

# **Разработка приложений на iOS**

Сардарян Артур, Тихонов Павел, Бабков Андрей,  
Носов Павел, Кулаков Константин



mail.ru  
group

Блоги Люди Программа Вакансии Расписание

Екатерина Черкасова

Открыт приём заявок!

сб, 4 сентября  
Занятый нет

вс, 5 сентября  
12:00  
Разработка приложени...

пн, 6 сентября  
---

вт, 7 сентября  
Занятый нет

ср, 8 сентября  
Занятый нет

чт, 9 сентября  
Занятый нет

пт, 10 сентября  
Занятый нет

17 апреля встречаемся на ИТ-квизе!

Общие вопросы

Изменить Удалить

Апрель – довольно скучный месяц: зимняя сессия уже прошла, следующие экзамены не скоро, а погода пока оставляет желать лучшего. Но мы не дадим вам зачахнуть в период апрельского ничегонеделания!

Приглашаем сразиться в командной интеллектуальной битве на всеми любимую тему технологий. Проверите свои знания, отлично проведете время и, возможно, выигрываете ценные призы.

Когда и как пройдет квиз?

Встречаемся **17 апреля** в 12:00 по московскому времени. Продолжительность игры – 1 час 30 минут.

Игра проходит в онлайн: ты сидишь у себя дома, на одной вкладке – трансляция игры, на другой – твоя команда на связи. Или собираетесь командой очно, но не забывайте про меры предосторожности.

Как принять участие?

- Соберите команду из студентов образовательных проектов и назначьте капитана. Найти людей в команду ты можешь в чате своей дисциплины. Команда должна состоять из 2-6 человек.
  - Если у тебя нет команды, то заполни эту [форму](#) и мы подберем её тебе.
- Капитан регистрируется по [ссылке](#) до 16 апреля включительно. Важно указать действующую почту.
- В день игры (за час до начала квиза) на электронную почту капитана придут ссылки на трансляцию и форму для отправки ответов.

А что по призам?

Победителей и призеров квиза ждут призы от Mail.ru Group, а какие именно узнаешь на самой трансляции!

Более подробная инструкция придет на почту капитанам команд. Если остались вопросы, то пиши их в комментарии к этому посту.

Победители и призёры:

;) DROP TABLE Students;-- Best Practices	25Top 1 25Top 1
---	--------------------

Разработка приложений на iOS

Смешанное занятие 1

Отметьтесь, что вы пришли на занятие. Так вы улучшите свою посещаемость и вас увидят преподаватель в своем "Курнале посещений".

Отметиться

Оставьте отзыв о занятии и мы сможем улучшить учебный процесс.

Оставить отзыв

Прямой эфир

Мон Все

Ярослав Янин 12 дней назад  
17 апреля встречаемся на ИТ-квизе! 1

Екатерина Черкасова 24 дня назад  
Автоматизированное тестирование - Стань ментором в новом семестре! 0

Отметьтесь на портале

---

# Layout

1. Базовые элементы UIKit

---

# Layout

1. Базовые элементы UIKit
2. Подходы для создания интерфейсов

---

# Layout

1. Базовые элементы UIKit
2. Подходы для создания интерфейсов
3. ResponderChain

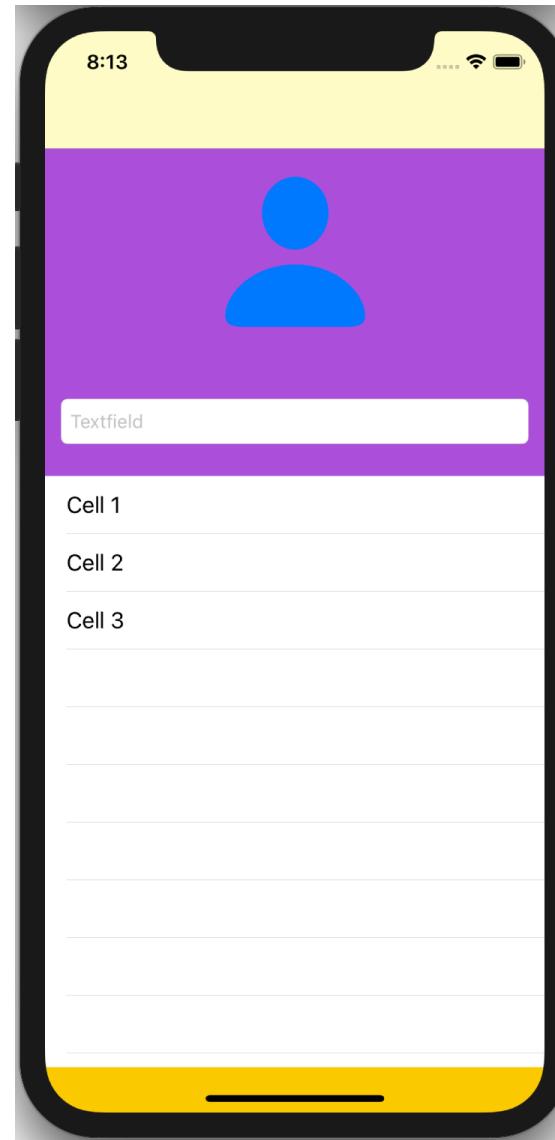
---

# Layout

1. Базовые элементы UIKit
2. Подходы для создания интерфейсов
3. ResponderChain
4. Распознавание жестов

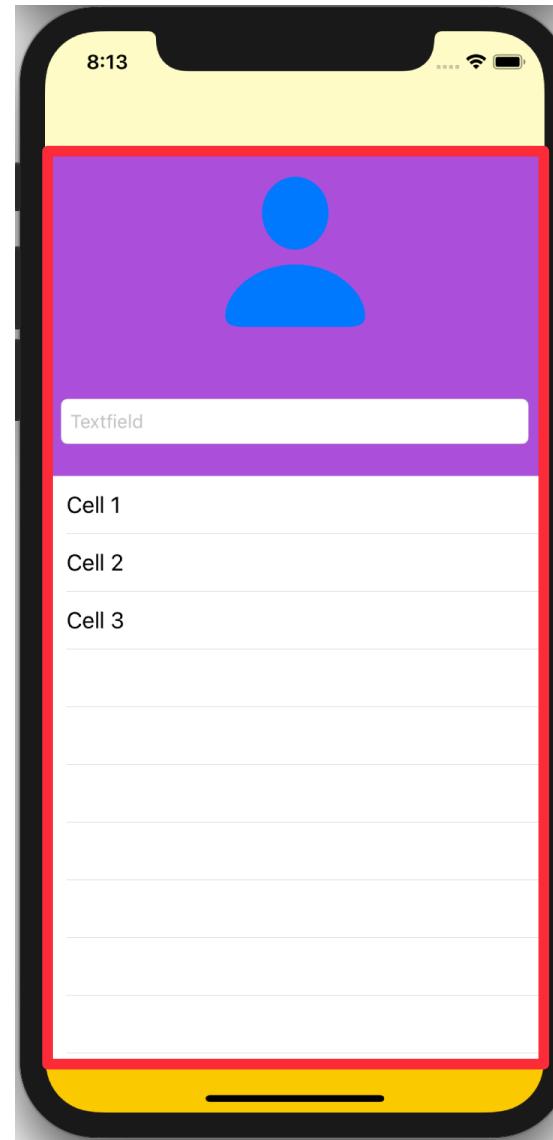
---

# Интерфейс приложения



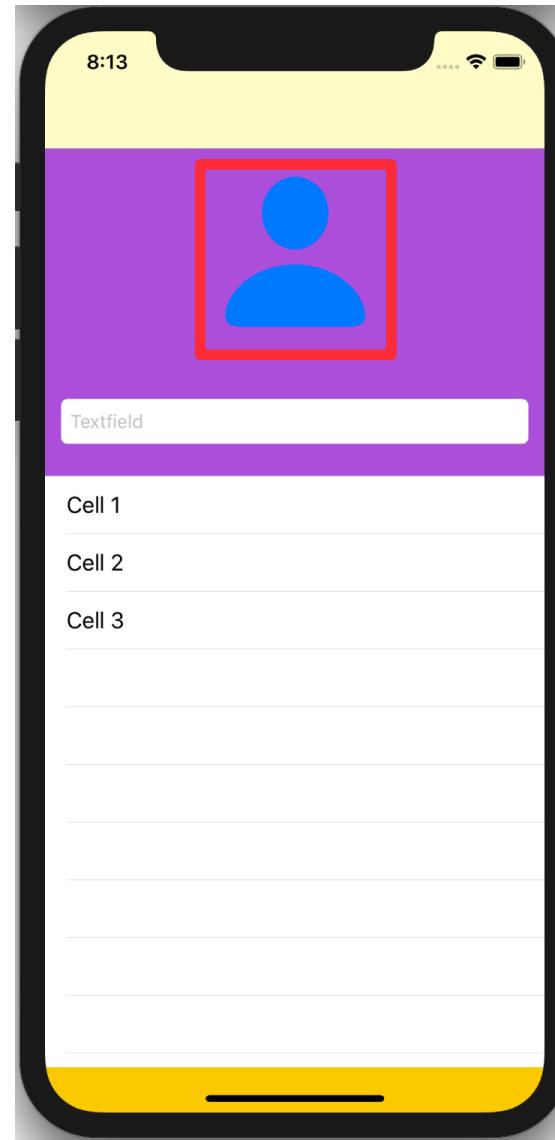
---

# Интерфейс приложения



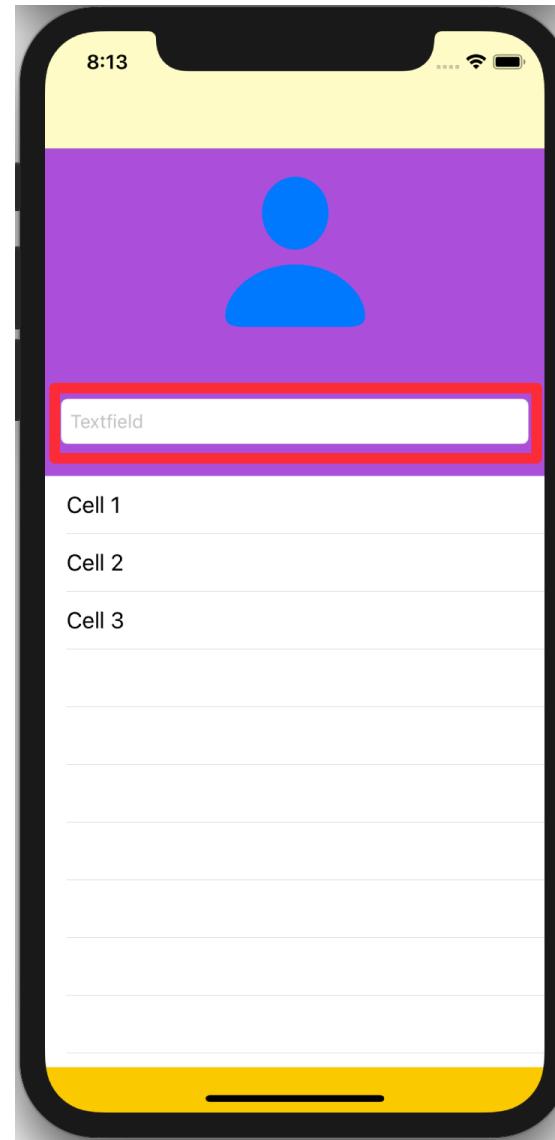
---

# Интерфейс приложения



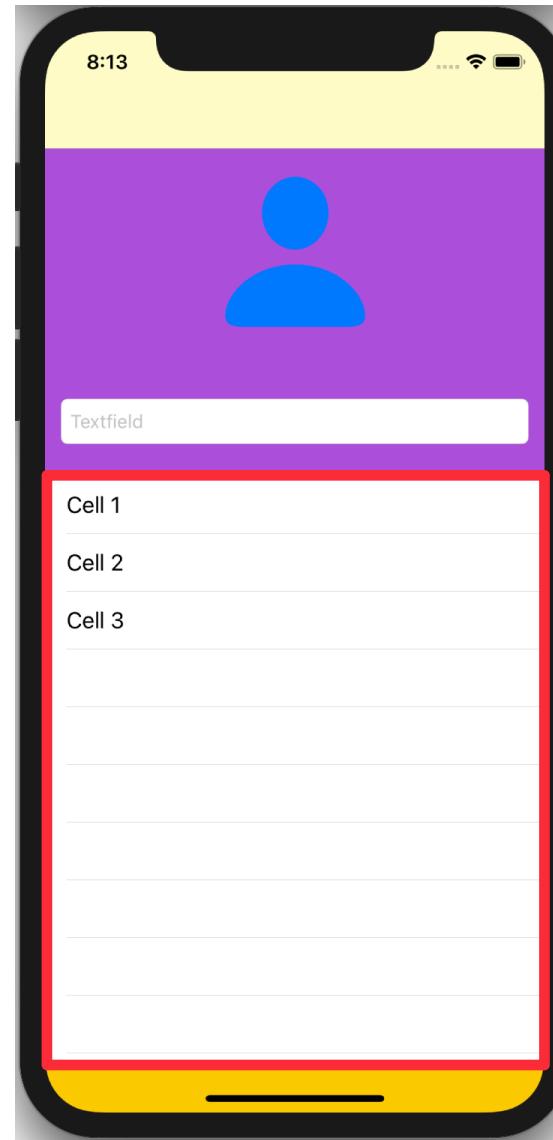
---

# Интерфейс приложения



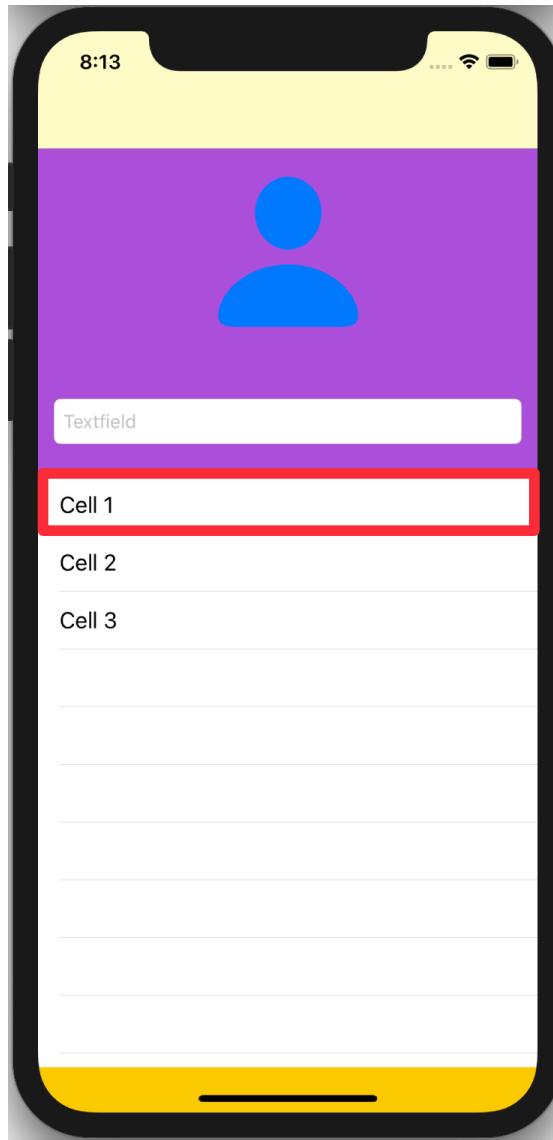
---

# Интерфейс приложения



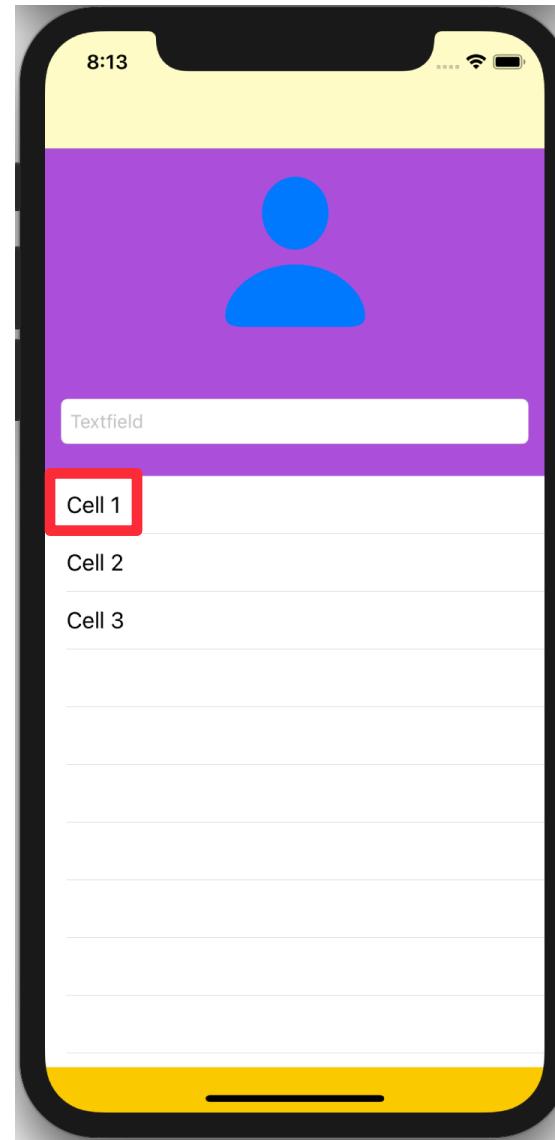
---

# Интерфейс приложения

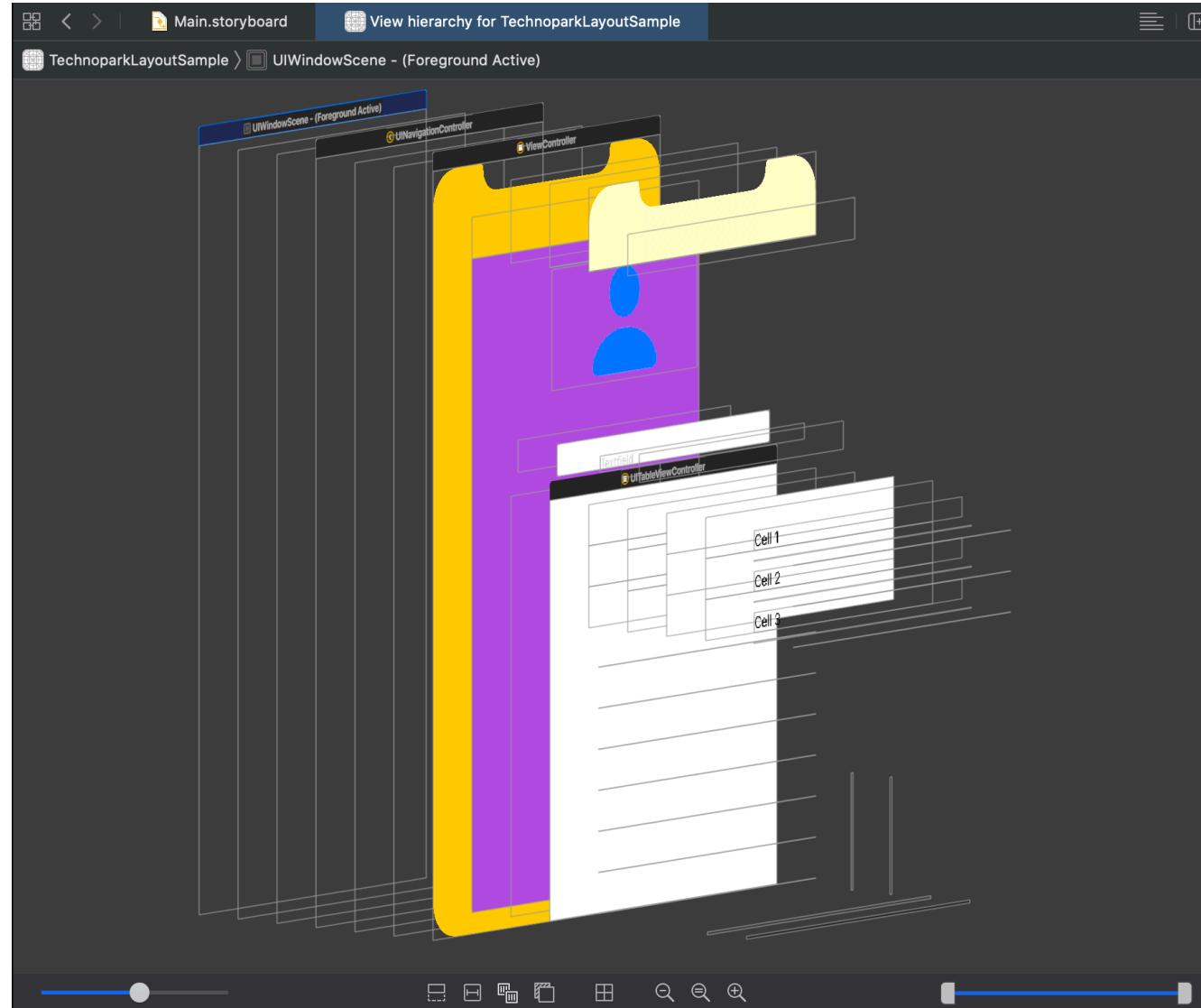


---

# Интерфейс приложения



# Интерфейс приложения (в разрезе)



# UIView

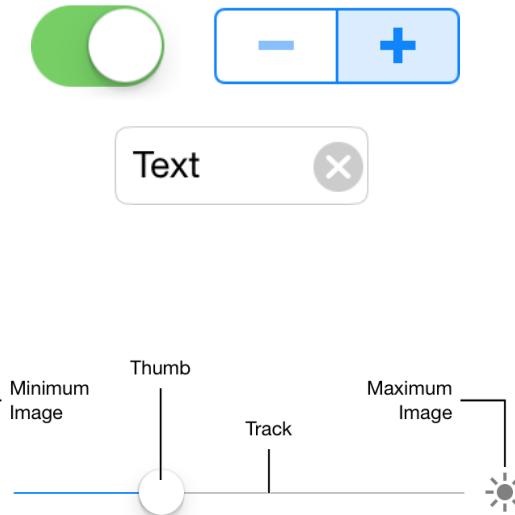
- Базовый класс UIKit («кирпичи» из которых строится интерфейс)
- Потомок UIResponder'a
  - обрабатывает события:
    - касания (touches)
    - движения (motions), например, встрихивание
- Отображает элементы интерфейса
- Может настраивать размеры и позиции своих subview
- Могут добавлять распознавателей жестов (gesture recognizers) для обработки типичных жестов

# UIView (потомки)

- UIControl
  - UIButton, UITextField и др.
- UIWindow (редко нужен в iOS, но может пригодиться, если нужно что-то показать поверх status bar'a)
- UILabel
- списки (UIScrollView, UITableView, UICollectionView)
- UITextView
- UIImageView
- MKMapView
- WKWebView

# UIControl

- UIControl
  - UIButton
  - UITextField
  - UISwitch
  - UISegmentedControl
  - UISlider
  - UIProgress
- Добавляет к UIView
  - механизм target/action
  - протокол (UITextFieldDelegate и пр.)



<https://developer.apple.com/reference/uikit/uicontrol>  
<https://developer.apple.com/reference/uikit/uislider>

# UIView

- один superview
- много subview
- addSubview: (у родительской view)
  - добавляет к нашему view subview
- removeFromSuperview
  - удаляет наш view из его superview
- анимации
  - ```
@available(iOS 4.0, *)
open class func animate(withDuration duration: TimeInterval, delay: TimeInterval,
options: UIView.AnimationOptions = [], animations: @escaping () -> Void,
completion: ((Bool) -> Void)? = nil)
```

# UIView (основные свойства)

- CGFloat alpha
  - прозрачность (от 0 до 1)
- Bool isOpaque (непрозрачный)
  - true/false
- Bool isHidden (невидимый)
- Bool userInteractionEnabled (отключенный)
- Bool clipsToBounds (обрезать по границам)
- Bool translatesAutoresizingMaskIntoConstraints  
(при использовании auto layout кодом)

# UIView (bounds / frame)

```
@property(nonatomic) CGRect frame;
```

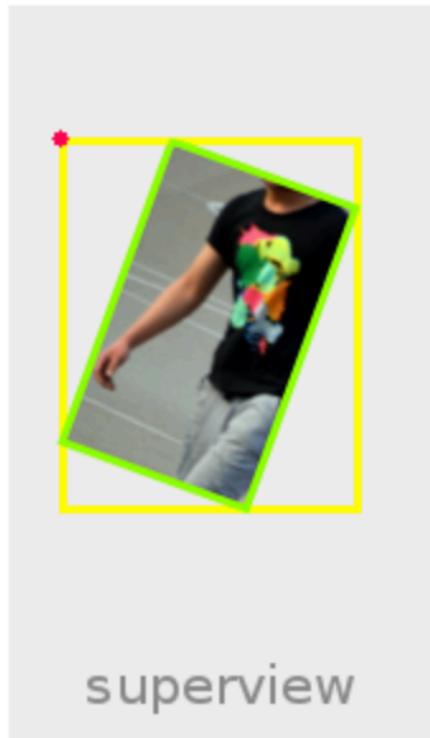
```
@property(nonatomic) CGRect bounds;
```

## **UIView (bounds / frame)**

- bounds – положение и размер в собственной системе координат
- frame – положение и размер в системе координат родительской вью
- Если нужно спозиционировать view, лучше менять center, а frame пусть считается сам

# UIView (frame / bounds)

Frame



Bounds



---

# UIViewController

??  
.



---

# **UIViewController**

- реагирует на взаимодействие с view

---

# **UIViewController**

- реагирует на взаимодействие с view
- адаптирует размеры view под общий интерфейс

# **UIViewController**

- реагирует на взаимодействие с view
- адаптирует размеры view под общий интерфейс
- взаимодействует с другими объектами и viewControllers в приложении

# **UIViewController**

- реагирует на взаимодействие с view
- адаптирует размеры view под общий интерфейс
- взаимодействует с другими объектами и viewControllers в приложении
- обновляет контент на экране в результате изменения данных

# Распознавание жестов

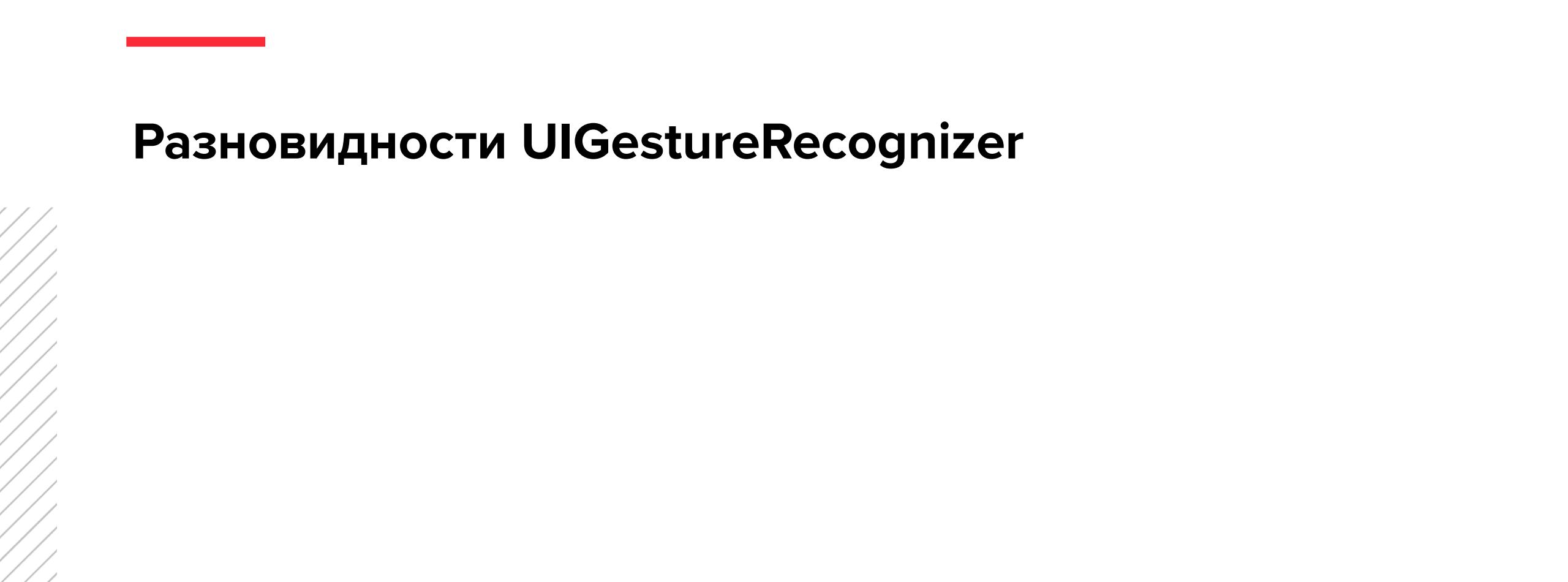
- Вручную
  - touchesBegan(\_ touches: Set<UITouch>, with event: UIEvent?)touchesEnded:withEvent:
  - touchesMoved...
  - touchesCancelled...
- Использование UIGestureRecognizer (точнее, его потомков)
  - кодом
  - с помощью Interface Builder

# Пример UITapGestureRecognizer

```
.....  
let gestureRecognizer =  
    UIGestureRecognizer(target: self, action: #selector(viewTapped(_:)))  
view.addGestureRecognizer(gestureRecognizer)  
.....  
@objc func viewTapped(_ sender: UIGestureRecognizer) {  
    print("viewTapped: \(gestureRecognizer.view)")  
}
```

---

# Разновидности **UIGestureRecognizer**



# Разновидности **UIGestureRecognizer**

- tap
- swipe
- pan
- long press
- pinch
- rotation

# Кастомные UIView

- Кастомизация
  - `init(frame:)` (могут отличаться у наследников `UIView`, например, у `UITableViewCell`)
    - конструктор используется только для создания view кодом
  - `init(coder:)`
    - для view, созданного через Interface Builder
  - `draw(_:)`
    - если нужно во view что-то нарисовать

# Кастомные UIView, метод drawRect(\_:)

- В UIView можно рисовать:
  - UIBezierPath
    - moveToPoint:,
    - addLineToPoint
  - Core Graphics
    - контекст CGGetCurrentContext()
    - функции для рисования
- Используется CPU, а не GPU

```
let path = UIBezierPath()
```

```
override func draw(_ rect: CGRect) {
    // получаем ссылку на контекст
    guard let context = UIGraphicsGetCurrentContext() else {
        return
    }

    // очищаем контекст
    context.clear(rect)

    // выставляем цвет
    context.setFillColor(UIColor.yellow.cgColor)

    // заполняем рект
    context.fill(rect)
}
```

```
override func draw(_ rect: CGRect) {
    let origin: CGPoint = .init(x: 50, y: 50)
    let size: CGSize = .init(width: 200, height: 400)
    let rect = CGRect(origin: origin, size: size)

    UIColor.green.setFill()
    let path = UIBezierPath(roundedRect: rect, cornerRadius: 10)
    path.fill()
}
```

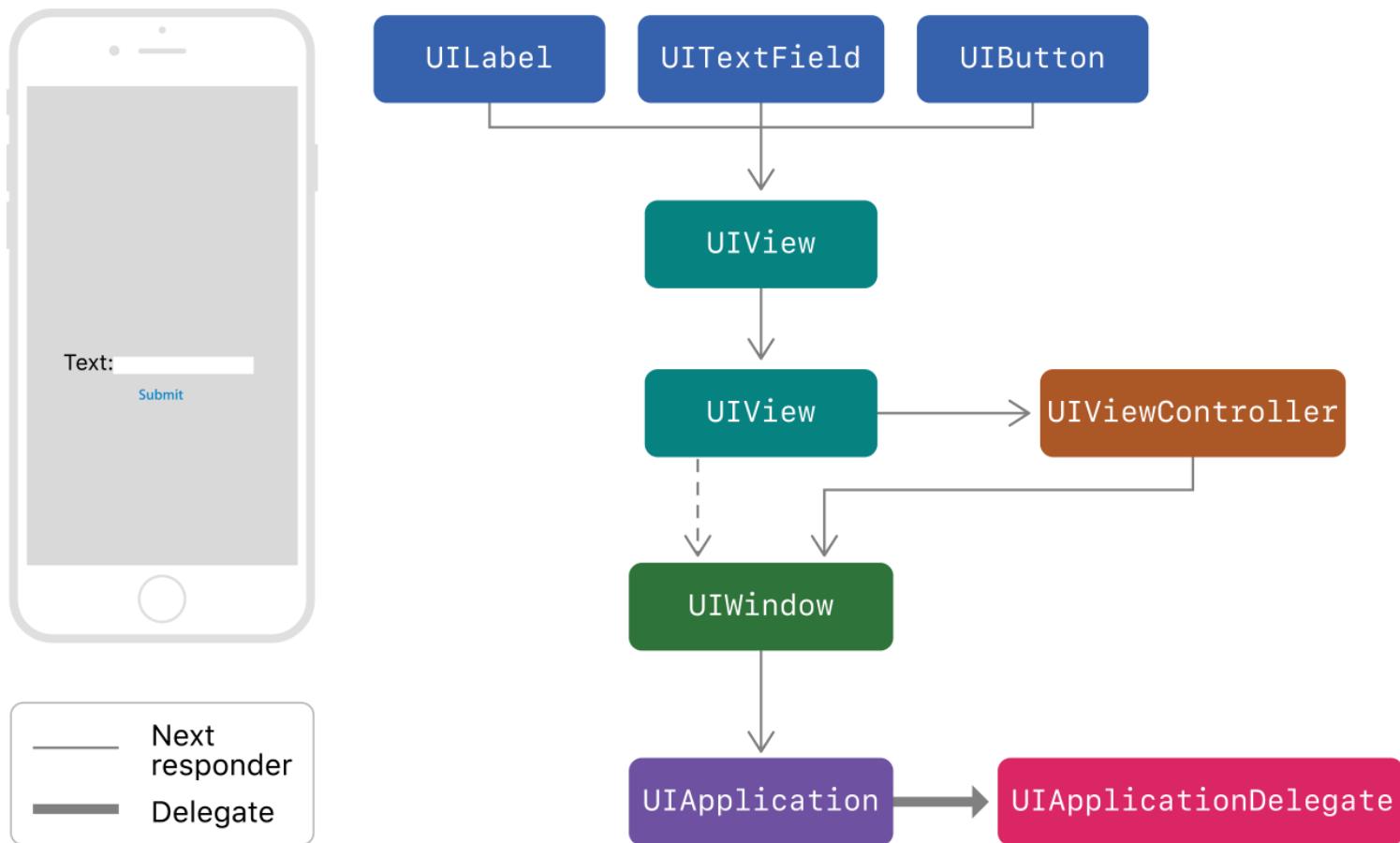
```
override func draw(_ rect: CGRect) {
    let bezierPath = UIBezierPath()

    UIColor.green.setFill()

    bezierPath.move(to: CGPoint(x: 10, y: 10))
    bezierPath.addLine(to: CGPoint(x: 160, y: 10))
    bezierPath.addLine(to: CGPoint(x: 160, y: 160))
    bezierPath.addLine(to: CGPoint(x: 10, y: 160))
    bezierPath.close()

    bezierPath.fill()
}
```

# Responder Chain



# Frame-based верстка

- До iOS 6 (2012) верстка делалась только через изменение frame у view
- С помощью autoresizingMask задавались правила изменения frame, если родительские размеры изменятся



# AutoLayout

- Система линейных неравенств
- Constraints (ограничения)
  - основной класс NSLayoutConstraint
- Можно задать разными способами:
  - Interface Builder
  - Код (NSLayoutConstraint)
  - Код (visual format language)
  - Библиотеки (PureLayout, SnapKit)

# AutoLayout (VFL)

- addConstraints — deprecated :)

```
let subview = UIView(frame: .zero)
subview.backgroundColor = .purple
subview.translatesAutoresizingMaskIntoConstraints = false

view.addSubview(subview)

let views = ["subview": subview]
let metrics = ["bottomMargin": 50]
let horizontalConstraints: [NSLayoutConstraint] = NSLayoutConstraint.constraints(withVisualFormat: "H:[subview]|",
    options: [],
    metrics: metrics,
    views: views)

view.addConstraints(horizontalConstraints)

let verticalConstraints: [NSLayoutConstraint] = NSLayoutConstraint.constraints(withVisualFormat: "V:[subview]-bottomMargin-|",
    options: [],
    metrics: metrics,
    views: views)

view.addConstraints(verticalConstraints)
```

# AutoLayout (Anchors)

```
// создаем constraints через NSLayoutConstraint-конструктор

NSLayoutConstraint(item: self.view!,
                    attribute: .top,
                    relatedBy: .equal,
                    toItem: subview,
                    attribute: .top,
                    multiplier: 1,
                    constant: 0).isActive = true

NSLayoutConstraint(item: self.view!,
                    attribute: .leading,
                    relatedBy: .equal,
                    toItem: subview,
                    attribute: .leading,
                    multiplier: 1,
                    constant: 0).isActive = true

// создаем те же constraints через Anchor-API

self.view.topAnchor.constraint(equalTo: subview.topAnchor).isActive = true
self.view.leadingAnchor.constraint(equalTo: subview.leadingAnchor).isActive = true
```

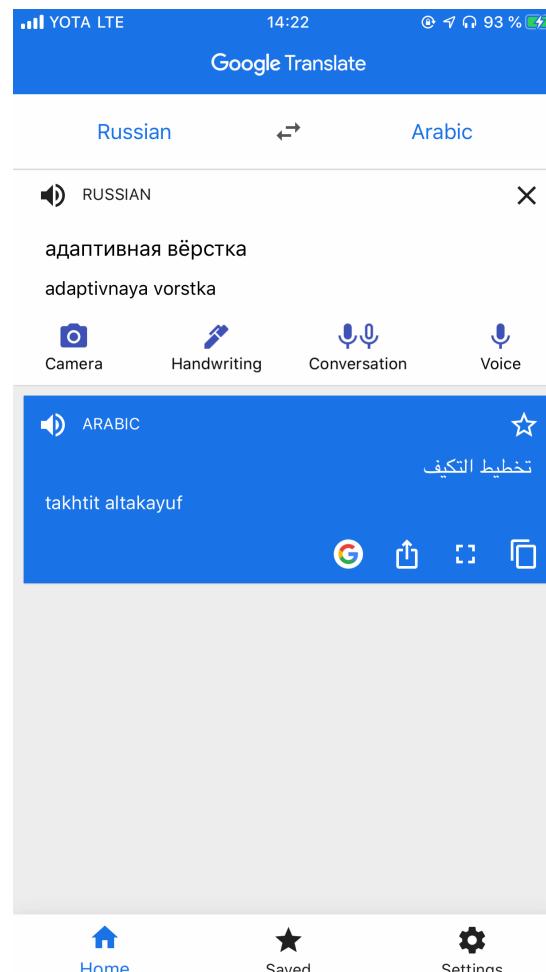
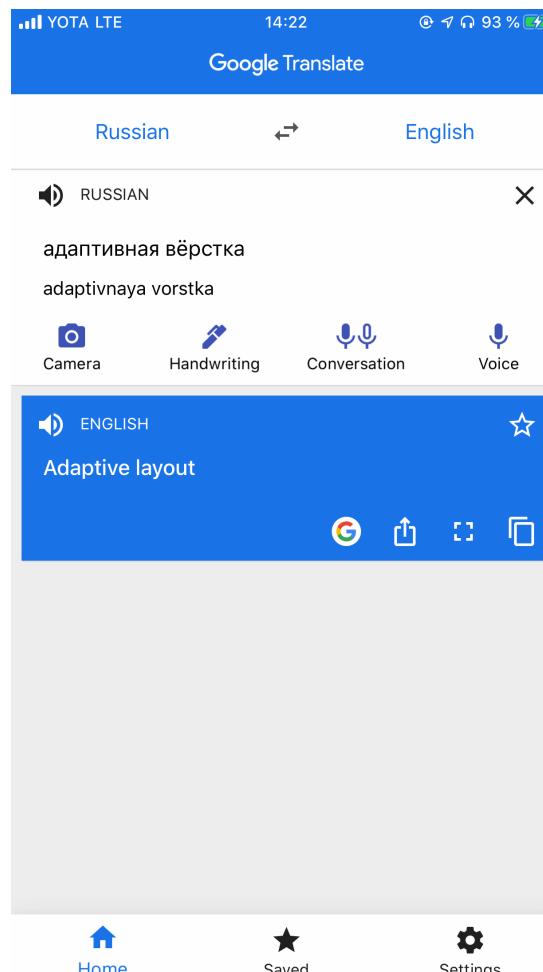
# CALayer

- Входит в состав CoreAnimation framework
- Создается UIView и UIView становится его делегатом по дефолту
- Не умеет обрабатывать события (не наследуется от UIResponder)
- Нужен для большего контроля над отображением интерфейса и анимациями
- Пригодится:
  - cornerRadius
  - borderColor, borderWidth
  - shadowPath

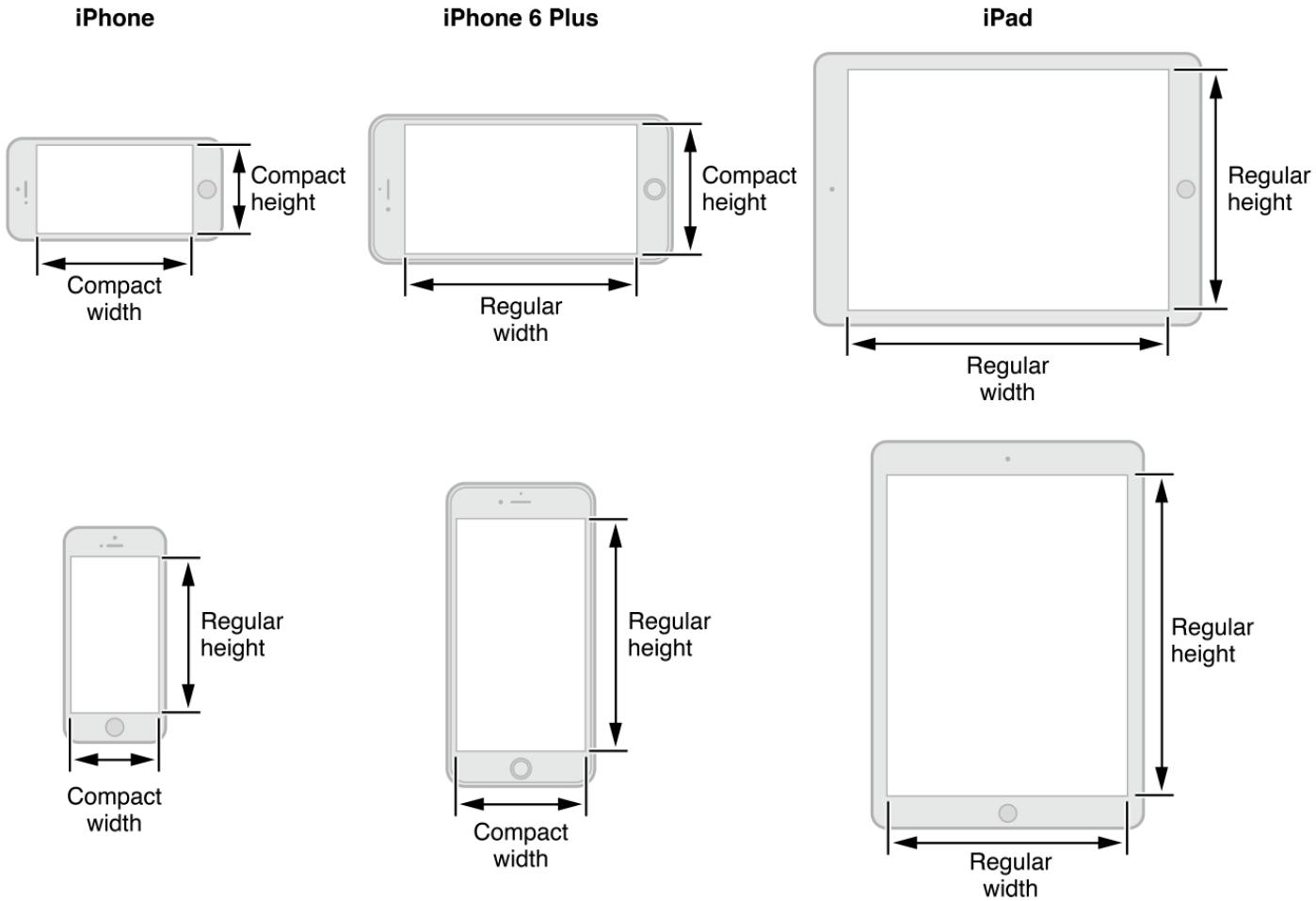
# Adaptivity

- Разные размеры экранов
- Разные ориентации (вертикальная, горизонтальная)
- Split View (iPad)
- Меняющийся текст
- Layout direction в зависимости от страны

# Пример adaptivity



# Size Classes



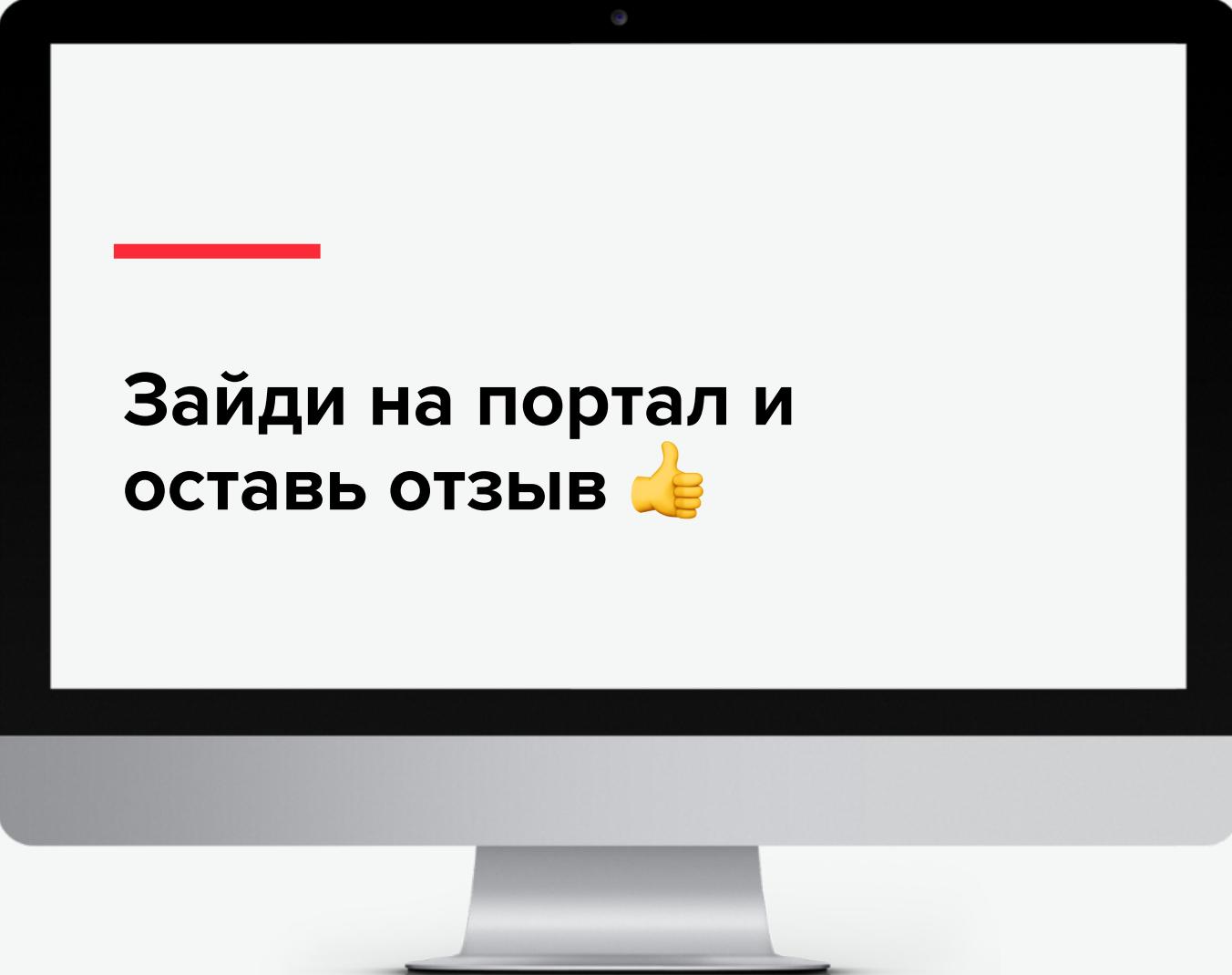
# SizeClasses

- Доступно в Storyboard
- В зависимости от size class можно:
  - добавлять/удалять constraint
  - добавлять/удалять view
  - менять атрибуты (например, шрифт)
- В коде

```
if view.traitCollection.horizontalSizeClass == .compact {  
    // какая-то логика  
}
```

**СПАСИБО  
ЗА ВНИМАНИЕ**





Зайди на портал и  
оставь отзыв 👍



# Полезные статьи

- [https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html#/apple\\_ref/doc/uid/TP40010853](https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html#/apple_ref/doc/uid/TP40010853) — Autolayout
- <https://www.raywenderlich.com/443-auto-layout-tutorial-in-ios-11-getting-started> - Autolayout
- <https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/adaptivity-and-layout/> — Adaptivity
- <https://developer.apple.com/documentation/uikit/nslayoutanchor> — Anchors
- <https://developer.apple.com/ios/human-interface-guidelines/> — Apple Human Interface Guidelines
- <https://www.objc.io/issues/3-views/custom-controls/> — о кастомных control'ах
- <http://nshipster.com/ibinspectable-ibdesignable/> — IBDesignable / IBInspectable (редко используется, для фанатов Interface Builder'a)
- <https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutolayoutPG/VisualFormatLanguage.html> — visual format language (для auto layout)
- <https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/Size-ClassSpecificLayout.html> — Size Classes
- <https://developer.apple.com/library/archive/documentation/2DDrawing/Conceptual/DrawingPrintingiOS/BezierPaths/BezierPaths.html> — рисование с помощью Bezier Paths