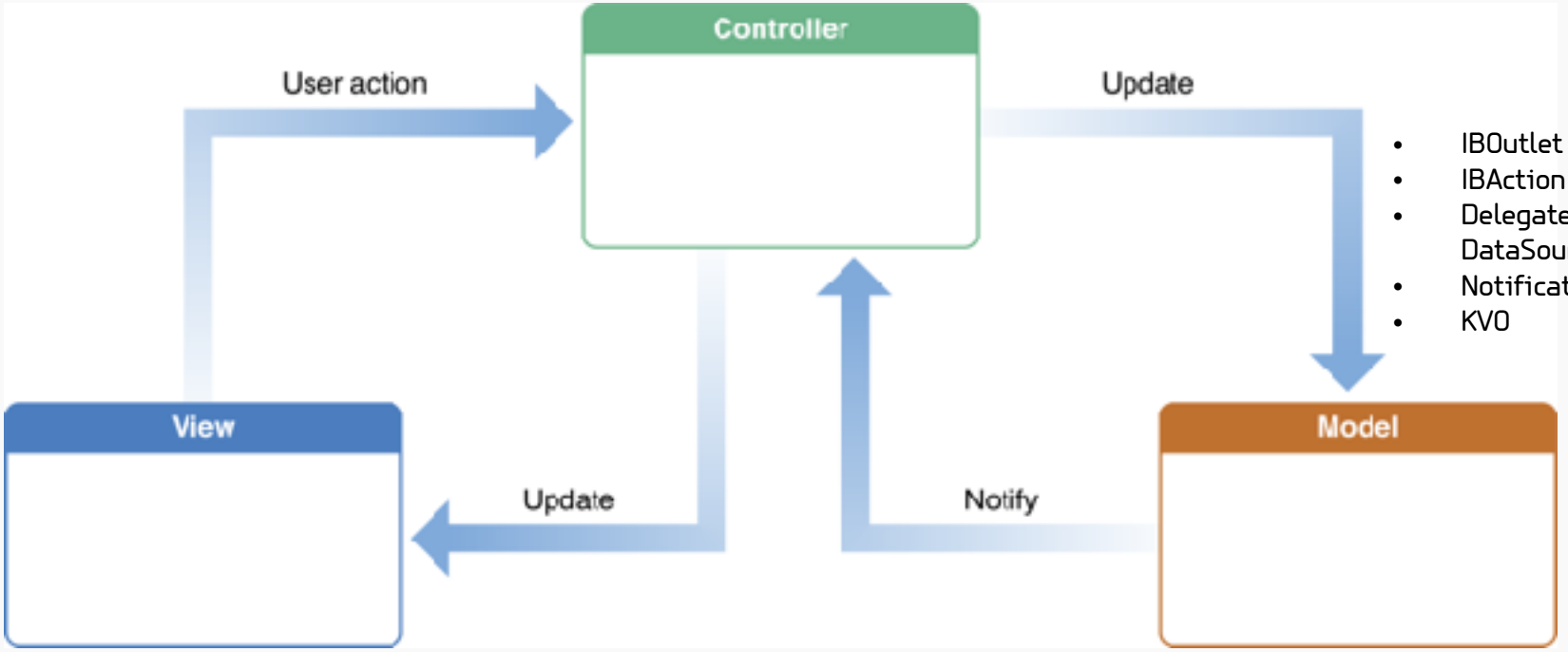
Controller

Координирует взаимодействие между View и Model (посредник)

- Заполняет view данными
- Обрабатывает события от view

- Передаёт данные в модель
- Берёт данные из модели (модель уведомляет об изменении данных)

Схема MVC



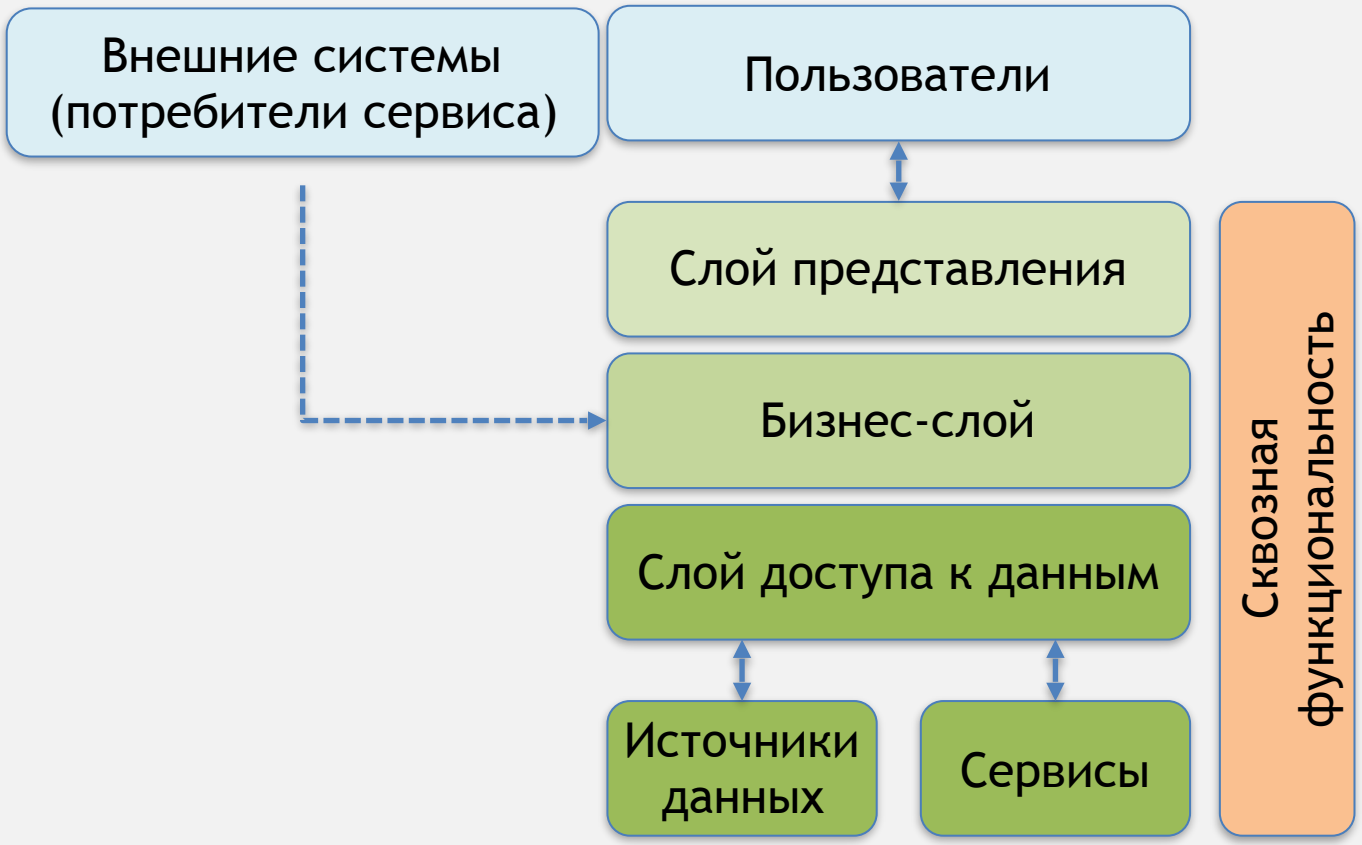
- IBOutlet
- IBAction
- Delegate/ DataSource
- Notification
- KVO

https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

Типовые составляющие приложения

1. Пользовательский интерфейс (UI)
2. Бизнес-логика
3. Сетевое взаимодействие
4. Хранение/кеширование данных

Layered архитектура приложения



Способы коммуникации между слоями

1. Делегат
2. Блоки
3. Notifications
4. KVO
5. IBOutlet / IBAction (между View и Controller)

Основные архитектурные паттерны

1. MVC
2. MVP
   - похож на MVC, но
     - presenter не занимается layout
     - view controller относится к view
3. MVVM
4. VIPER
   - ещё большее разделение обязанностей
     - view - UI
     - presenter - бизнес-логика, связанная с UI
     - interactor - бизнес-логика, связанная с данными
     - router - переходы между модулями
     - entity - объекты данных

https://habrahabr.ru/company/badoo/blog/281162/

Ссылки

- Про «тяжёлые» view controller'ы
  https://www.objc.io/issues/1-view-controllers/

- Про структуру проекта в Xcode:
  https://habrahabr.ru/post/261907/

- «Архитектурный дизайн мобильных приложений»
  часть 1: https://habrahabr.ru/company/redmadrobot/blog/246551/
  часть 2: https://habrahabr.ru/company/redmadrobot/blog/251337/

- Про архитектурные паттерны:
  https://habrahabr.ru/company/badoo/blog/281162/