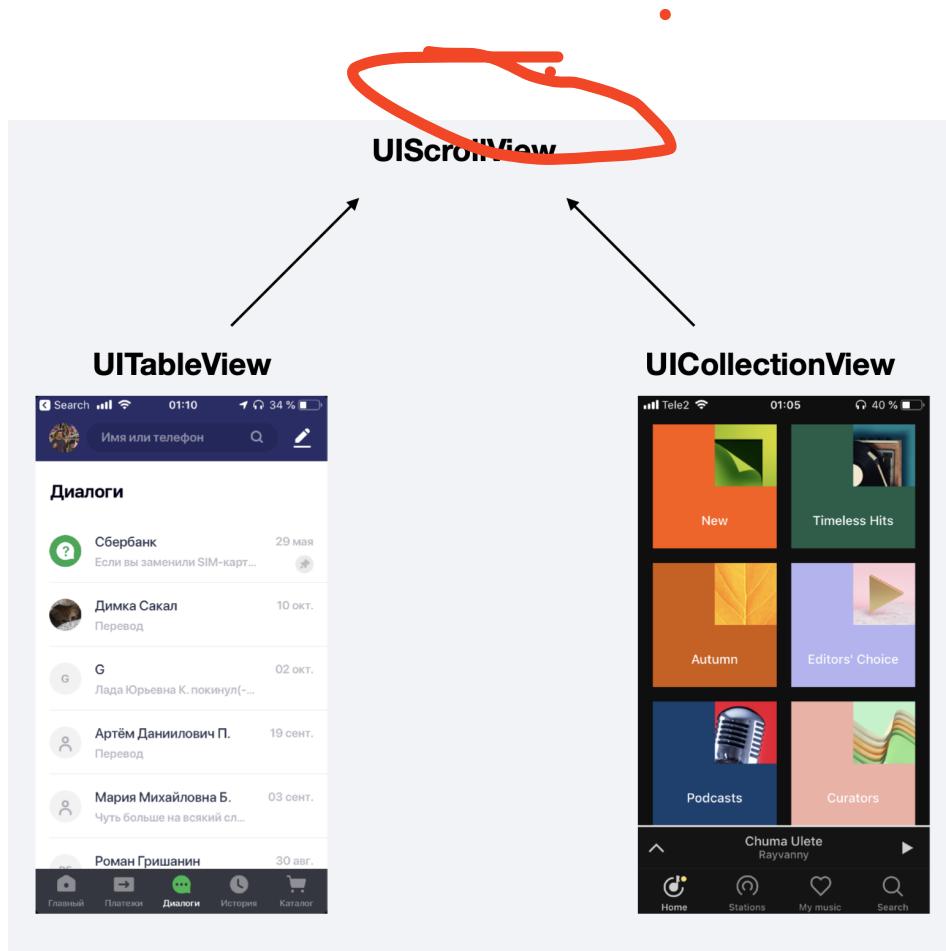


# **Привет!**

Меня зовут Лёня Серебряный

# Скролл вью, оффсеты и инсеты

## Тейбл вью



# UIScrollView



```
let scrollView = UIScrollView()  
scrollView.addSubview(imageView) // imageView size is (4096x2304)
```

```
Offset (206.0, 344.666666666667)  
imageView bounds (0.0, 0.0, 4096.0, 2304.0)●  
imageView frame (0.0, 0.0, 4096.0, 2304.0)●  
ScrollView bounds (206.0, 344.666666666667, 390.0, 844.0)  
ScrollView frame (0.0, 0.0, 390.0, 844.0)
```

```
print (scrollView.contentSize) // CGSize(width: 4096, height: 2304)
```



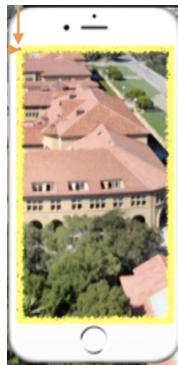
## Zoom

```
scrollView.maximumZoomScale = 2           ! Zoom меняет contentSize
scrollView.minimumZoomScale = 0.04
scrollView.delegate = self

extension ScrollViewController: UIScrollViewDelegate {
    func viewForZooming(in scrollView: UIScrollView) -> UIView? {
        return self.imageView
    }

    func scrollViewDidEndZooming(_ scrollView: UIScrollView, with view: UIView?, atScale scale: CGFloat) {
        print(scrollView.contentSize)
    }
}

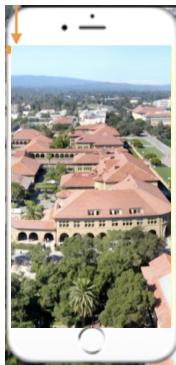
// зум из кода
scrollView.setZoomScale(0.3, animated: true)
// либо
//scrollView.zoom(to: //CGRect(origin: ..., size: ...), animated: true)
```



Было

contentSize = 4096x2304  
offset = 206x344

0.5 zoom →



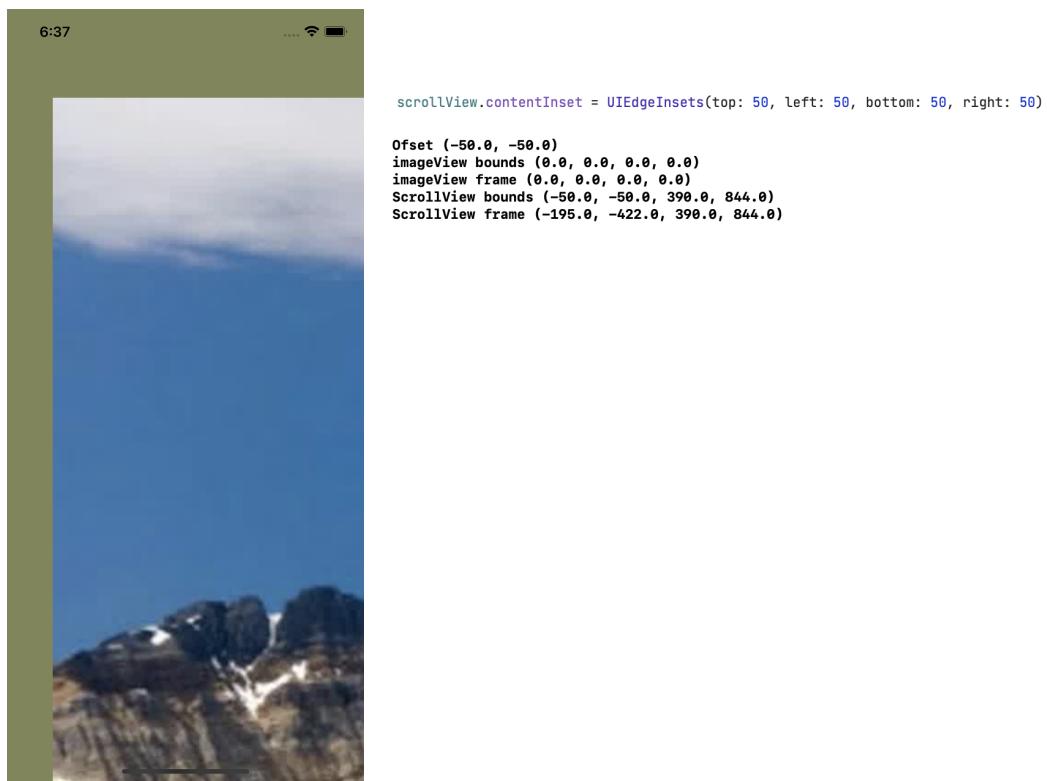
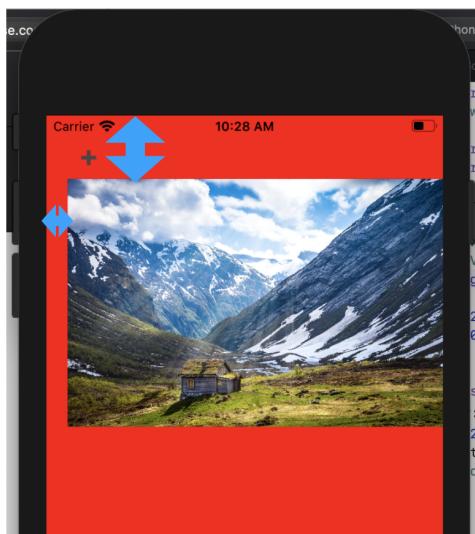
Стало

contentSize = ?  
offset = ?

# insets

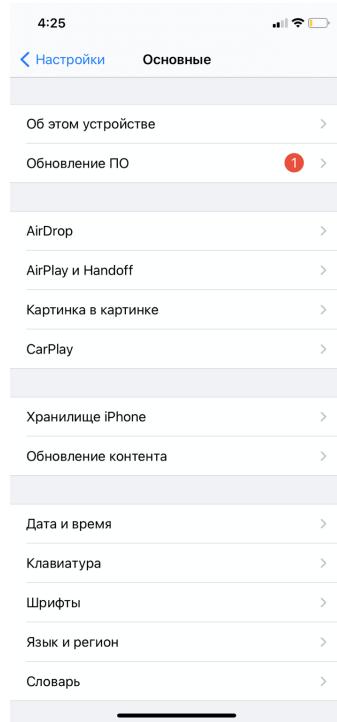
## Content insets

```
scrollView.contentInset = UIEdgeInsets(top: 60, left: 20, bottom: 20, right: 40)
```



# UITableView

Наследуется от UIScrollView



## Использование

✓ UITableView

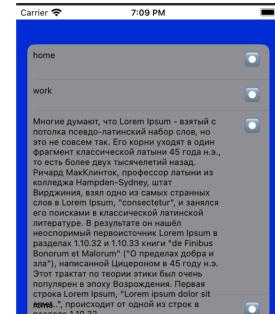
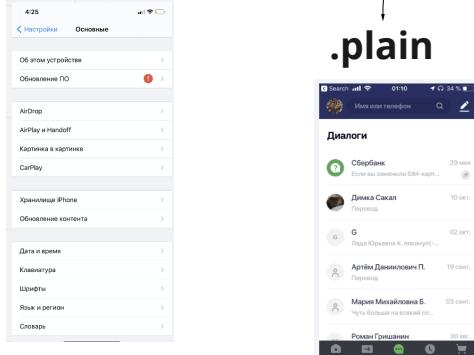
UITableViewController

## Стиль

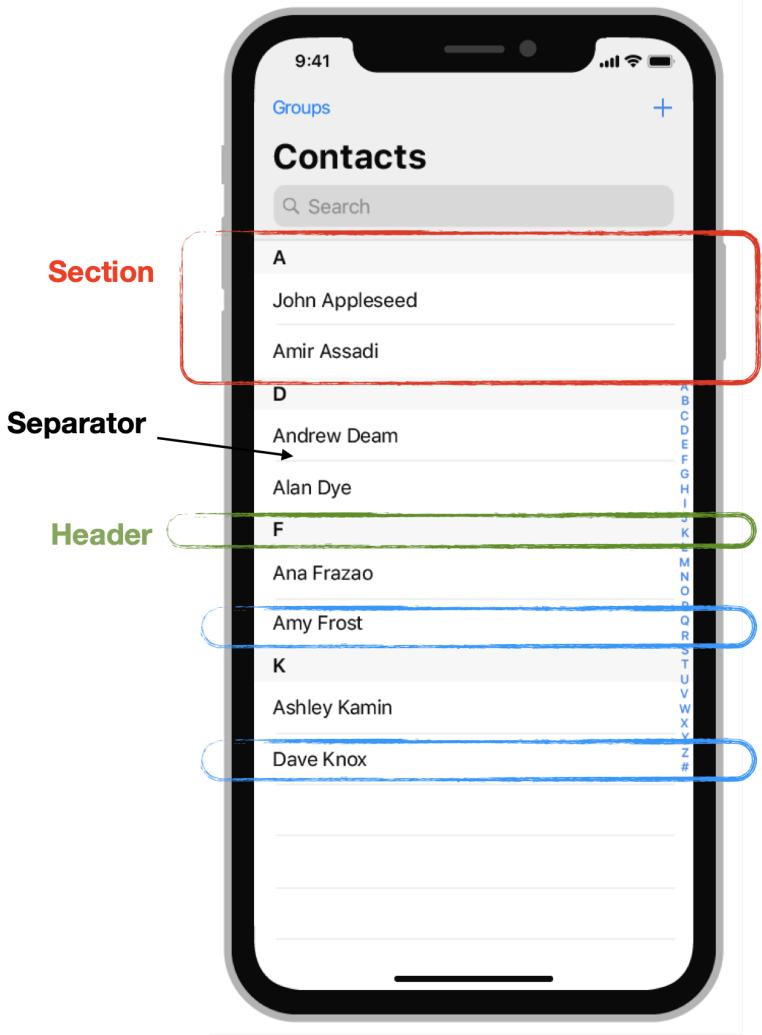
.grouped

.plain

.inset grouped



# Как всё устроено



Section

Separator

Header

Что показывать:  
UITableViewDataSource

Сообщает о событиях:  
UITableViewDelegate

IndexPath:  
Описывает положение ячейки

`indexPath.section // номер секции`  
`indexPath.row // номер ряда`

Cells

Ячейки переиспользуются

# Ячейки (подклассы от UITableViewCell)

```
// Ячейка
class SLVPersonVCCell: UITableViewCell {

    private var avatarView = UILabel()
    private var nameLabel = UITextField()

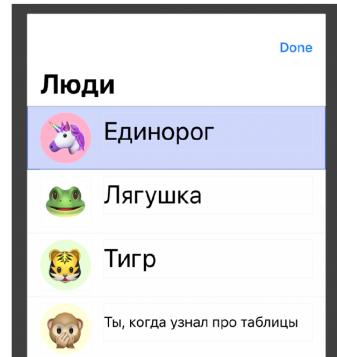
    // дефолтный конструктор ячейки
    override init(style: UITableViewCellStyle, reuseIdentifier: String?) {
        super.init(style: style, reuseIdentifier: reuseIdentifier)

        // настраиваем self.avatarView
        contentView.addSubview(avatarView)

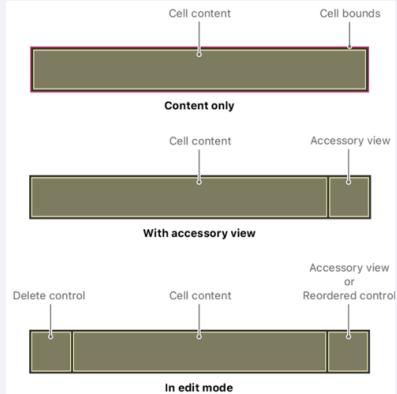
        self.nameLabel.font = UIFont.systemFont(ofSize: 32)
        self.nameLabel.adjustsFontSizeToFitWidth = true
        contentView.addSubview(nameLabel)
    }

    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }
}

func setName(_ name: String?) {
    nameLabel.text = name
}
```



## Ячейки



```
override init(style: UITableViewCellStyle, reuseIdentifier: String?) {
    super.init(style: style, reuseIdentifier: reuseIdentifier)
    contentView.addSubview(cellImageView)
    contentView.addSubview(merchantLabel)
    contentView.addSubview(deltaLabel)
    contentView.addSubview(amountLabel)
}
```

4:34

Списки

iOS

- Wwdc 404 2012
- Как делать отложенную загрузку и тд
- Контроллеры ембед
- <https://developer.apple.com/videos/play/wwdc2017/403/>
- <https://developer.apple.com/videos/play/wwdc2017/412/>
- Паттерны
- Профилировщики стэнфорд
- Symbolic link

4:34

Списки

Готово

О выбраны

- Wwdc 404 2012
- Как делать отложенную загрузку и тд
- Контроллеры ембед
- <https://developer.apple.com/videos/play/wwdc2017/403/>
- <https://developer.apple.com/videos/play/wwdc2017/412/>
- Паттерны
- Профилировщики стэнфорд
- Symbolic link

+ Напоминание

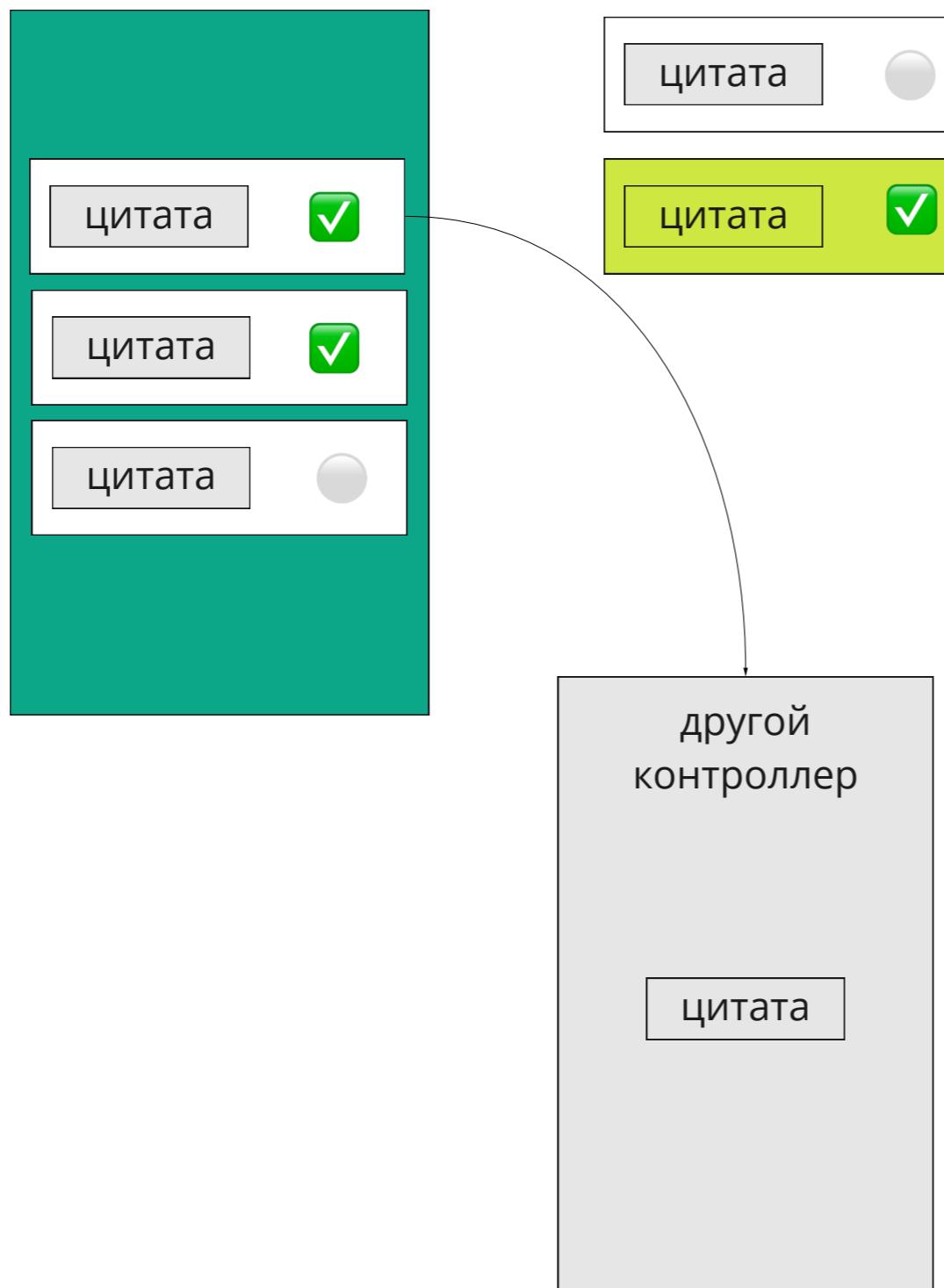
## **Header - сверху, Footer - снизу**

```
func tableView(_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {  
    return UIView(frame: CGRect(x: 0, y: 0, width: 200, height: 20))  
}
```

```
func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
```

## **Table Header, table footer**

```
t.tableFooterView = UIView(frame: CGRect(origin: .zero, size: CGSize(width: t.frame.width, height: 0)))
```



Задание - приложение для вдохновляющих цитат!

Контроллер

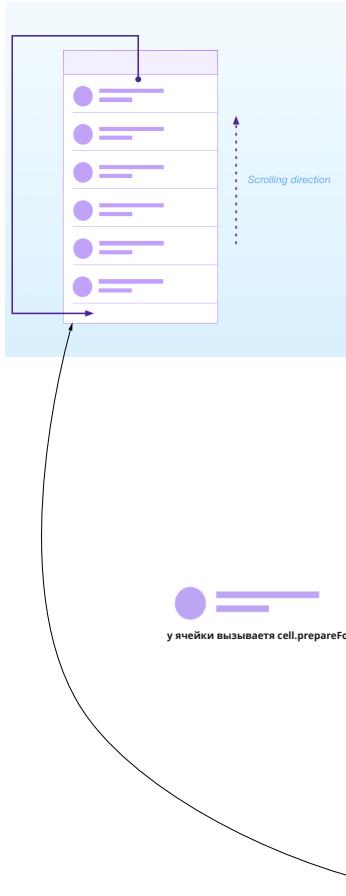
1. сделать контроллер QuotesController
2. в контроллере - массив цитат, который мы будем показывать (строки)
3. вывести список цитат на UITableView

Ячейка

1. Ячейка состоит из надписи и кнопки
2. Надпись - это цитата
3. Кнопка - это индикатор, она показывает, изучили ли мы уже цитату. Изначально ○, при нажатии - □
4. При нажатии на ячейку показываем контроллер с title = цитата (или с текстом с цитатой, как удобнее)

# Как отображаются?

Ячейки переиспользуются



мы регистрируем ячейку

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    tableView.register(SLVPersonVCCell.self, forCellReuseIdentifier: "my cell identifier")  
    //...}
```

UITableViewDataSource предоставляет данные

```
13 // Контроллер добавления людей  
14 class SLVPeopleVC: UIViewController, UITableViewDataSource //, ...  
15 {  
16  
17 //=/ ... code  
18  
19 // UITableViewDataSource  
20  
21 // количество секций  
22 func numberOfSections(in tableView: UITableView) -> Int {  
23     return 1  
24 }  
25  
26 // количество элементов в секции.  
27 func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
28     return people.count  
29 }  
30  
31 // возвращаем ячейку для indexPath  
32 func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
33  
34     // Получаем переиспользуемую ячейку по идентификатору  
35     let cell = tableView.dequeueReusableCell(withIdentifier: "my cell identifier") as! SLVPersonVCCell  
36  
37     // Что отображать  
38     let person = self.people[indexPath.row]  
39  
40     // Заполняем ячейку  
41     cell.nameLabel.text = person.name  
42     cell.setAvatar(avator:person.avatar)  
43  
44     return cell  
45 }
```

**tableView.reloadData()**

UITableViewDelegate рассказывает о событиях

```
249 public protocol UITableViewDelegate : UIScrollViewDelegate {  
250  
251     // Выбрали ячейку  
252     override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {  
253         tableView.deselectRow(at: indexPath, animated: true) // отменяем выбор  
254         let selectedPerson = people[indexPath.row]  
255         print(selectedPerson)  
256         // делайте что считаете нужным  
257     }  
258 }
```

Он наследник UIScrollViewDelegate!

```
let cell = tableView.cellForRow(at: indexPath) // получаем ячейку
```

Пример  
кода про  
клавиатуру

**ДЗ**

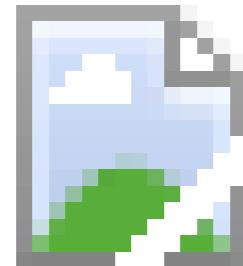
**вопросы?**

**Спасибо!**

**Чтобы передать события от ячейки:**

Либо подписываем контроллер её делегатом

код в ячейке



код в контроллере



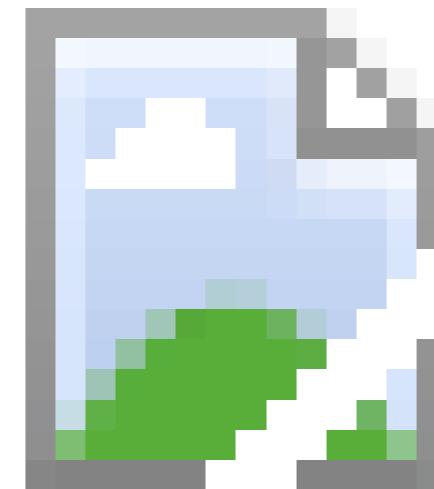
Obj-  
style

Либо передаём closure

код в ячейке



код в контроллере



Swift-  
style

# Удаление, добавление

Принцип:  
сначала  
данные,  
потом  
отображение

## Удаление

```
// Можно не изменять виджет
override func tableView(_ tableView: UITableView, canEditRowAtIndexPath indexPath: IndexPath) -> Bool {
    return true
}

// Это когда мы выбрали одну из функций edit'a в ячейке
override func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCellEditingStyle, forRowAt indexPath: IndexPath) {
    if editingStyle == .delete {
        // Итак, это удаление
        // Вызовем у tableView метод, готовящий его к изменениям
        // удаление элемента из массива, который отображаем
        self.people.remove(at: indexPath.row)
        // Удаляем ячейку из таблицы
        tableView.deleteRows(at: [indexPath], with: .automatic)
        tableView.reloadData()
    }
}
```



## Добавление

```
// Очищаем добавленные ячейки с помощью, который отобразим
self.tableView.reloadData()

let newIndexIndexPath = IndexPath(row: 0, section: 0)

// Добавляем место новой ячейки
let newCellIndexPath = IndexPath(row: 1, section: 0)

// Начинаем обновление
self.tableView.beginUpdates()

// Вставляем ради
self.tableView.insertRows(at: [newCellIndexPath], with: .automatic)

// Завершаем обновление
self.tableView.endUpdates()
```

Это  
важная  
фишка!

## Для множества разных изменений

```
// Для множества разных изменений
tableView.performBatchUpdates({
    // Удаляем все deleteRows и insertRows
}) { [weak self] in
    // здесь то, что надо сделать после всех этих изменений
}
```

## Меняем местами

```
func tableView(_ tableView: UITableView, canMoveRowAt indexPath: IndexPath) -> Bool {
    return true
}

func tableView(_ tableView: UITableView, moveRowAt sourceIndexPath: IndexPath, to destinationIndexPath: IndexPath) {
    let movedSpot = spots.remove(at: sourceIndexPath.row)
    spots.insert(movedSpot, at: destinationIndexPath.row)
}
```

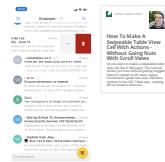
## Из кода:

```
[available iOS 5.0 +]
open func moveRow(at indexPath: IndexPath, to newIndexPath: IndexPath)
```

## Если хотите менять местами при долгом нажатии (как в напоминаниях)

```
107
108     var currentDragTransaction: DragTransaction?
109
110     @objc
111     func LongPressGestureRecognized(_ gestureRecognizer: UIGestureRecognizer) {
112         guard let longPress = gestureRecognizer as? UILongPressGestureRecognizer else { return }
113         let state = longPress.state
114         let locationInView = longPress.location(in: tableView)
115         guard state == .began || state == .changed || state == .ended || state == .cancelled else { return }
116
117         switch state {
118             case .began:
119                 self.currentDragTransaction = DragTransaction(viewModel: self.viewModel!, tableView: self.tableView)
120                 self.currentDragTransaction?.begin(startingAtIndexPath: gestureIndexPath)
121             case .changed:
122                 self.currentDragTransaction?.update(toIndexPath: gestureIndexPath)
123             case .ended:
124                 self.currentDragTransaction?.end()
125             case .cancelled:
126                 self.currentDragTransaction = nil
127         default:
128             break
129         }
130     }
131
132     // Идея - memory card
133     class DragTransaction {
134
135         var fromIndex: IndexPath?
136         var toIndex: IndexPath?
137         var tableView: UITableView?
138
139         init(fromIndexPath: IndexPath, toIndexPath: IndexPath, tableView: UITableView) {
140             self.fromIndex = fromIndexPath
141             self.toIndex = toIndexPath
142             self.tableView = tableView
143         }
144
145         private var free: UITableViewDragInteraction?
146
147         func start(atIndexPath: IndexPath) {
148             free = tableView?.dragInteractionEnabled(true)
149             free?.start(at: tableView.cellForRow(at: fromIndex!)!.center)
150             free?.start(at: tableView.cellForRow(at: toIndex!)!.center)
151             free?.cancel()
152         }
153
154         func update(toIndexPath: IndexPath) {
155             guard let free = free else { return }
156             if free.inTableview() { return }
157             free.cancel()
158             free.start(at: tableView.cellForRow(at: toIndex!)!.center)
159             free.cancel()
160             free = nil
161         }
162
163         func end() {
164             guard let free = free else { return }
165             free.cancel()
166             free = nil
167         }
168     }
169
170     func cancelDragTransaction() {
171         guard let free = free else { return }
172         if free.inTableview() { return }
173         free.cancel()
174         free = nil
175     }
176
177     @objc
178     func handleCellWillBeginEditing(_ notification: Notification) {
179         let cell = notification.userInfo![#cell] as! UITableViewCell
180         cell.addGestureRecognizer(UITapGestureRecognizer(target: self, action: #selector(handleCellTapped)))
181     }
182
183     @objc
184     func handleCellTapped(_ tapGesture: UITapGestureRecognizer) {
185         let cell = tapGesture.view as! UITableViewCell
186         cell.beginInteractiveMovementWith(#cancelDragTransaction)
187     }
188
189     @objc
190     func handleCellDidEndInteractiveMovement(_ notification: Notification) {
191         let cell = notification.userInfo![#cell] as! UITableViewCell
192         cell.endInteractiveMovement()
193     }
194 }
```

## Если хотите несколько экшнов при свайпе



# Размеры ячеек

```
func configureTableView() {
    self.tableView = UITableView(frame: self.view.frame, style:.plain)
    self.tableView.delegate = self
    self.tableView.dataSource = self

    self.tableView.rowHeight = SLVPersonVCCell.cellHeight
}
```

ИЛИ

```
func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
    switch indexPath.section {
        case Sections.graph.rawValue:
            return 220
        default:
            return 70
    }
}
```

```
95     static func heightFor(_ text: String, width: CGFloat) {
96         let attributedString = NSAttributedString(string: text,
97             [NSAttributedString.Key.font: titleFont])
98         let rect = attributedString.boundingRect(
99             with: CGSize(width: width - 72, height: .greatestFiniteMagnitude),
100            options: .usesLineFragmentOrigin,
101            context: nil
102        )
103        return rect.size.height + 16
    }
```

или autolayout!

```
t.estimatedRowHeight = 200

61 // in cell
62
63 override class var requiresConstraintBasedLayout: Bool {
64     return true
65 }
66
67 override func updateConstraints() {
68     let inset = CGFloat(16)
69     NSLayoutConstraint.activate([
70         cellImageView.topAnchor.constraint(equalTo: contentView.topAnchor, constant: 8),
71         cellImageView.heightAnchor.constraint(equalToConstant: 50),
72         cellImageView.widthAnchor.constraint(equalToConstant: 50),
73         cellImageView.leadingAnchor.constraint(equalTo: contentView.leadingAnchor, constant: inset),
74         cellImageView.bottomAnchor.constraint(equalTo: contentView.bottomAnchor, constant: -8)
75     ])
76
77     super.updateConstraints()
78 }
```

# Советы

не хранить  
данные в  
ячейках

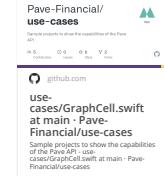
не делать  
логику в  
ячейках

передавать  
события

а если ячейки  
очень умные?

делать их  
контроллерами

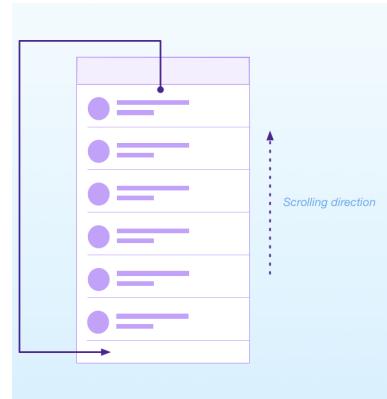
```
198 func tableView(_ tableView: UITableView, willDisplay cell: UITableViewCell, forRowAt indexPath: IndexPath) {  
199     // we add ChartViewController to cell  
200     if indexPath.section == Sections.graph.rawValue,  
201         let cell = cell as? GraphCell {  
202             addChild(cell.chartViewController)  
203             cell.chartViewController.didMove(toParent: self)  
204         }  
205     }  
206  
207     func tableView(_ tableView: UITableView, didEndDisplaying cell: UITableViewCell, forRowAt indexPath: IndexPath) {  
208         // we remove ChartViewController from cell  
209         if indexPath.section == Sections.graph.rawValue,  
210             let cell = cell as? GraphCell {  
211                 cell.chartViewController.willMove(toParent: nil)  
212                 cell.chartViewController.removeFromParent()  
213             }  
214     }  
...}
```



приложу  
код  
ячейки

# Гладкий скролл

штука для измерения фпс  
(FPSCounter)  
<https://github.com/konomafps-counter>



asyncrenderkit (texture)



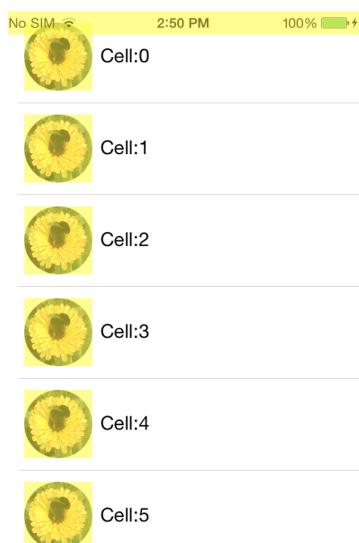
yoga, component kit

```
extension UIImage {
    func opaqueImage(backgroundColor: UIColor) -> UIImage {
        let format = UIGraphicsImageRendererFormat(for: traitCollection)
        format.opaque = true
        let rect = CGRect(origin: .zero, size: size)

        return UIGraphicsImageRenderer(size: size, format: format).image { _ in
            backgroundColor.setFill()
            UIBezierPath(rect: rect).fill()
            draw(in: rect)
        }
    }
}
```

## Ресайз картинок, углы!

Прежде всего, включите опцию Color Offscreen-Rendered Yellow. Каждая ячейка UIImageView покрыта желтым слоем.



```
@implementation UIImage (YALEExtension)

-(UIImage *)yal_imageWithRoundedCornersAndSize:(CGSize)sizeToFit {
    CGRect rect = (CGRect){0.f, 0.f, sizeToFit};

    UIGraphicsBeginImageContextWithOptions(sizeToFit, NO, UIScreen.mainScreen.scale);
    CGContextAddPath(UIGraphicsGetCurrentContext(),
                    [UIBezierPath bezierPathWithRoundedRect:rect cornerRadius:sizeToFit.width].CGPath);
    CGContextClip(UIGraphicsGetCurrentContext());

    [self drawInRect:rect];
    UIImage *output = UIGraphicsGetImageFromCurrentImageContext();

    UIGraphicsEndImageContext();
}

return output;
}
```

# Что почитать?



про гладкий скролл

podlodka - про гладкий скролл (один из первых выпусков)

## DiffableDataSource - DataSource на максималках

Before iOS 13, you'd configure a `UICollectionView`'s data source by adopting `UICollectionViewDataSource`. This protocol tells the collection view what cell to display, how many cells to display, which section to display the cells in, and so on.

**Note:** If you haven't used `UICollectionView` before, check out [UICollectionView Tutorial: Getting Started](#) to understand how it works.

The new `UICollectionViewDiffableDataSource` abstracts a significant amount of `UICollectionViewDataSource`'s logic. This leaves less room for client code errors when handling collection view's data source.

Rather than telling the data source how many items to display, you tell it what sections and items to display.

The *diffable* part of `UICollectionViewDiffableDataSource` means that whenever you update the items you're displaying, the collection view will automatically calculate the difference between the updated collection and the one previously shown. This will in turn cause the collection view to animate the changes, such as updates, insertions and deletions.

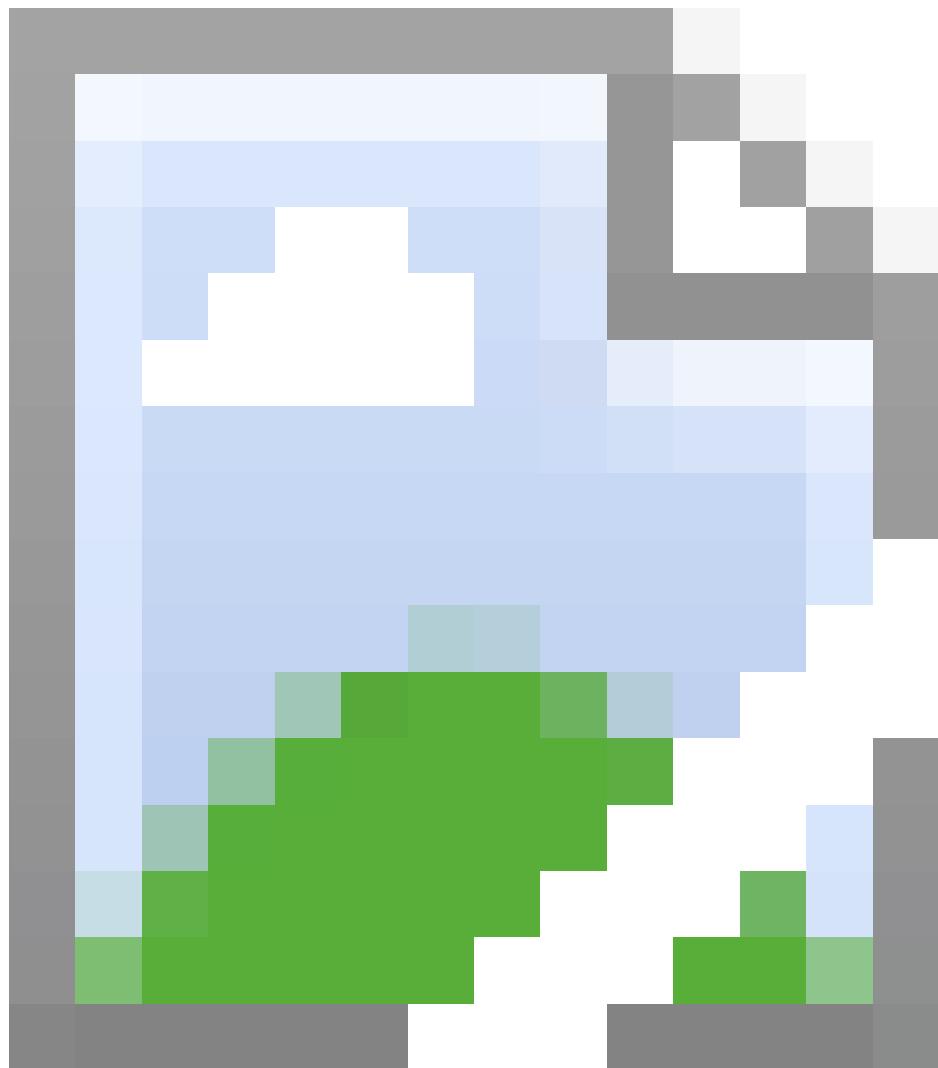


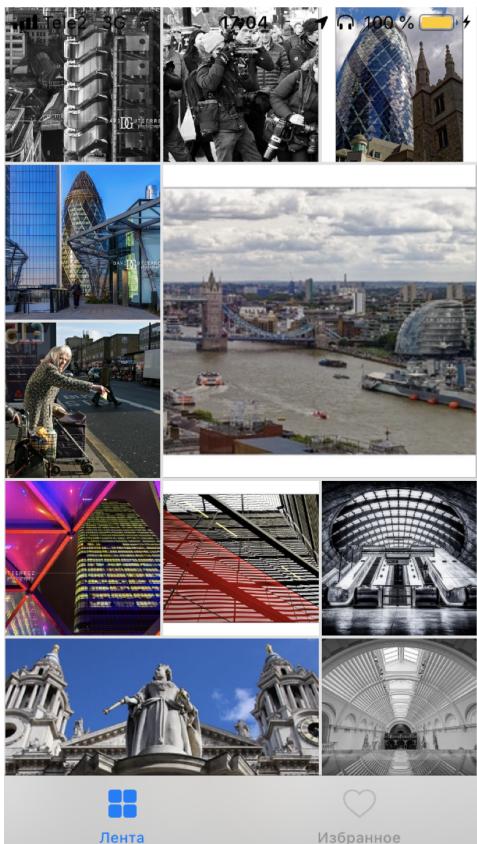
UITableViewDiffableDataSource



UICollectionViewDiffableDataSource

Вопросы?





Реџем:

1) *.dataSource*

- *numberOfSections(...)* -> *Int*
- *numberOfItemsInSection(...)* -> *Int*
- *cellForItemAtIndexPath* -> *Cell*
- ...

2) *.delegate*

- *didSelectItemAtIndexPath*
- + *UIScrollViewDelegate*
- ...

3) *Cells*

4) *UICollectionViewLayout*

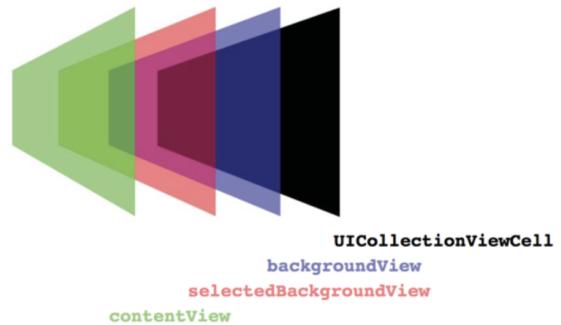
5) *Supplementary views and decoration views*

*IndexPath.item*

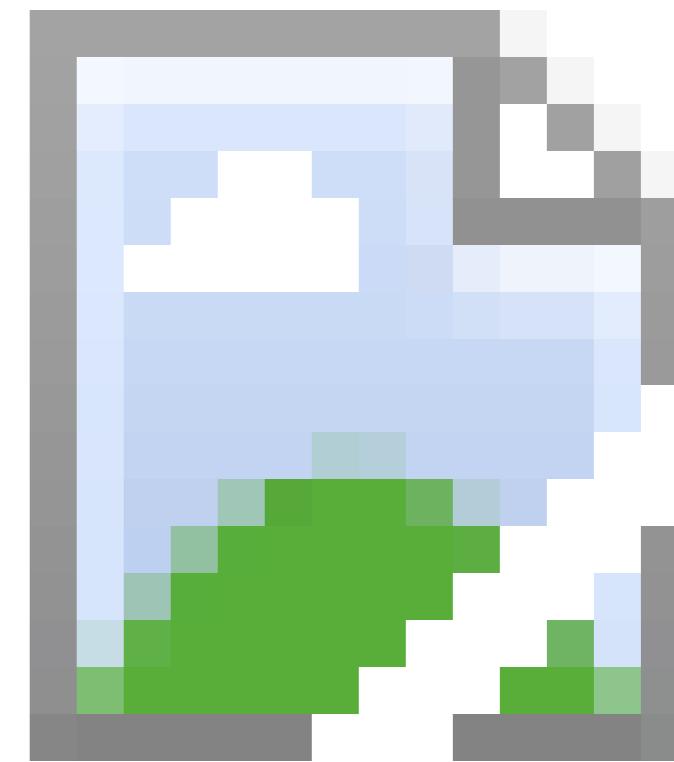
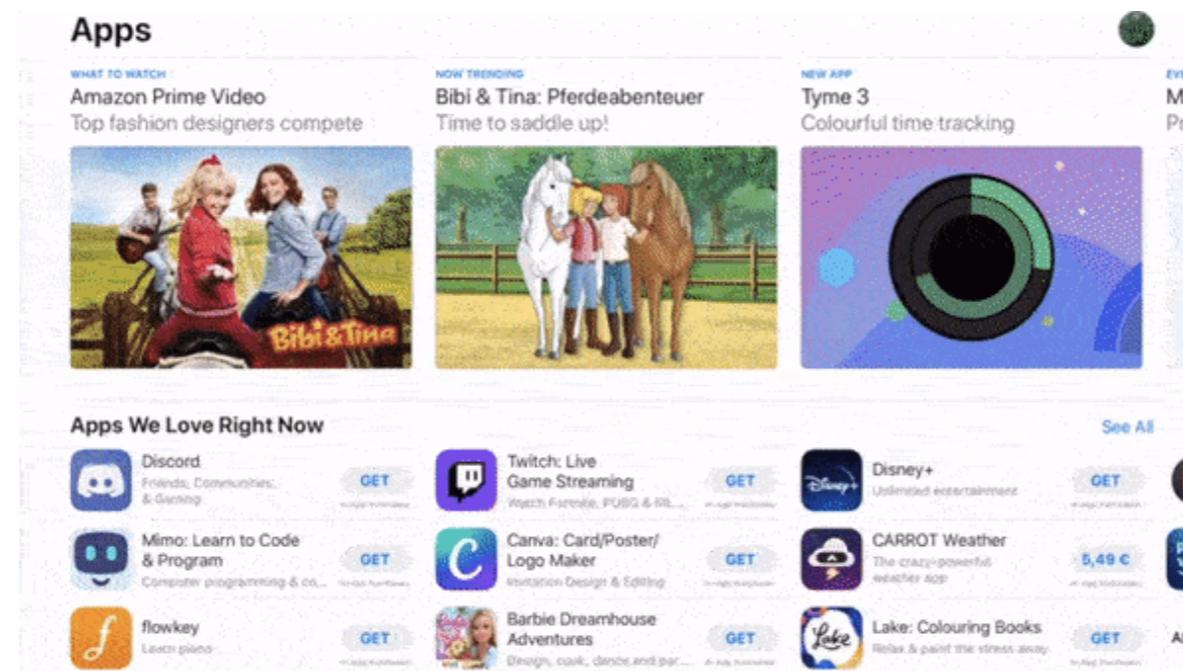
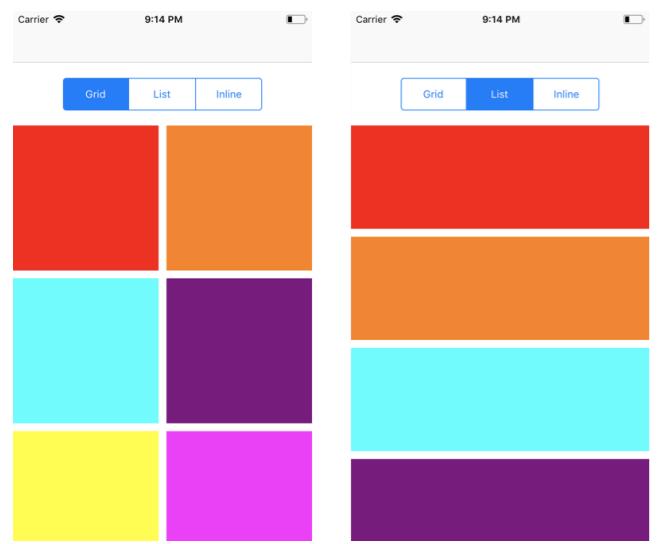
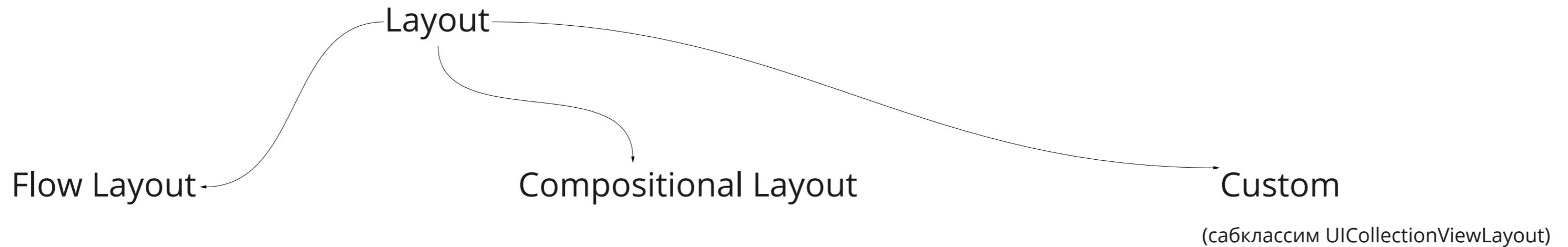
*IndexPath.section*

## Ячейки

- UICollectionViewCell и его наследники
- Ячейки переиспользуются, как у UITableView
  - collectionView.registerClass:forCellReuseIdentifier:



# Layout - определяет положение ячеек



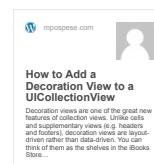
## SupplementaryViews



## DecorationViews



<https://mpospese.com/2012/12/11/decorationviews/>



## ИЗМЕНЕНИЕ COLLECTIONVIEW

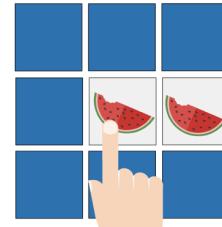
---

- !!! Сначала меняем количество элементов, потом уже меняем таблицу.
- insertItems(...) - добавляет элементы по индексам.
- deleteItems(...) - удаляет элементы по индексам.
- performBatchUpdates(...) - позволяет сделать несколько изменений одновременно и что-то по окончании.
- reloadData

## УПРАЖНЕНИЕ

---

- Под карточками скрыты пары разных картинок (вместо картинок можно использовать эмодзи)
- Правила: пользователь кликает на карточку и ему показывается картинка на карточке. Потом пользователь кликает на вторую картинку. Если картинки совпали - они остаются перевернутыми (или обе исчезают, как хотите). Иначе - обе переворачиваются обратно.



# Кастомное расположение ячеек (Layout)

## ОСНОВНЫЕ ИНГРЕДИЕНТЫ

- UICollectionViewLayoutAttributes - описывают положение каждого элемента
- shouldInvalidateLayout(forBoundsChange newBounds: CGRect) - вызывается на каждое изменение баундсов коллекции.
- prepare() - вызывается каждый раз при инвалидации лэйаута + при первом отображении. В ней хорошо создавать лэйаут.
- collectionViewContentSize - определяет размер коллекши вью
- layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? - лэйаут элементов для заданного квадрата. Часто вызывается. Например при прокрутке. Поэтому иногда надо оптимизировать.
- layoutAttributesForItem(at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? - лэйаут для конкретного айтема.

*Demo!*

```
@available(iOS 6.0, *)
open class UICollectionViewLayoutAttributes : NSObject, NSCopying, UIDynamicItem {

    open var frame: CGRect
    open var center: CGPoint
    open var size: CGSize
    open var transform3D: CATransform3D
    @available(iOS 7.0, *)
    open var bounds: CGRect
    @available(iOS 7.0, *)
    open var transform: CGAffineTransform
    open var alpha: CGFloat
    open var zIndex: Int // default is 0
    open var isHidden: Bool // As an optimization, UICollectionView might not create a view for
                           // items whose hidden attribute is YES
```

## Рецепт

- Рассчитать frame для всех ячеек в методе prepare .
- Вернуть видимые ячейки в методе layoutAttributesForElements(in:).
- Вернуть параметры ячейки по её индексу в методе layoutAttributesForItem(at:).  
Например, этот метод используется при вызове у коллекшна метода scrollToItem(at:).
- Вернуть размеры получившегося контента в collectionViewContentSize . Так коллекшен узнает, где граница, до которой можно скролить.

```
- (CGPoint)targetContentOffsetForProposedContentOffset:(CGPoint)proposedContentOffset API_AVAILABLE(ios(7.0)); // a layout
can return the content offset to be applied during transition or update animations
```

## Drag & Drop

- <https://habr.com/en/post/425817/> - статья на хабре
- <https://developer.apple.com/videos/play/wwdc2017/223> - видос от эппла
- <https://www.raywenderlich.com/9477-uicollectionview-tutorial-reusable-views-selection-and-reordering> - в конце drag&drop, а так - ещё много всего полезного

# Кастомное расположение ячеек (Layout)

## Практика

создаём флоу  
лэйаут сами

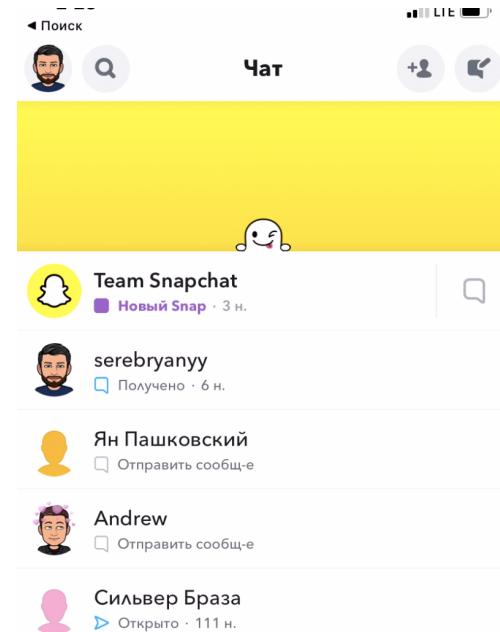
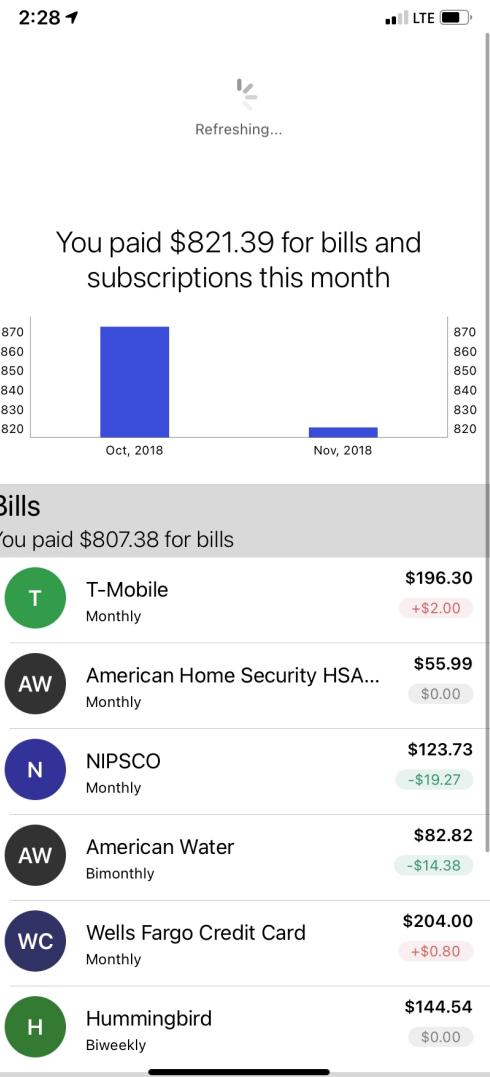
# Pull to refresh

```
var refreshControl = UIRefreshControl()

refreshControl.attributedTitle = NSAttributedString(string: "Refreshing...")
refreshControl.addTarget(self, action: #selector(self.refresh(_:)), for: .valueChanged)
tableView.addSubview(refreshControl)

// ...
@objc func refresh(_ sender: AnyObject) {
    refreshControl.endRefreshing()
    // do smth
    tableView.reloadData()
}

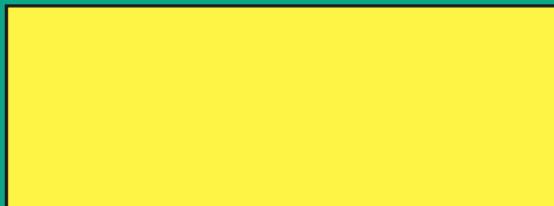
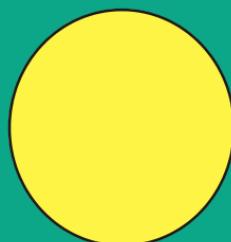
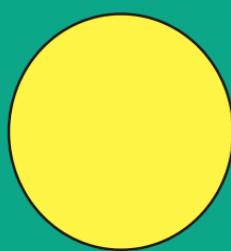
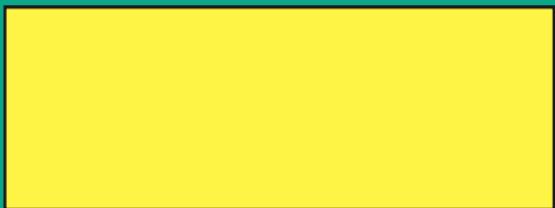
```



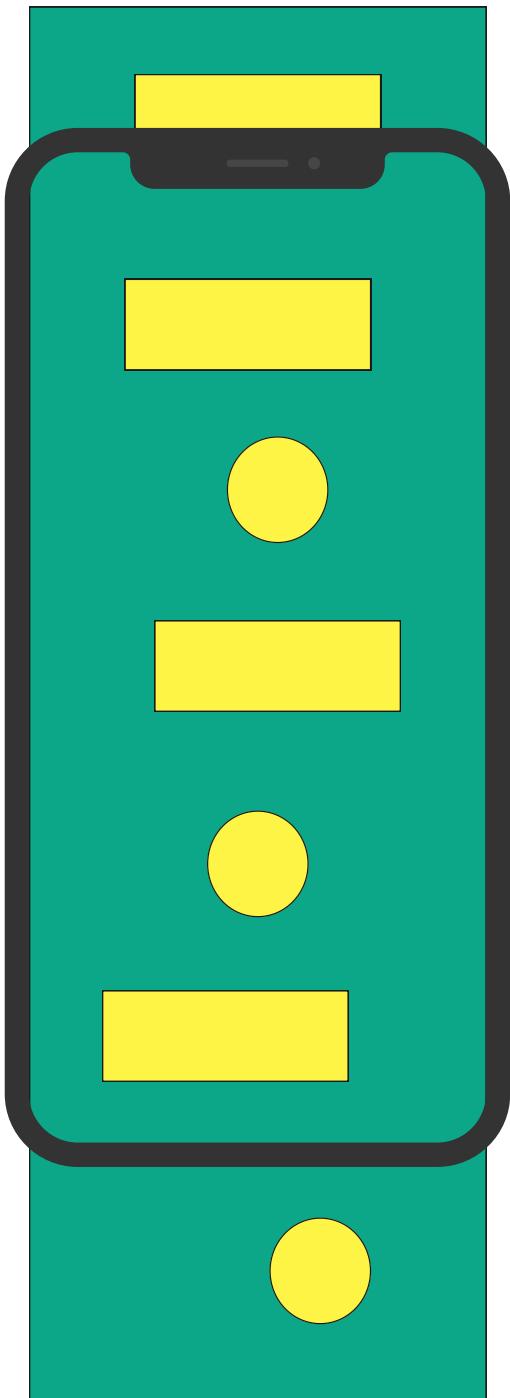
- Скролл вью <https://www.objc.io/issues/3-views/scroll-view/>
- Bounds vs Frames - где баундс, а где фрейм, и когда что есть  
<https://www.raywenderlich.com/4040-scroll-view-school/lessons/2>
- <https://developer.apple.com/documentation/uikit/uicollectionview>
- Статья ДОДО Пиццы про то, как они сделали крутой лэйаут для половинчатых пицц. Он увеличивает выбранную пиццу и меняет её прозрачность. Очень советую, она ещё и классно написана -  
<https://habr.com/ru/company/dodopizzaio/blog/452876/>
- <https://developer.apple.com/videos/play/wwdc2018/225/> - WWDC про кастомные лэйауты
- <https://github.com/roberthein/BouncyLayout> - пружинящий лэйаут
- Гайды по коллекшн вью
  - <https://www.raywenderlich.com/9334-uicollectionview-tutorial-getting-started>
  - <https://www.raywenderlich.com/9477-uicollectionview-tutorial-reusable-views-selection-and-reordering>
- <https://mpospese.com/2012/12/11/decorationviews/> - кастомные декораторы (как сепараторы у тэйбл вью)

**Пример!**

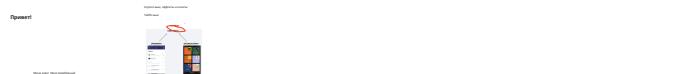
ScrollView  
Delegate -  
показать



**а как сделать только вертикальный скролл?**



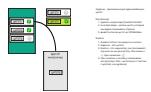
content size = ?

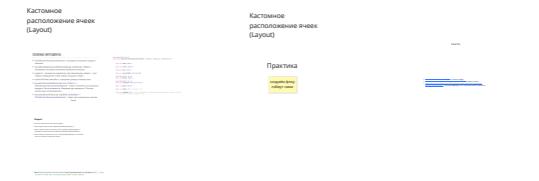
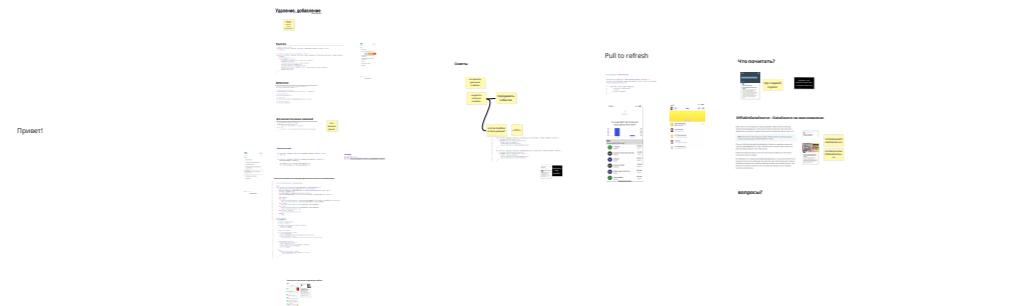


# UIScrollView

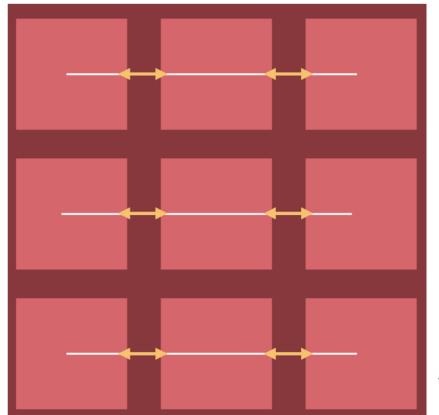


# UITableView

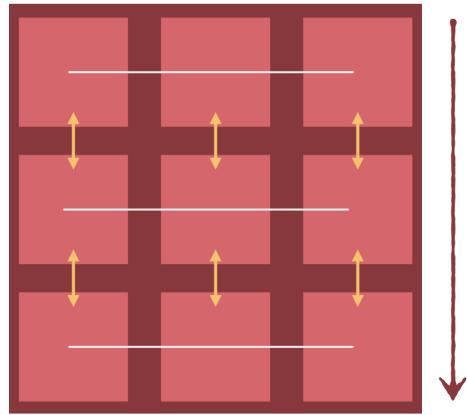




## UICollectionViewFlowLayout

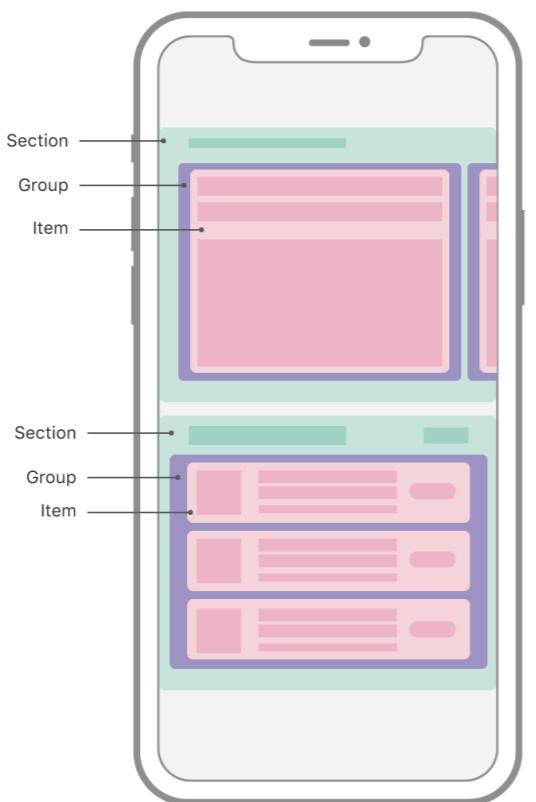


*interItem spacing*



*line spacing*

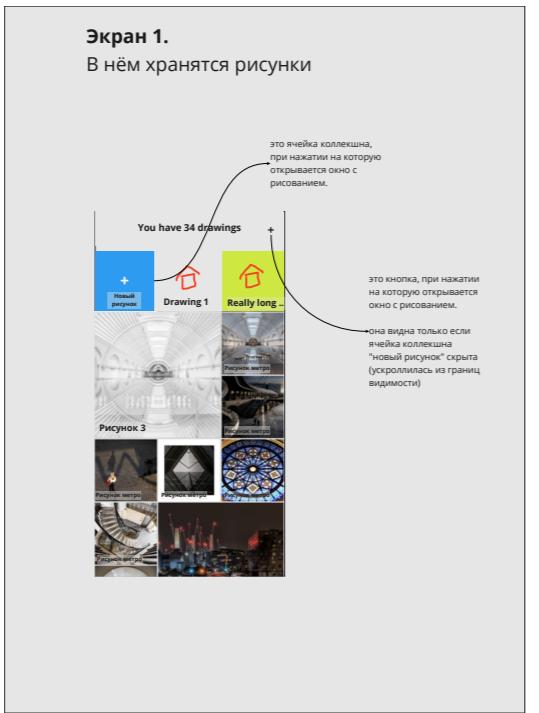
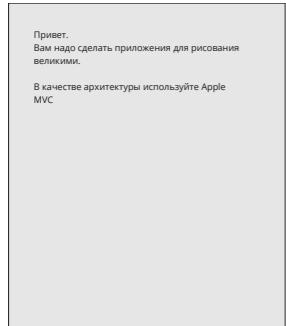
# CompositionalLayout



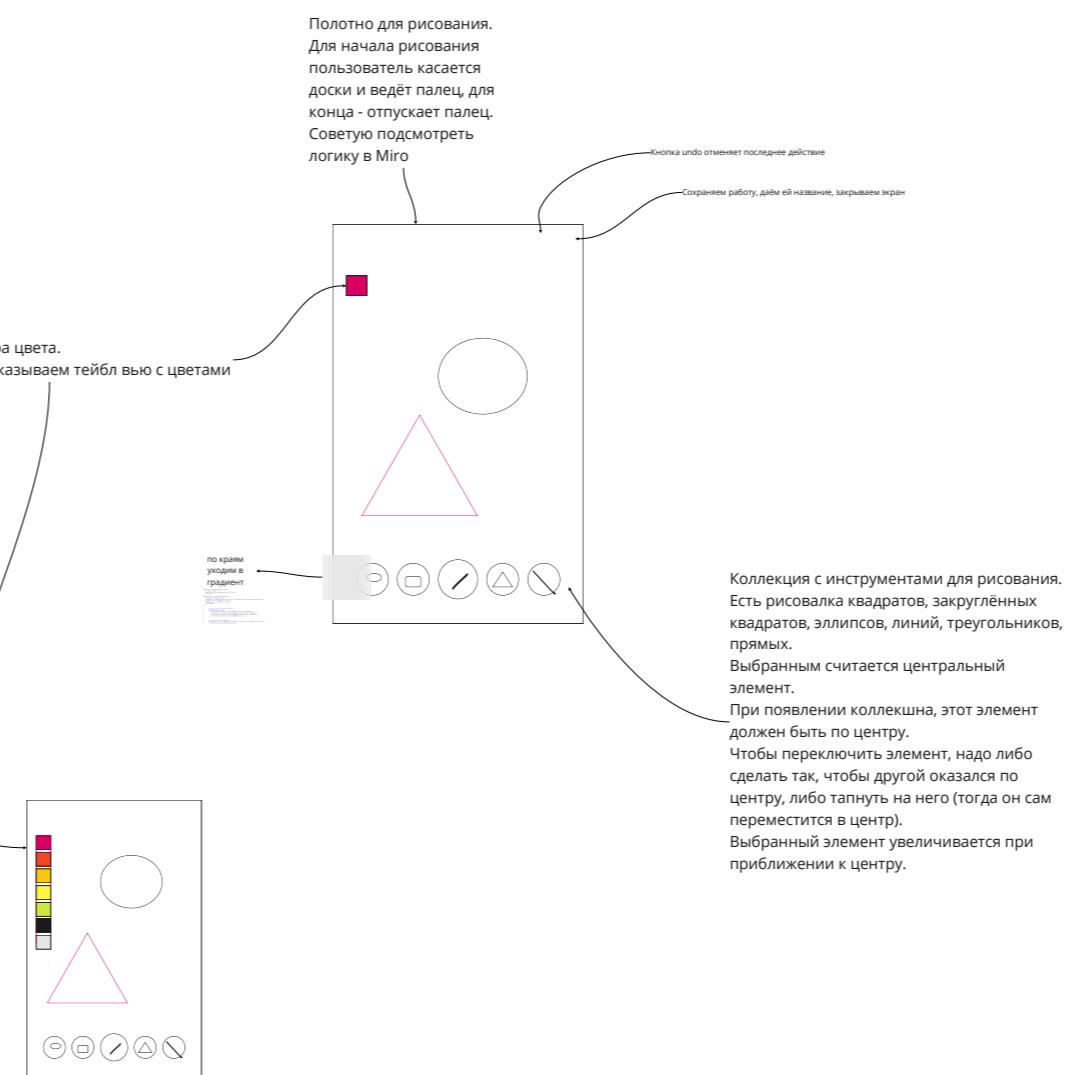
Срок: до 23:59 первого августа

Дз рассчитано на команды по три человека. Завтра пришлю составы команд.

У каждой команды должен быть приватный репозиторий под этот проект, ссылки кидайте мне.



**Экран 2.**  
Окно с рисованием



#### Как сделать рисование?

Есть много вариантов.

Несколько советов:

1. Советую вам использовать новый layer для каждой новой фигуры и объединять всё в одну картинку уже при сохранении
2. Для перевода вью в картинку, используйте UIGraphicsImageRenderer
3. Для рисования кругов, квадратов с закруглёнными углами и овалов используйте UIBezierPath <https://developer.apple.com/documentation/uikit/uibezierpath>
4. Или CAShapeLayer <https://medium.com/flawless-app-stories/drawing-using-cashapeplayer-in-ios-9a6c83de7eb2>
5. В любом случае, прочитайте вот это <https://developer.apple.com/documentation/uikit/uiview/1622529-drawrect> - поможет с рисованием
6. Как сделать распознавание жестов вы знаете
7. Используйте шаблоны проектирования, с ними будет сильно легче
8. Не забывайте про всякие хорошие фишки гита вроде веток и описаний коммитов

Хранить картинки можете в оперативке (хранить массив в корневом контроллере и всё в него передавать - самое простое), или в файлах, или в UserDefaults (второе место по простоте)

Привет!