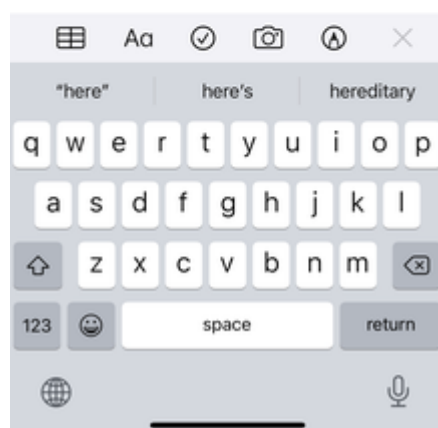
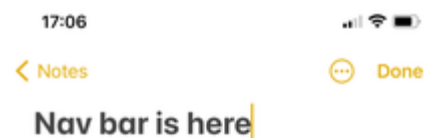


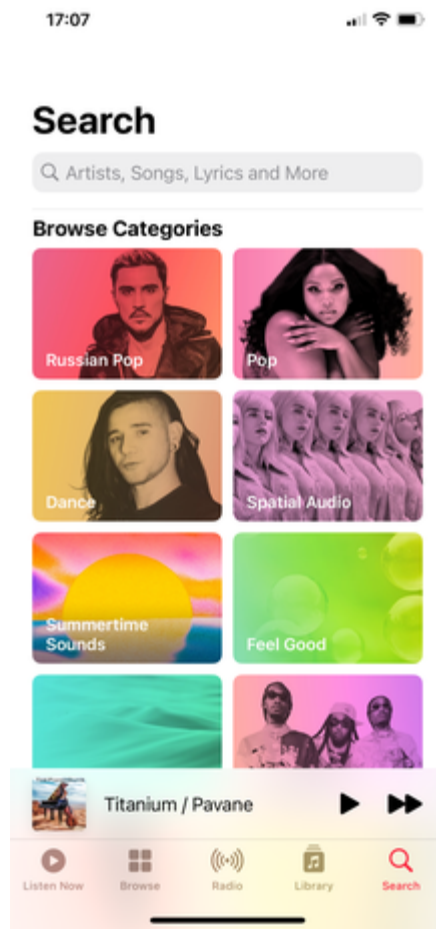
Часть 5. Навигация между экранами, условия и классы

Наш новый проект кардинально отличается от предыдущих - теперь в нашем сториборде окажется целых три экрана, каждый из которых будет показан в зависимости от действий пользователя! Оценить их внешний вид можно [на макетах в Фигме \(экраны 2 - 4\)](#).

В iOS навигация между экранами может выглядеть по-разному. Самый распространенные виды - это *Navigation Bar* сверху (как в приложении “Заметки”)



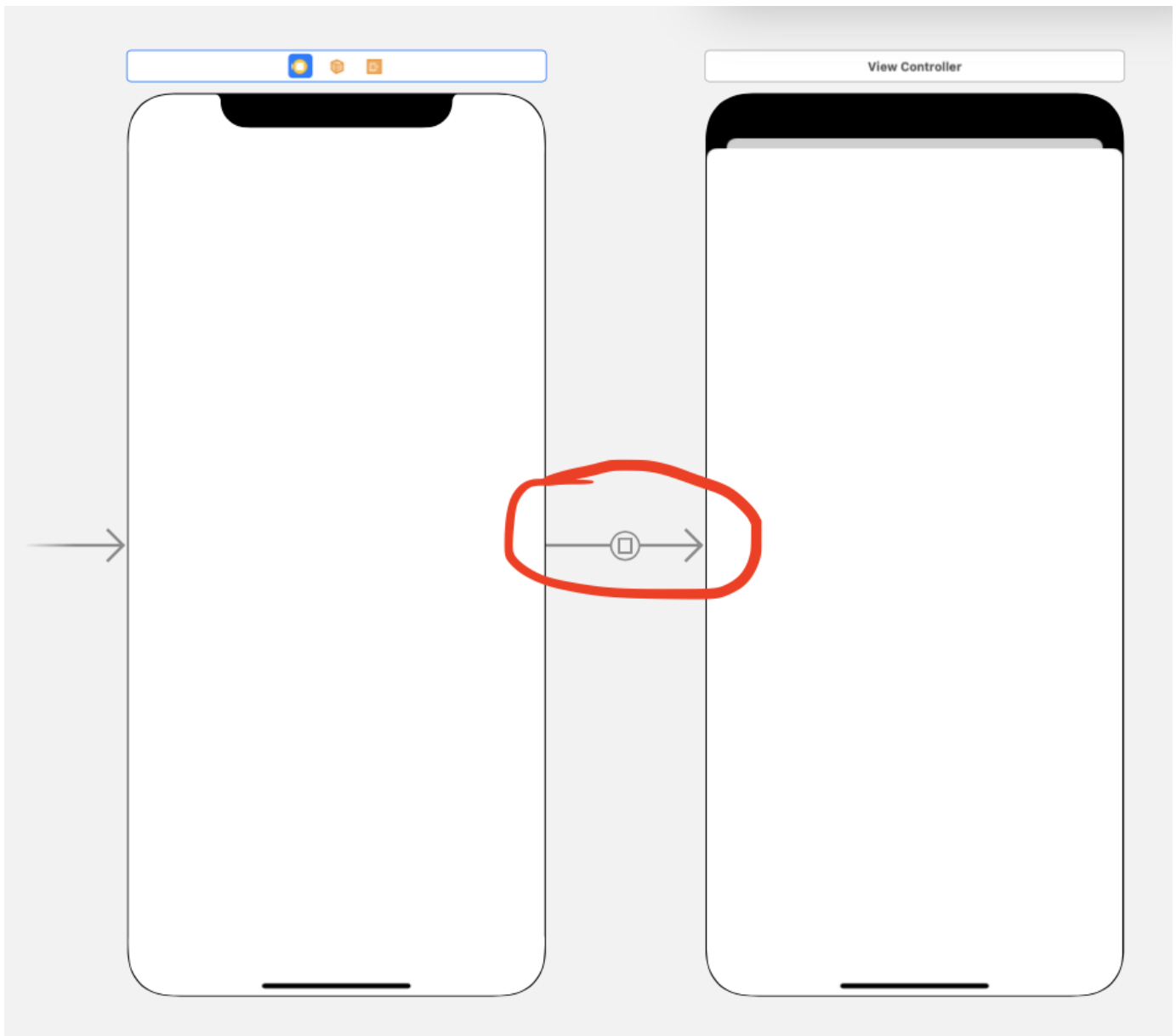
Tab Bar снизу (как в приложении “Музыка”)



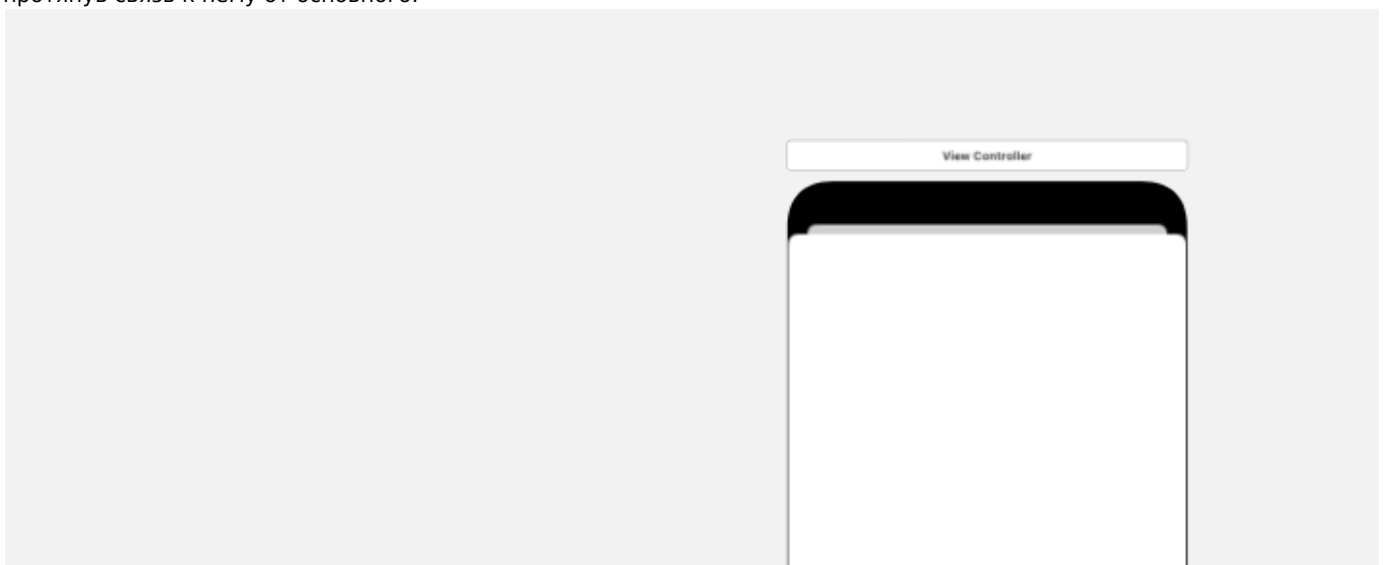
И модальные окна, как в наших макетах. Для них мы будем использовать сегвеи (от слова segue - переход).
Сначала добавим экраны в наш сториборд. Откроем список компонентов (кнопка "+"), найдем *View Controller* и "перетащим" его рядом с уже существующим

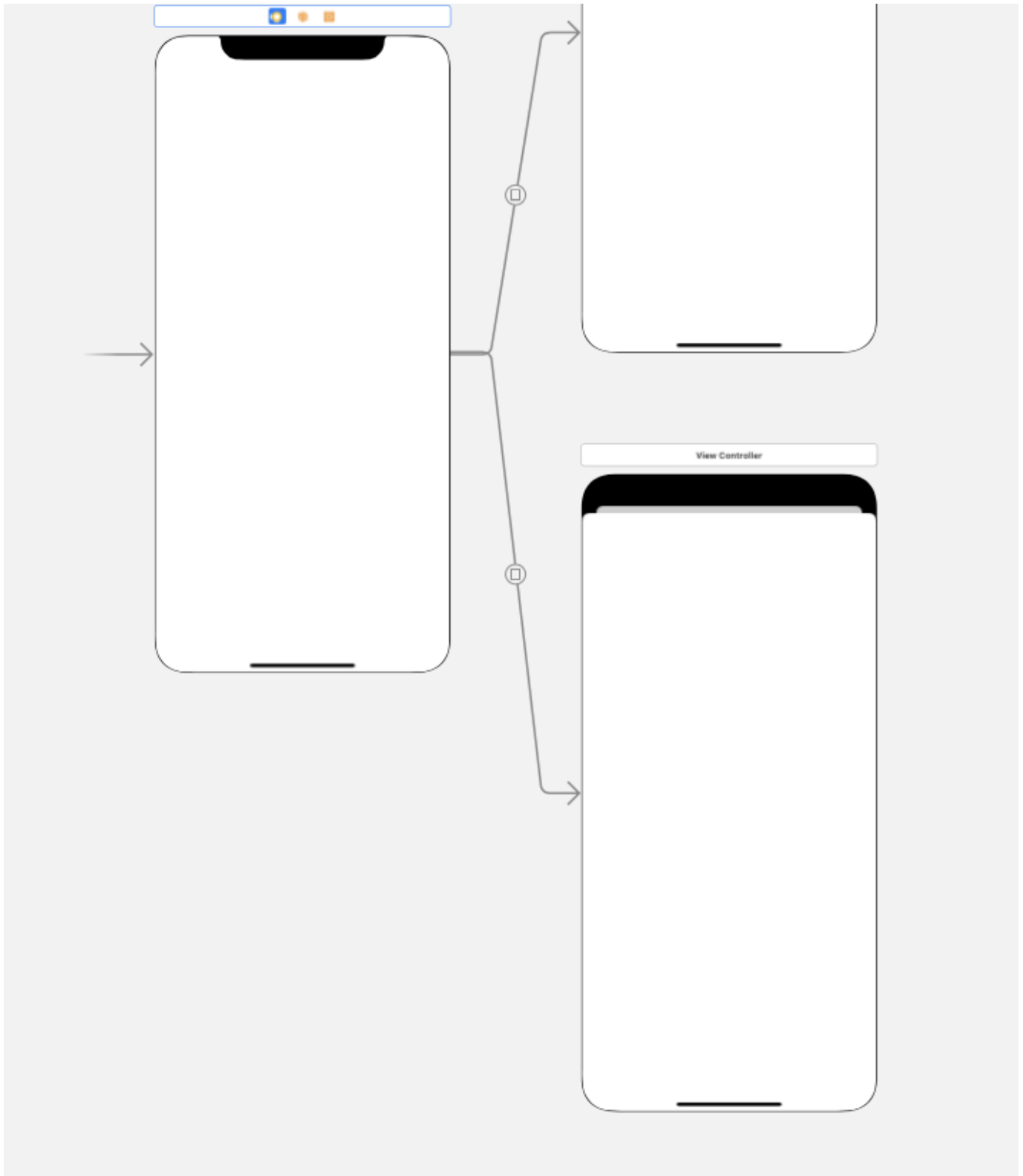


У изначального контроллера слева есть стрелка, обозначающая точку входа, то есть, при запуске приложения, этот экран будет показан первым. Назовем его основным экраном (он будет соответствовать макету в Фигме №2) и создадим связь между ним и второстепенным экраном. Выберем основной контроллер (нажав на левую желтую иконку в шапке экрана), и протянем связь к другому контроллеру, зажав *Control*. В появившемся меню выберем *Present Modally*.



Появившаяся линия между экранами - это и есть наш сегвей! Повторим все шаги, добавив третий *View Controller* и протянув связь к нему от основного.





Чтобы Xcode понимал, к какому экрану мы хотим перейти, у каждого сегвее должен быть уникальный идентификатор. Нажмем на связь между экранами, и в атрибутах увидим поле Identifier. Установим для первого перехода идентификатор "goToSuccess", а для второго - "goToFail".

Storyboard Segue

Identifier

Class

Module

☐ Inherit Module From Target

Selector

Kind

Presentation

Transition

☒ Animates

Совершить переход между экранами поможет стандартная функция *performSegue*. Добавим на главный экран 5 кнопок с числами. От одной из них протянем связь и установим следующие параметры

Connection	<input type="text" value="Action"/>
Object	<input type="text" value="View Controller"/>
Name	<input type="text" value="didTapNumberButton"/>
Type	<input type="text" value="UIButton"/>
Event	<input type="text" value="Touch Up Inside"/>
Arguments	<input type="text" value="Sender"/>
<input type="button" value="Cancel"/> <input type="button" value="Connect"/>	

В поле *Arguments* **обязательно** нужно оставить *Sender*, он нам пригодится)

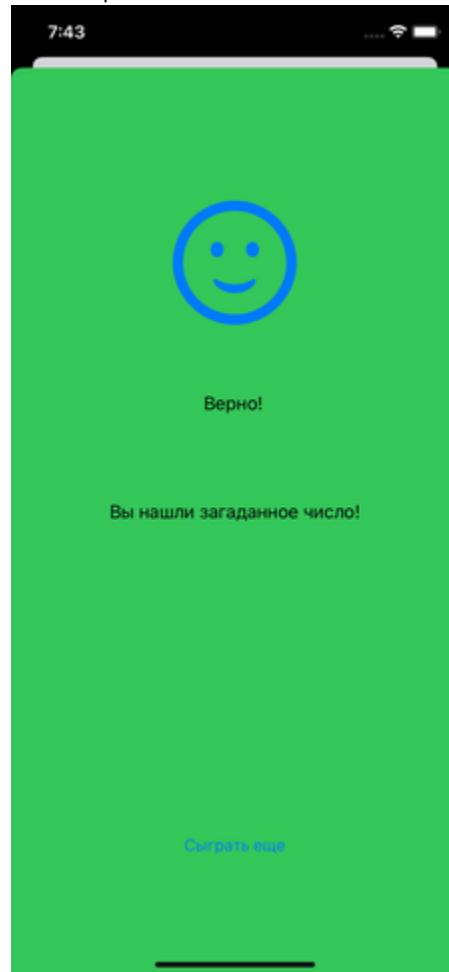
Теперь соединим получившийся *IBAction* со всеми кнопками на экране! При нажатии любой из кнопок будет выполняться один и тот же код. Чтобы проверить этот трюк, протянем связь в обратную сторону. В файле *ViewController.swift* рядом с *IBAction* есть закрашенная точка (она означает, что этот кусок кода связан со сторибордом).

● **@IBAction func** didTapNumberButton(_ sender: UIButton) {

От этой точки протянем связь к нашему главному экрану, к каждой из оставшихся кнопок (если навести мышкой на закрашенную точку, в сториборде подсвелятся элементы, с которыми она связана). Внутри функции *didTapNumberButton* напомним

```
performSegue(withIdentifier: "goToSuccess", sender: self)
```

Теперь при нажатии на кнопку наше приложение будет осуществлять переход (сегвей) с идентификатором *goToSuccess* - он как раз ведет на один из второстепенных экранов



Появившееся модальное окно можно закрыть свайпом сверху вниз и вернуться на главный экран. Обратите внимание, что кнопка “Сыграть еще” пока не работает, к ней вернемся чуть позже)

Логика нашего будущего приложения такова: при нажатии на кнопку происходит выбор случайного числа от одного до пяти. Если это число совпадает с указанным на кнопке, юзер видит экран успеха. Если нет - экран проигрыша. Реализовать это поможет оператор **условия**, который выглядит так

```
if <> { 1 } else { 2 }
```

Первое действие выполняется, если условие верное, второе - если неверное. Например

```
if a > 2 {  
    print(" ")  
} else {  
    print(" ")  
}
```

Если переменная *a* больше двух, мы увидим первую надпись, если нет - вторую

Вернемся к нашему контроллеру. По нажатию кнопки мы должны сделать следующее:

1. Создать константу с случайным целым числом от одного до пяти
2. Узнать номер кнопки, на которую нажал пользователь (*Sender* - это и есть наша кнопка. Мы можем взять её title (текст на кнопке) и перевести в целое число)
3. Сравнить константы из пунктов 1 и 2. Если они равны, выполним сегвей *"goToSuccess"*, если нет - *"goToFail"*

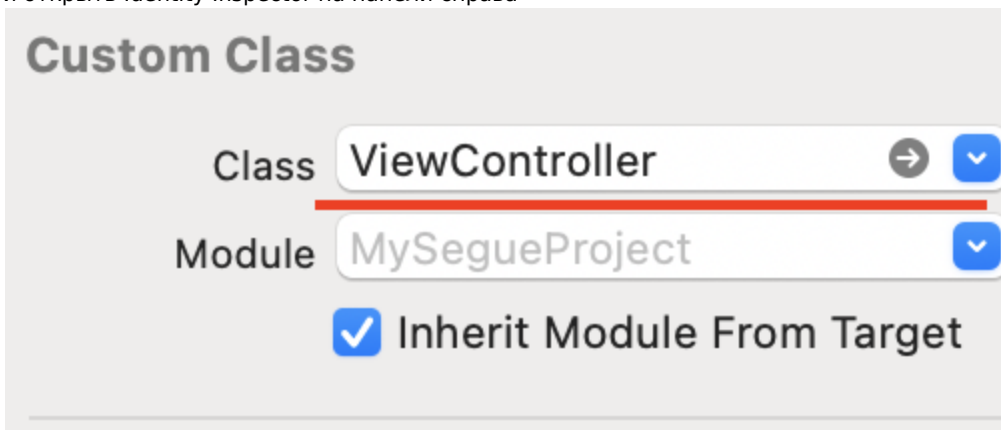
Итог выглядит так

```
@IBAction func didTapNumberButton(_ sender: UIButton) {
    let randomNumber = Int.random(in: 1...5)
    let buttonNumber = Int((sender.titleLabel?.text)!)

    if randomNumber == buttonNumber {
        performSegue(withIdentifier: "goToSuccess", sender: self)
    } else {
        performSegue(withIdentifier: "goToFail", sender: self)
    }
}
```

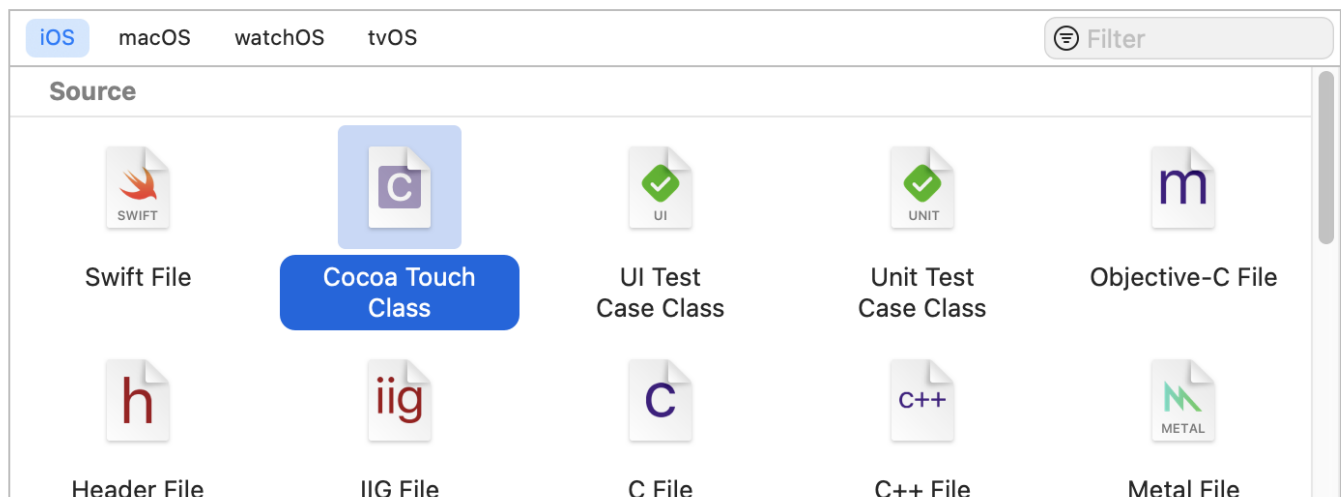
Следующий шаг - реализовать возврат пользователя на начальный экран по нажатию на кнопки "Сыграть еще" и "Повторить попытку". Для этого нам нужно познакомиться с **классами**.

Класс - это шаблон для создания объектов. В нашем проекте уже есть класс *ViewController* (его можно увидеть в файле *ViewController.swift*). Из него создается наш главный контроллер - это можно проверить, если выбрать его в сториборде и открыть Identity Inspector на панели справа




У второстепенных экранов нет своих файлов с кодом, а значит и нет кастомных классов. Создадим их прямо сейчас, нажав в верхней панели File → New → File... и выбрав Cocoa Touch Class


Choose a template for your new file:




User Interface




SwiftUI View




Storyboard



View



Empty



Launch Screen

Cancel

Previous

Next

На следующем экране нужно назвать новый класс (например, `SuccessViewController`) и выбрать шаблон для него (Subclass of: `UIViewController`)

Choose options for your new file:

Class:

Subclass of:

UIViewController

☐ Also create XIB file

Language:

Swift

Cancel

Previous

Next

Среди файлов нашего проекта появился новый - с кастомным классом `SuccessViewController`! Откроем Identity Inspector одного из второстепенных контроллеров (того, который отвечает за успешно угаданное число) и "познакомим" его с этим классом, указав его в нужном поле

Custom Class

Class

SuccessViewController

→

▼

Module

MySegueProject

▼



Inherit Module From Target

Теперь если выбрать этот контроллер, и открыть *Assistant*, рядом с ним появится наш новенький класс! Поскольку экран связан с этим файлом, мы можем протянуть в него связь от кнопки “Сыграть еще”, создать *IBAction*, и по нажатию кнопки скрывать этот контроллер (для скрытия модального окна используется простая функция *dismiss*). Весь файл в итоге выглядит так

```
import UIKit

class SuccessViewController: UIViewController {

    @IBAction func didTapPlayAgainButton() {
        dismiss(animated: true)
    }
}
```

Для контроллера “неудачи” повторим эти действия - создадим класс *FailViewController*, укажем его в *Identity Inspector* оставшегося экрана, протянем связь и используем *dismiss*.

Наше новое приложение с игровыми элементами и переходом между экранами готово!