



# Мобильные приложения

РИП. Часть 1



# Что нужно для разработки

iOS приложений



## Языки для iOS-разработки



Objective-C



Swift



## Кроссплатформенные языки



# Flutter

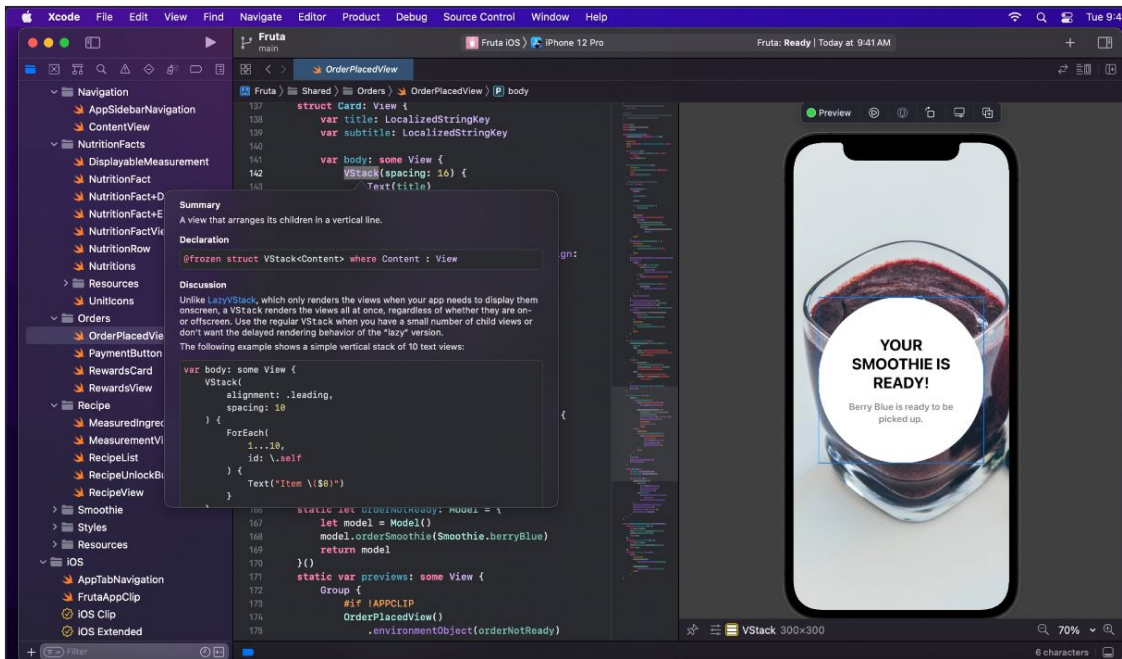


Языки для android-разработки



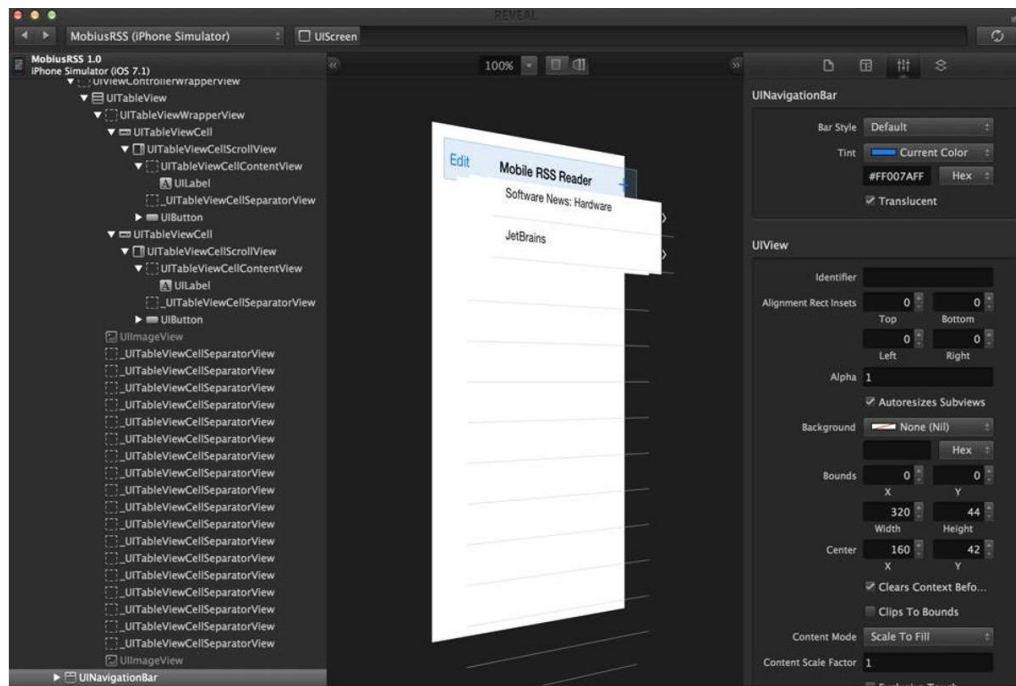


# Xcode for iOS



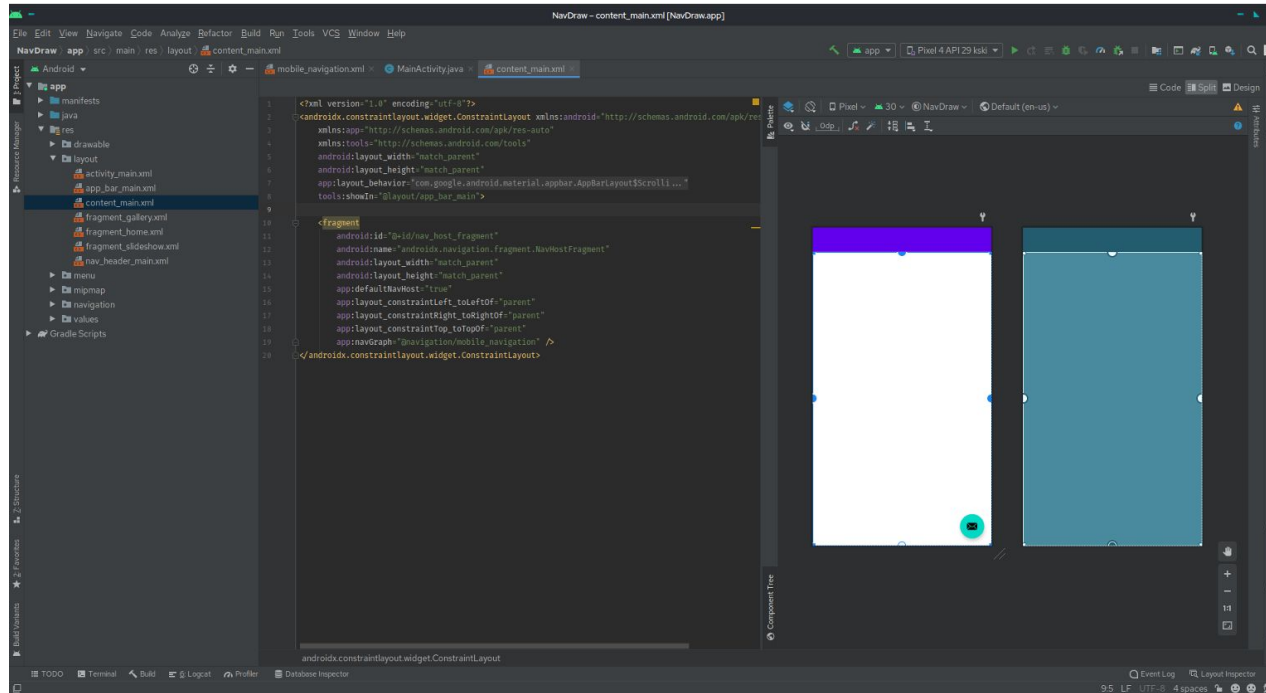


# Appcode iOS





# Android studio







# Основные элементы экранов и управления

iOS приложений

# Основные элементы экранов и управления

## UIControl:

- UIButton
- UITextField
- UISwitch



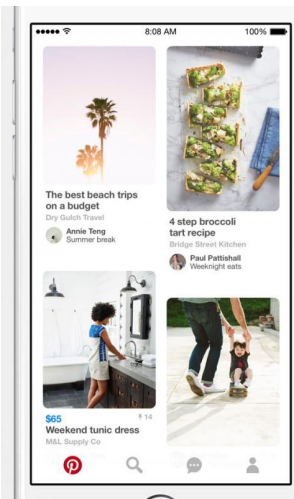
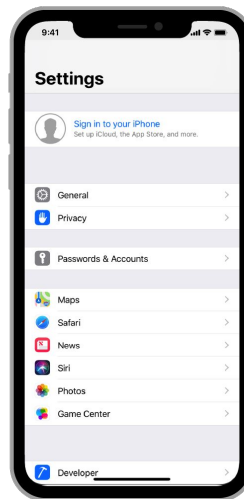
## UIView:

- UILabel
- UIImageView



## UIScrollView:

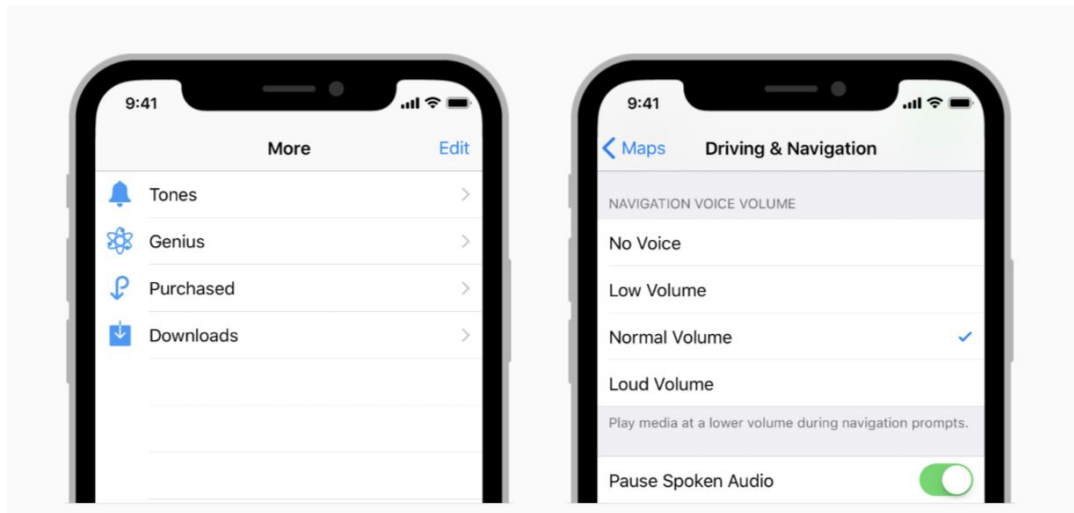
- UITableView
- UICollectionView



# UITableView

Виды:

- static / dynamic
- plain / grouped
- разные виды ячеек



Static Table

Access to some settings >

Access to some settings, but this time with a much longer description. This label should extend over multiple lines especially for the larger text sizes >

An example of a cell with a title and subtitle ✓

A subtitle using subhead style

# UITableViewCell

## Структура



none

disclosureIndicator >

detailButton ⓘ

detailDisclosureButton ⓘ >

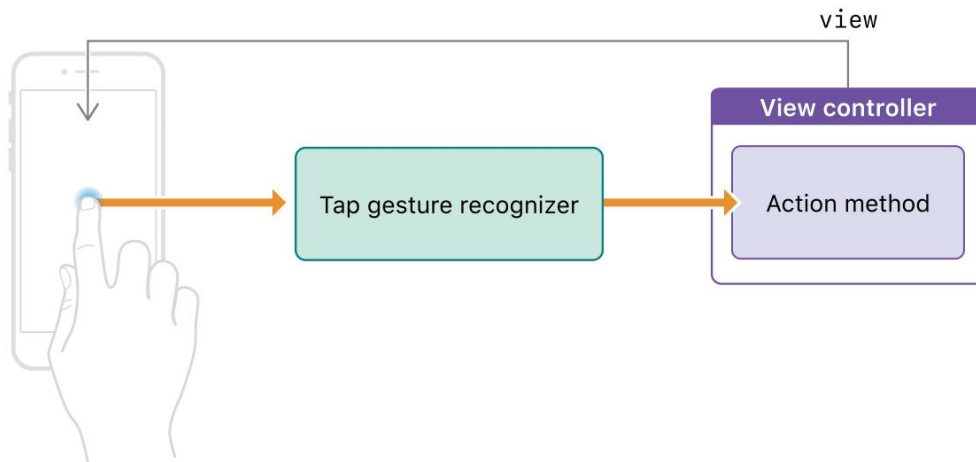
checkmark ✓



textLabel  
detailTextLabel

# Обработка касаний

1. Вручную
2. Gesture Recognizer



# UITapGestureRecognizer



Tap



Swipe



Pan

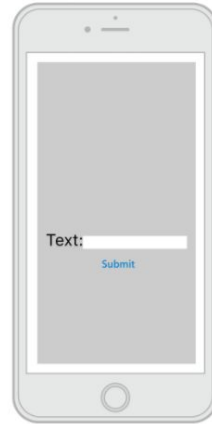


Pinch

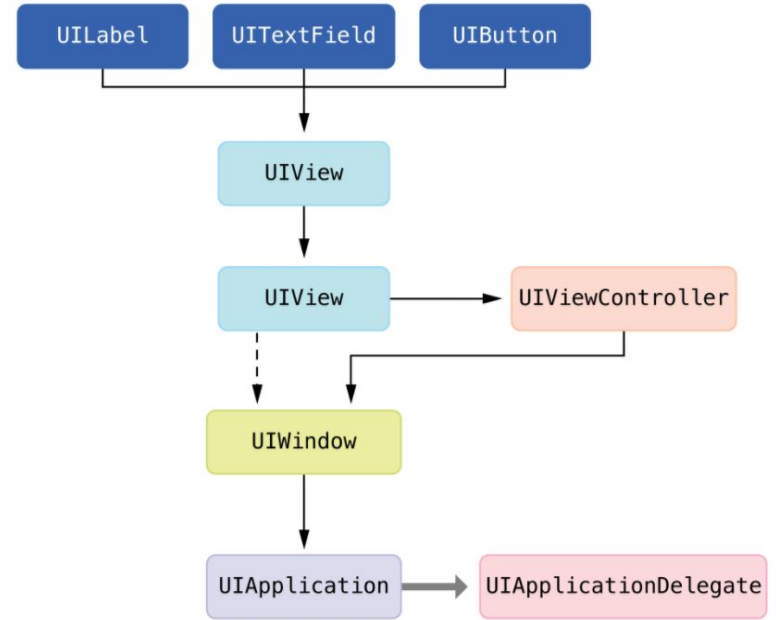


Rotation

# Responder Chain



— Next responder  
— Delegate





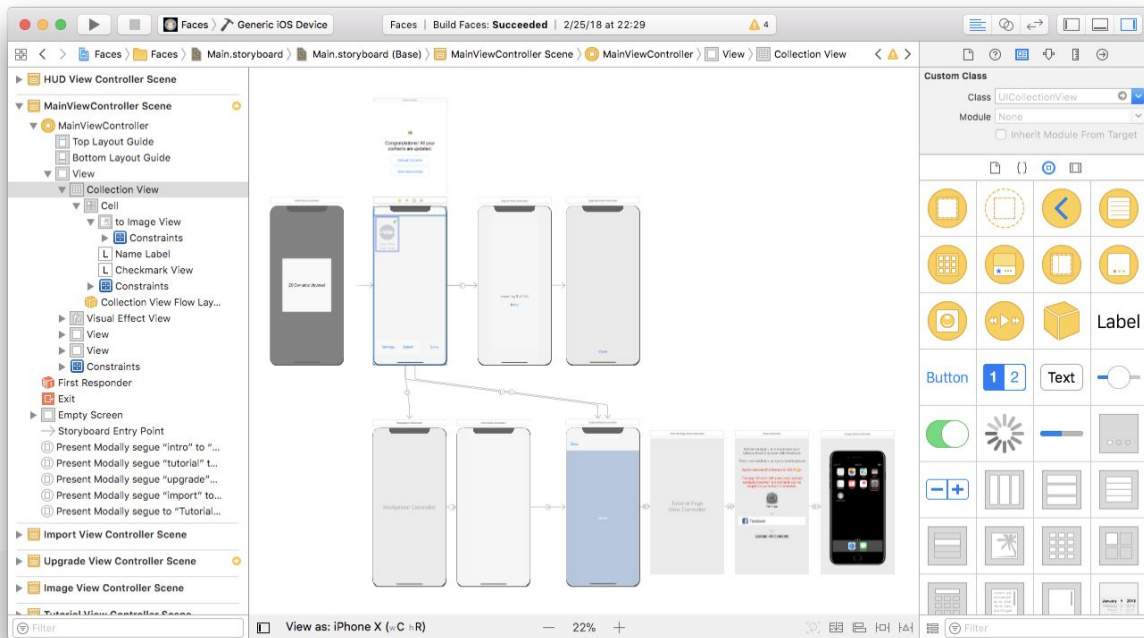
# Верстка

iOS приложений



# Способы верстки

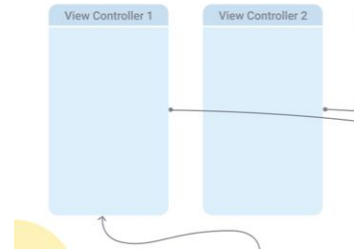
```
let redBox = UIView()
redBox.backgroundColor = .red
redBox.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(redBox)
NSLayoutConstraint.activate([
    redBox.widthAnchor.constraint(equalToConstant: 100),
    redBox.heightAnchor.constraint(equalToConstant: 100),
    redBox.centerXAnchor.constraint(equalTo: view.centerXAnchor),
    redBox.centerYAnchor.constraint(equalTo: view.centerYAnchor),
])
```



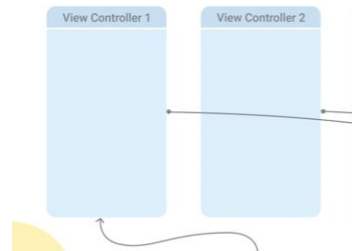


# Преимущества storyboards

1. Визуализация пользовательского интерфейса и constraints
2. Возможность мгновенно увидеть результат своих действий
3. Предварительный просмотр всех размеров экрана одновременно
4. Удаление шаблонного UI кода



# Недостатки storyboards



1. Большой вес файлов;
2. Загроможденность экрана при больших проектах
3. Сложно делать анимации
4. Возможность переиспользования
5. Возможность изменить IBOutlet
6. Нельзя использовать кастомные инициализаторы для контроллеров, созданных в Storyboard
7. Конфликт слияния при работе в команде, поскольку все файлы визуального представления – это XML файлы
8. Нельзя сменить тип специальных UIViewControllers
9. Менее гибкие, в отличие от кода


```
translatesAutoresizingMaskIntoConstraints = false
subview.translatesAutoresizingMaskIntoConstraints = false
addSubview(subview)
subview.activate(anchors: anchors, relativeTo: self)
```

## Преимущества кода

1. Возможность создать/изменить/удалить свои элементы и переопределять их в любом месте программы
2. Более гибкий способ, можно сделать любые анимации за счет непосредственного доступа к UIKit;
3. Не создается дополнительных файлов и уменьшается вес программы
4. Множество различных библиотек

## Недостатки кода

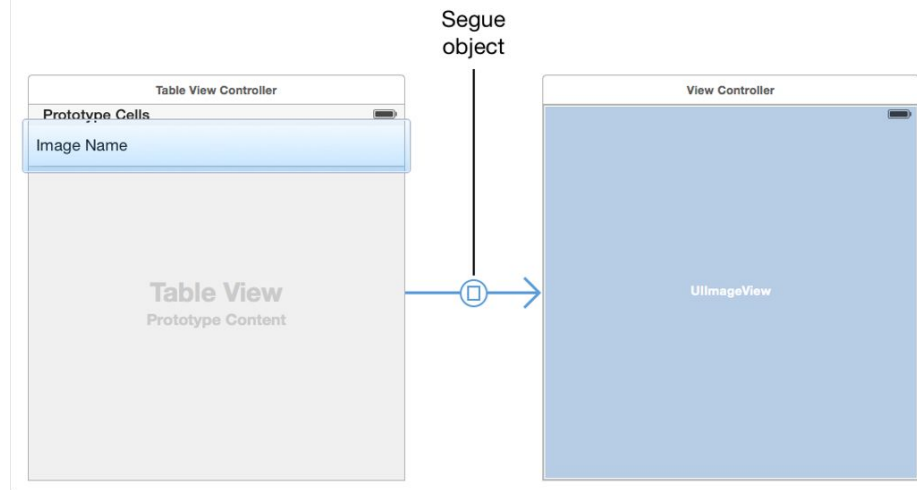
1. Сложнее разобраться в принципах работы
2. Императивный подход



# Передача данных между экранами

iOS приложений

# Segues



```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    let nickName = textField.text ?? "" // забираем текст из textField  
    let destinationVC = segue.destination as! SecondViewController // задаем  
    SecondViewController, как следующий контроллер  
    destinationVC.nickName = nickName // передаем на второй экран в переменную nickName  
}
```

Где var nickName: String = "" – определен во втором выюконтроллере

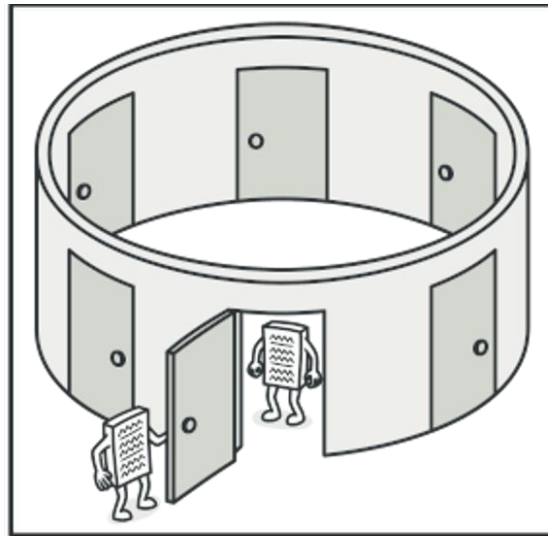
# Singleton Design Pattern

```
class Settings {
```

```
//Создаем экземпляр класса и гарантируем его уникальность  
static let instance = Settings()
```

```
private init() {  
}
```

```
//Создаем глобальную переменную  
var isSoundOn = true  
}
```



Теперь для обращения к этой переменной из любого места проекта достаточно написать:

```
Settings.instance.isSoundOn
```

# Closures



**В первом контроллере, в которым хотим подставить значения из второго напишем следующее:**

```
class Profile: UIViewController {  
    //добавим лейбл, на котором будет отображаться строка  
    var petLabel: UILabel!  
    func setPet(_ pet: String) {  
        //перезапишем значение строки нашего лейбла, которое получим при вызове функции  
        petLabel.text = pet  
    }  
}
```

**Во втором контроллере вызовем функцию setPet из первого контроллера, в качестве действия по закрытию экрана**

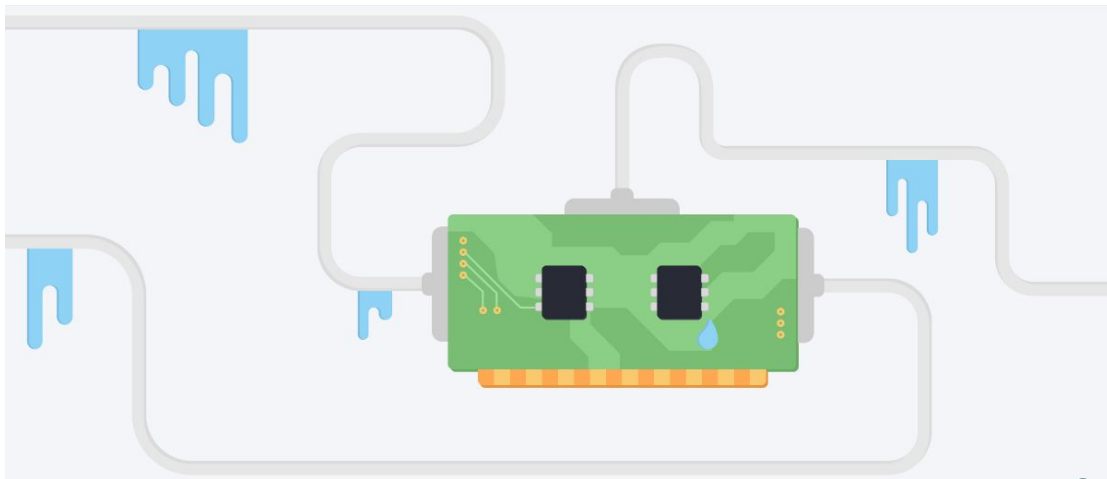
```
@IBAction func didPressOkButton(_ sender: Any) { //данная функция вызывается при закрытии экрана  
    if let vc = presentingViewController as? Profile {  
        //Прежде чем закрыть второй контроллер – передадим в первый контроллер строку из переменной favoritePet  
        dismiss(animated: true, completion: {  
            vc.setPet(self.favoritePet)  
        })  
    }  
}
```



# Weak self

```
button.touchUpInside = { [weak self] in  
    self?.output.send(.buttonTap)  
}
```

- Strong capturing
- Weak capturing
- Unowned capturing





# Мобильные приложения

РИП. Часть 2



# Архитектура

iOS приложений

Зачем?



Architecture

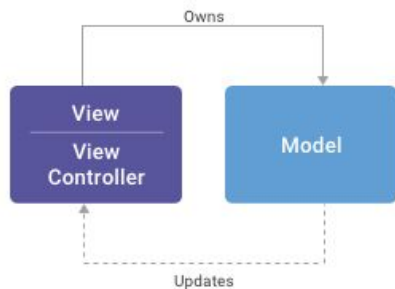




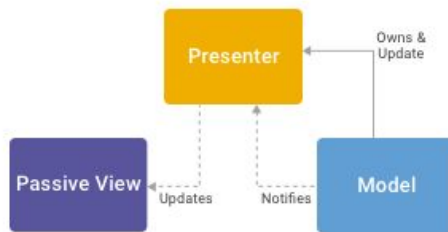
## Признаки хорошей архитектуры:

- сбалансированное распределение обязанностей между сущностями с жесткими ролями
- тестируемость
- простота использования и низкая стоимость обслуживания.

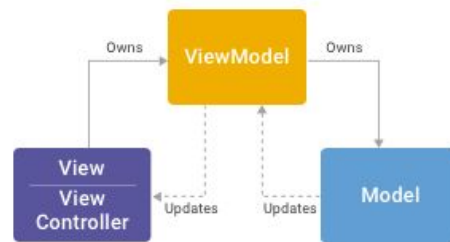
# Разновидности MVx архитектур



MVC

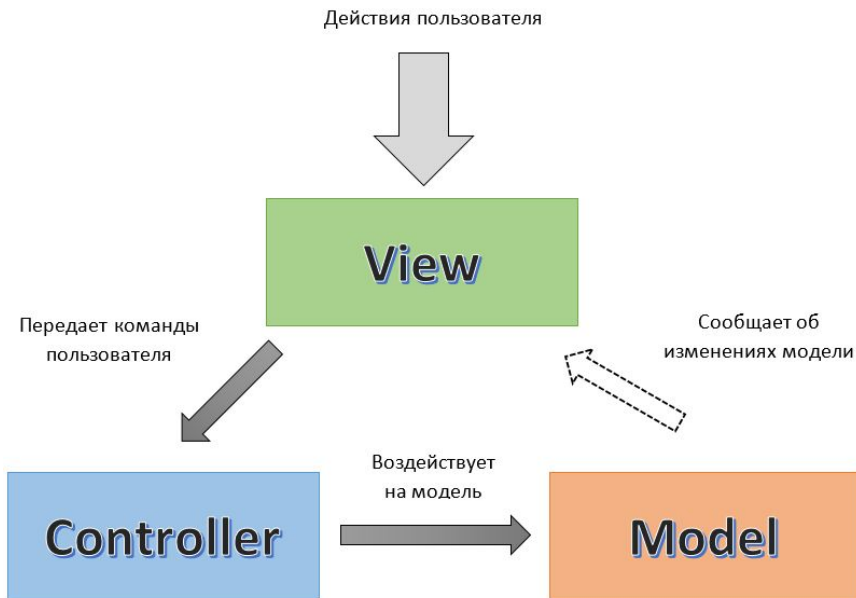


MVP

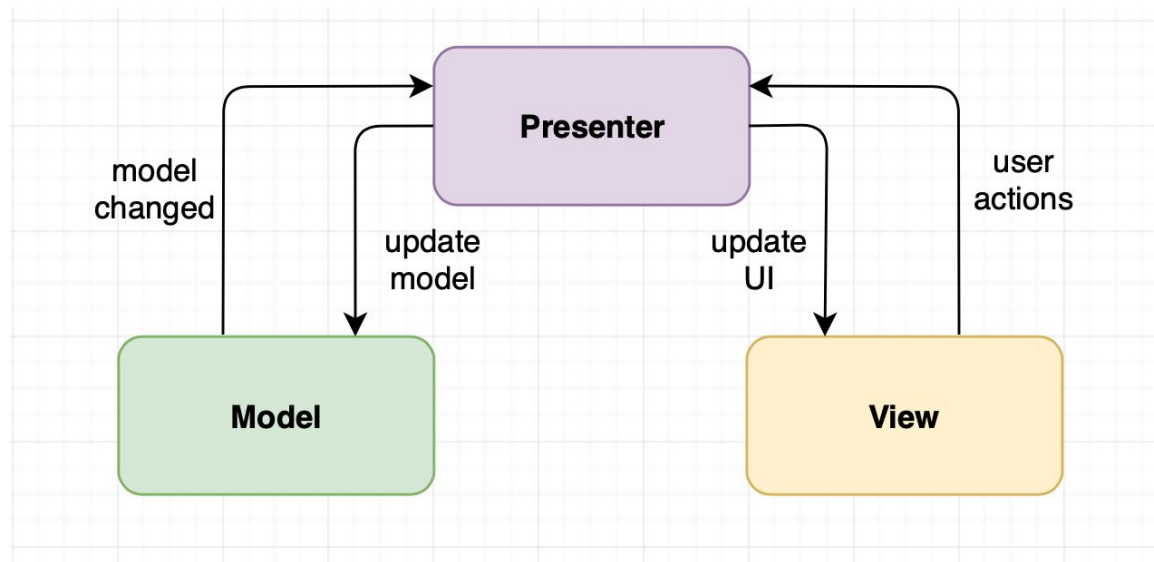


MVVM

# MVC



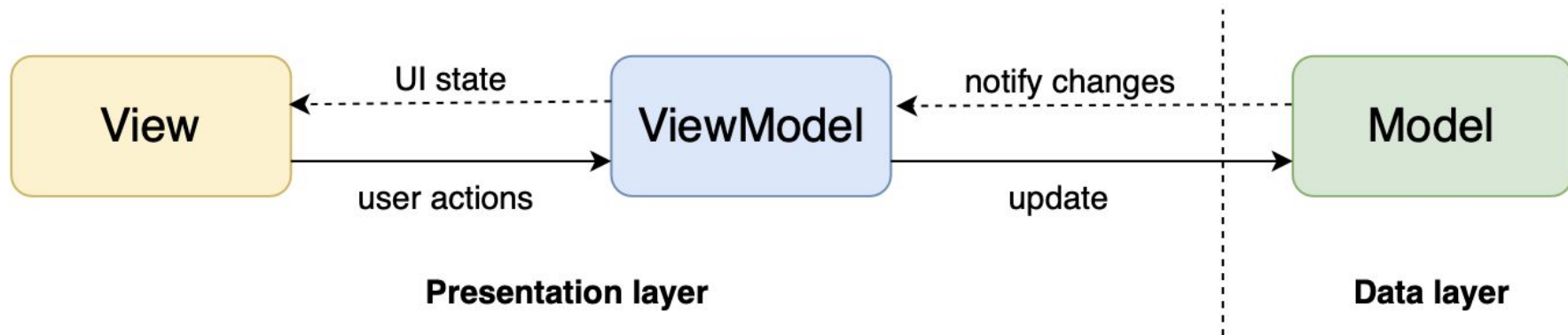
# MVP







# MVVM

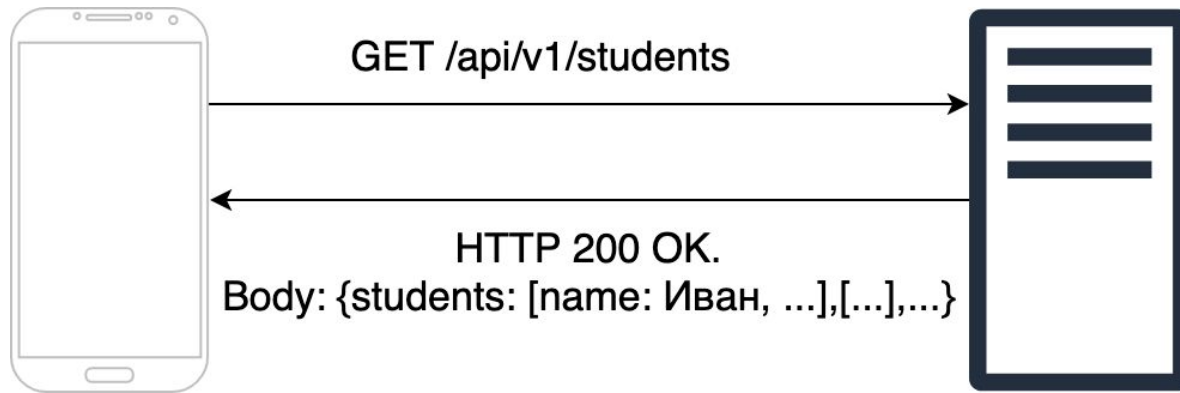




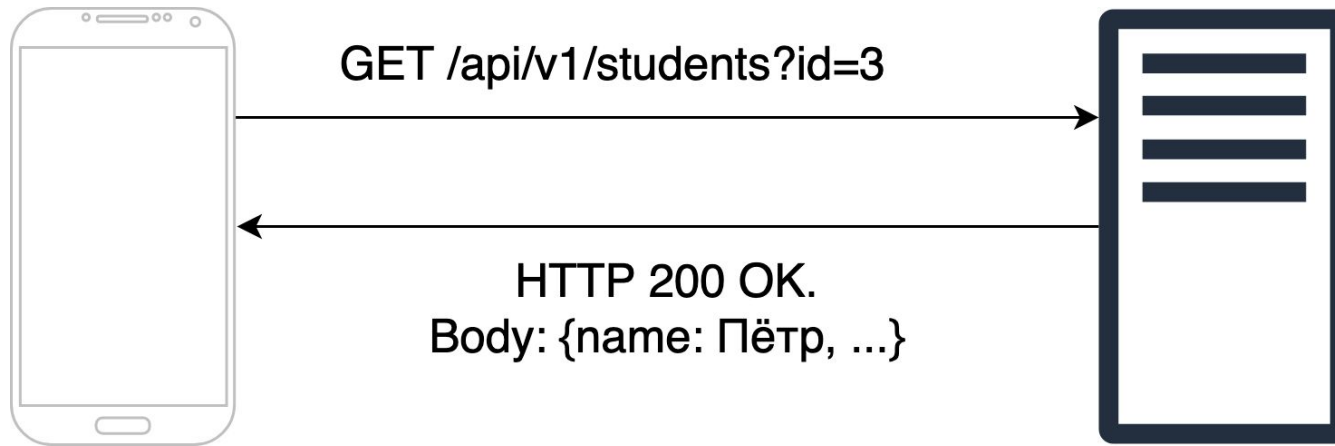
# Сетевое взаимодействие

iOS-приложений

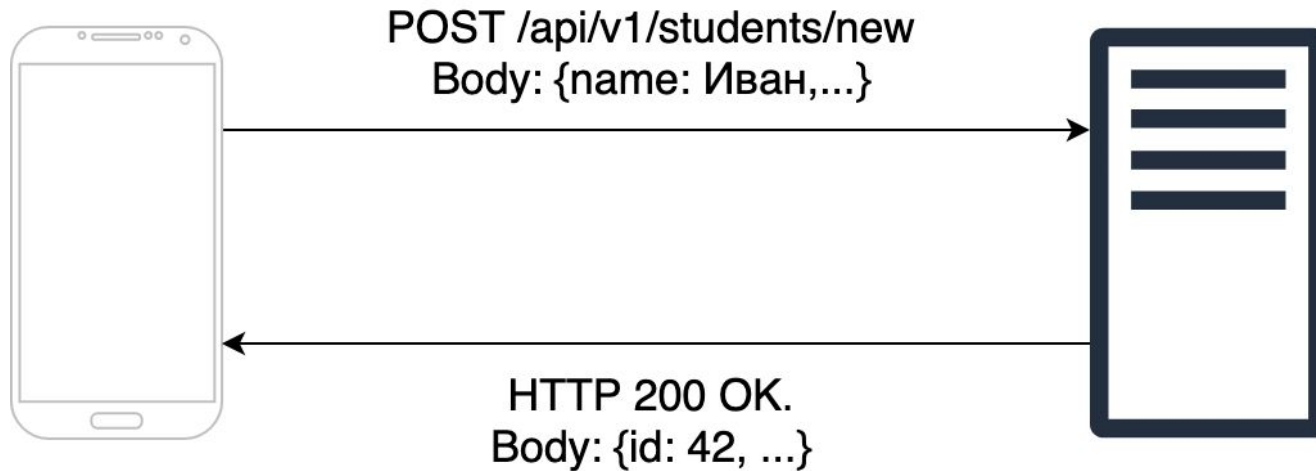
## Сетевое взаимодействие. Получение данных



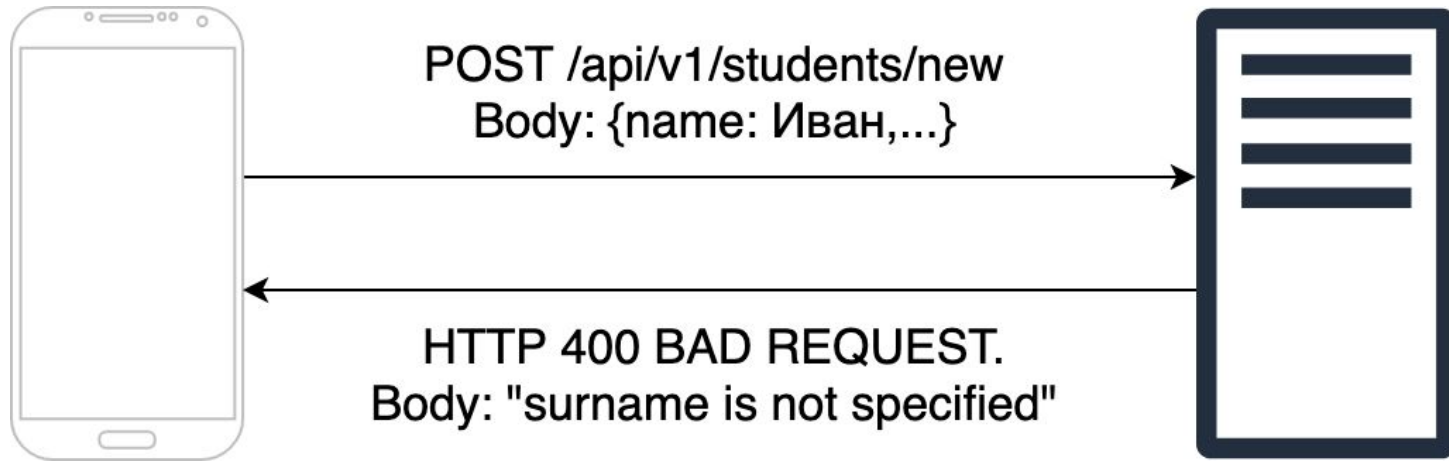
## Сетевое взаимодействие. Получение данных



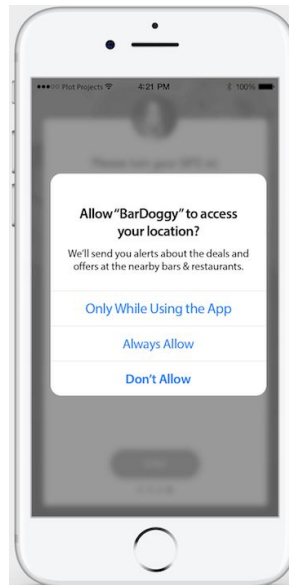
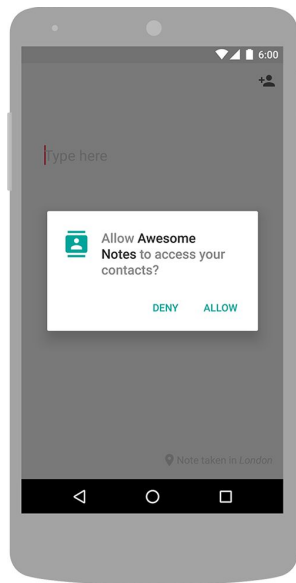
## Сетевое взаимодействие. Отправка данных на сервер



## Сетевое взаимодействие. Отправка данных на сервер



# Permissions. Разрешения





# Благодарности

Благодарим студентов ИУ5:

Курганову Александру (iOS),  
Федорову Антонину (iOS),  
Румянцева Олега (Android),  
Низовцева Романа (Android)

за помощь в подготовке материалов