FIAP

MBA+

DESENVOLVIMENTO DE APLICAÇÕES E GAMES PARA DISPOSITIVOS MÓVEIS INTERNET DAS COISAS

FIAP

MBA+

DESENVOLVIMENTO DE APLICAÇÕES IOS

Prof. Eric Alves Brito
proferic.brito@fiap.com.br

2017

Aula 03

FIAP

- Bundle
- UITableViewController
- JSON
- UITableViewDataSource/DataSource
- UITableView / UITableViewCell

BUNDLE

Bundle

FIAP

- Um Bundle é um diretório com uma estrutura hierárquica padrão que serve armazena códigos executáveis e recursos usados por esses códigos. Por fornecer uma localização conhecida e bem estruturada, eles simplificam a tarefa de acesso a esses recursos.
- Quando desejamos inserir arquivos em nosso projeto, utilizamos o objeto Bundle para termos um acesso rápido e prático a esse conteúdo.
- O Bundle principal de todo aplicativo é acessado utilizando-se a propriedade de classe ***main*** da classe ***Bundle*** (***Bundle.main***)
- Ao adicionarmos arquivos em nosso App, é importante não esquecer de adicioná-lo ao target que desejamos utilizar, para que dessa forma ele seja inserido no Bundle do aplicativo.

Bundle

FIAP

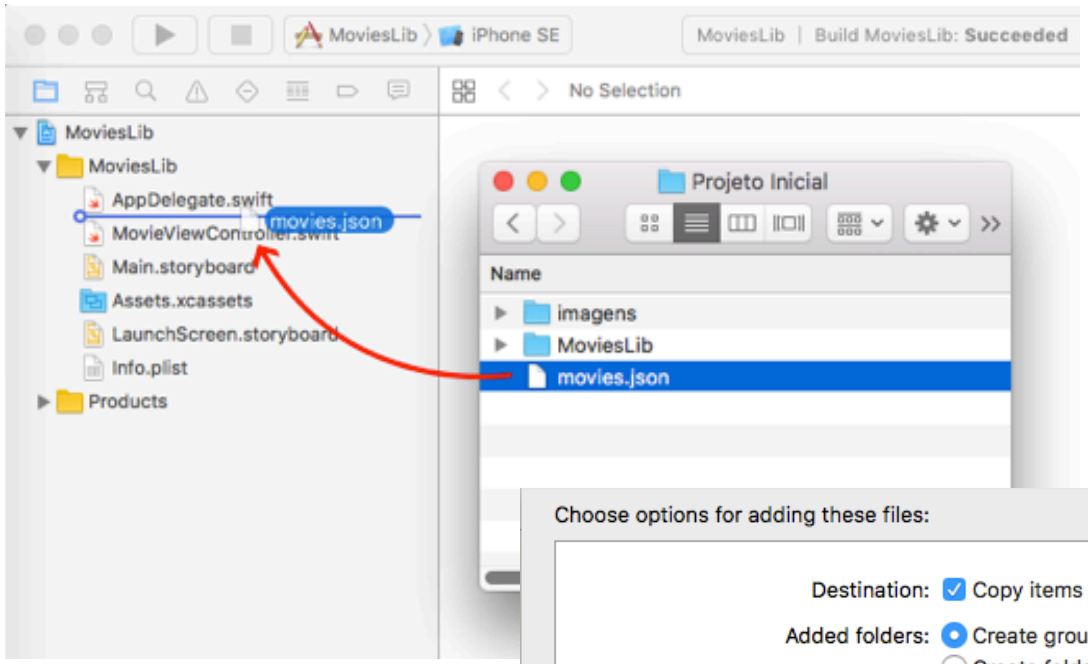
Projeto MoviesLib

- Para continuarmos o desenvolvimento do nosso aplicativo, vamos utilizar o arquivo que se encontra no Portal com o nome **movieslib2.zip**
- Ao descompactar, dentro de “Projeto Inicial” você terá 2 pastas (imagens e MoviesLib) e um arquivo movies.json. Abra o projeto MoviesLib e adicione o arquivo movies.json para o projeto. Para isso, arraste pelo Finder o arquivo e solte na estrutura de arquivos do seu projeto (***fig. 1***).
- Na janela que se abre, certifique-se de marcar as opções “***Copy items if needed***”, para que o arquivo seja inserido na pasta do projeto, e “***Add to targets:***” para que ele seja efetivamente inserindo no Bundle, caso contrário, o arquivo estará no seu projeto mas não estará disponível no Bundle quando o App for gerado (***fig 2***).
- Para recuperar o caminho (URL) de um arquivo no Bundle, usamos o método abaixo. Mais tarde, implementaremos este código no nosso projeto.

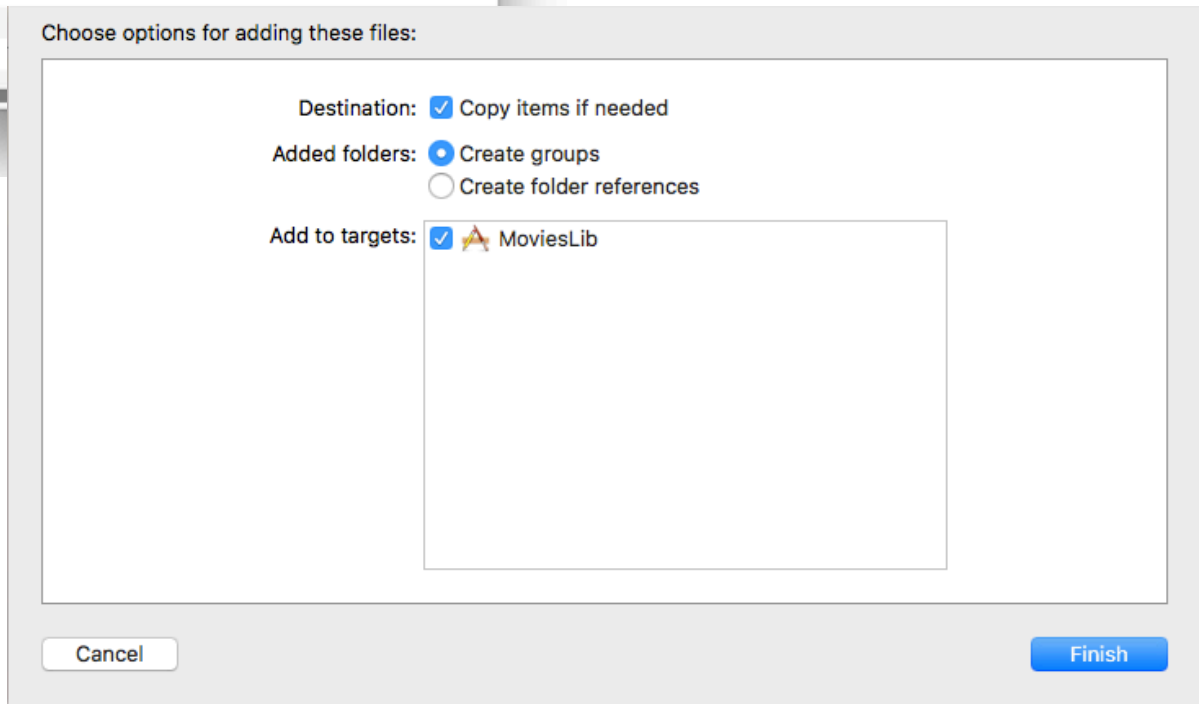
```
Bundle.main.url(forResource: "movies", withExtension: "json")
```

Bundle

1


MoviesLib
iPhone SE
MoviesLib | Build MoviesLib: Succeeded
No Selection
MoviesLib
MoviesLib
AppDelegate.swift
movies.json
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
Projeto Inicial
Name
imagens
MoviesLib
movies.json

2


Choose options for adding these files:
Destination:
Copy items if needed
Added folders:
Create groups
Create folder references
Add to targets:
MoviesLib
Cancel
Finish

UITABLEVIEWCONTROLLER

UITableViewController

FIAP

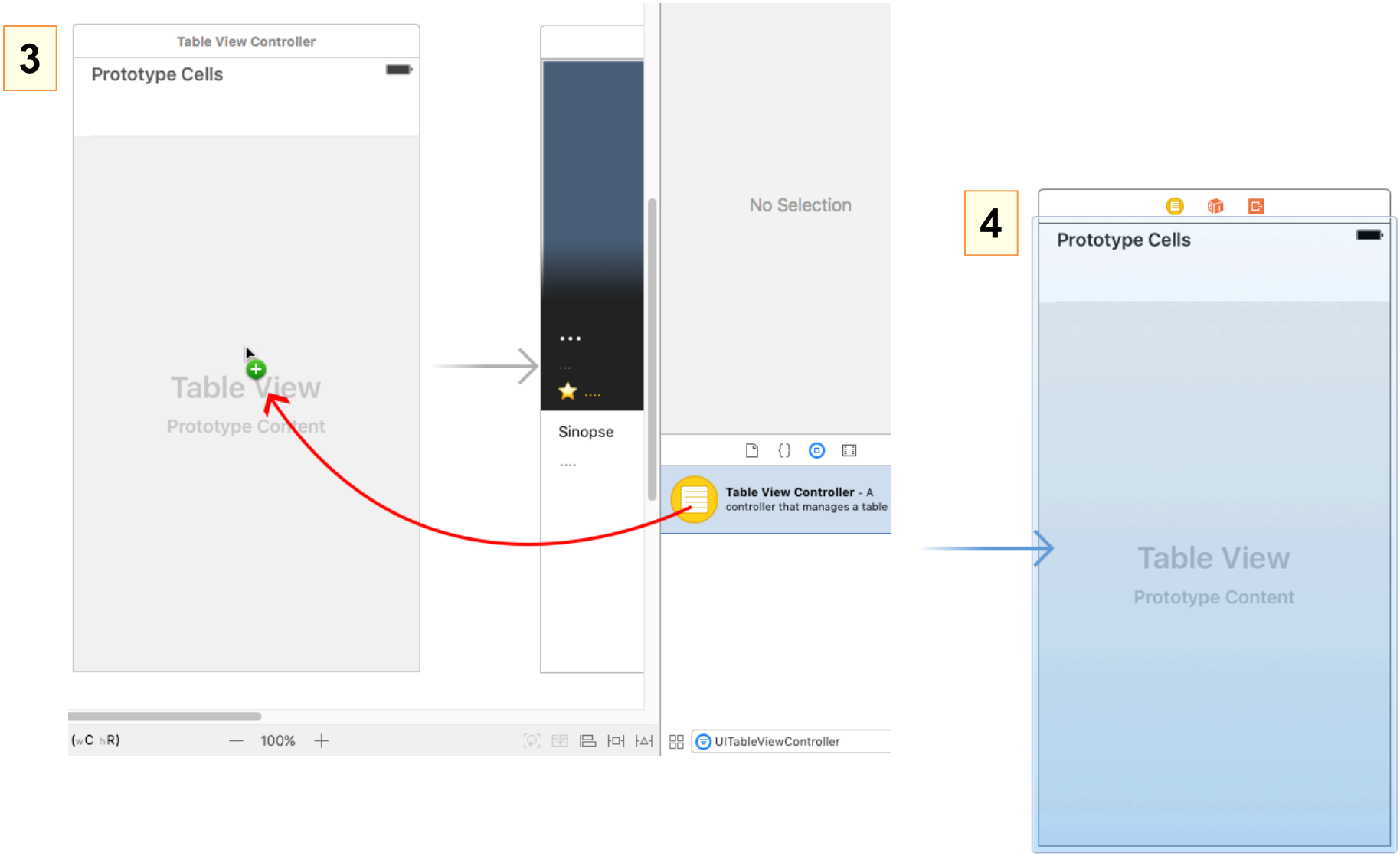
- Um dos componentes mais utilizados no desenvolvimento de aplicações iOS é a UITableView. Com ela, podemos apresentar ao usuário uma lista de informações, possibilitando a seleção e edição das mesmas.
- Existe uma controller desenvolvida especialmente para trabalhar com tableviews, a ***UITableViewController***.
- Esta controller já implementa os protocolos necessários para que possamos alimentar e configurar uma tableview, que são ***UITableViewDataSource*** e ***UITableViewDelegate***.
- Quando criamos uma UITableViewController, ela já nos trás configurada uma tableview para utilizar.

UITableViewController

FIAP

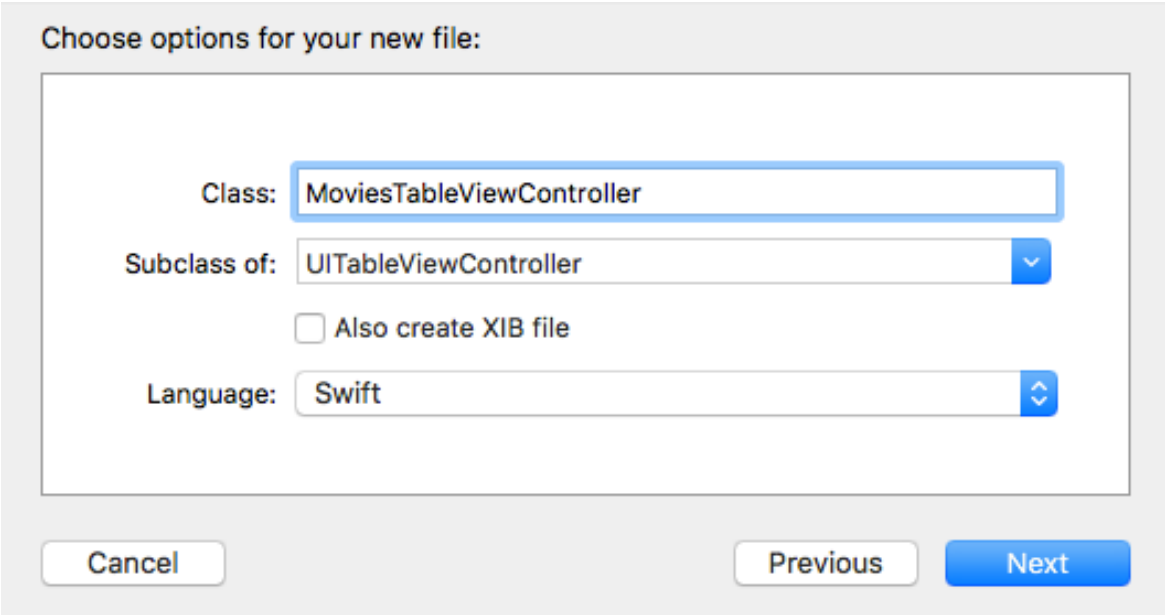
- Adicione uma UITableViewController no seu projeto (***fig. 3***).
- Tire a ***Storyboard Entry Point*** da viewcontroller ***Movies*** e mova para a recém-criada UITableViewController. Dessa maneira, ela será a tela inicial do nosso aplicativo (***fig. 4***).
- Crie uma nova Cocoa Touch Class que irá herdar de UITableViewController e dê o nome de ***MoviesTableViewController*** (***fig. 5***).
- Defina que a UITableViewController adicionada no Storyboard terá como classe a recém-criada ***MovieTableViewController*** (***fig. 6***).

UITableViewController

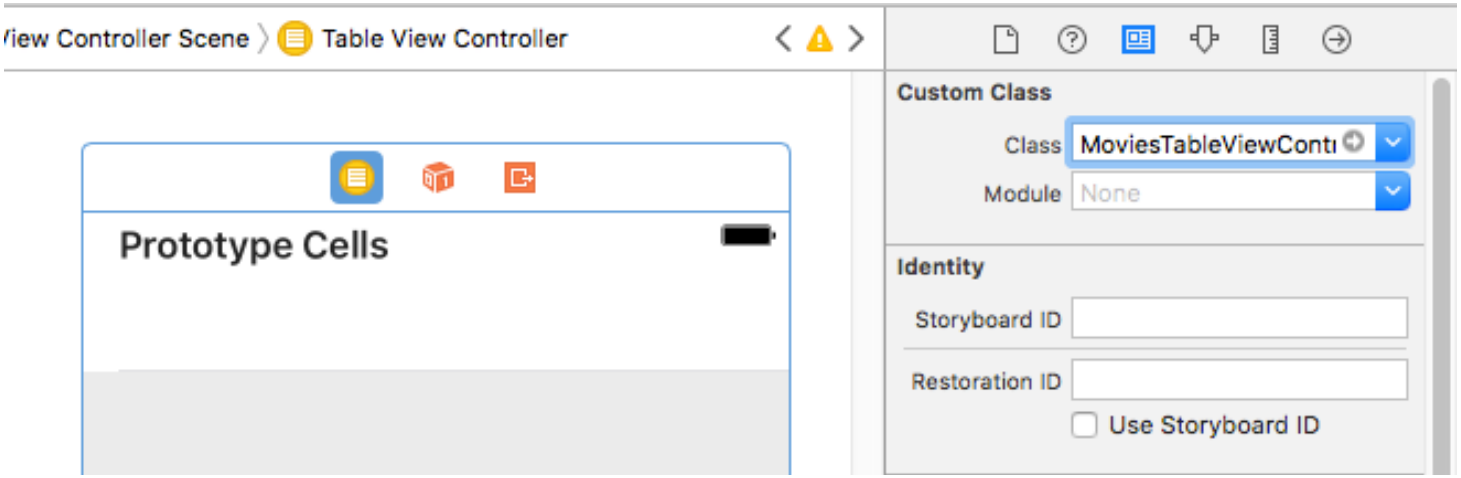


UITableViewController

5



6



JSON

JSON

FIAP

- JSON (JavaScript Object Notation) é um modelo para armazenamento e transmissão de informações no formato texto, amplamente utilizado em aplicações web e mobile devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML.

- Apesar de não termos uma classe ou estrutura nativa em Swift para o formato JSON, podemos utilizar métodos para conversão dados JSON em Dicionários ou Array de Dicionários.

JSON

- Crie um novo arquivo (***File -> New -> File -> Swift file***) chamado Movie.
- Crie a classe ***Movie*** (***fig. 7***). Esta classe será nosso modelo de dados para os filmes que será listado em nossa tabela.

7

```
import Foundation

class Movie {
    var title: String           //Título do filme
    var rating: Double          //Nota
    var summary: String         //Sinopse
    var duration: String        //Duração
    var imageName: String       //Nome base para o arquivo com a imagem do filme
    var categories: [String]!   //Lista de categorias

    //Propriedade computada que retornará o arquivo para a imagem na tabela
    var imageSmall: String { return imageName + "-small.jpg" }

    //Propriedade computada que retornará o arquivo para a imagem de destaque
    var imageWide: String { return imageName + "-wide.jpg" }

    //Propriedade computada que ajudará na apresentação das categorias na tela de detalhes do Filme.
    //O método reduce irá combinar as categorias em uma String
    var categoriesDescription: String {
        return categories.reduce("", {"\($0) | \($1)"})
    }

    //Construtor da classe
    init(title: String, rating: Double, summary: String, duration: String, imageName: String) {
        self.title = title
        self.rating = rating
        self.summary = summary
        self.duration = duration
        self.imageName = imageName
    }
}
```

FIAP

Trabalhando com JSON

- Para trabalharmos com JSON em Swift, usamos a classe ***JSONSerialization***. Esta classe possui métodos tanto para recuperarmos como também criarmos JSON em Swift.

- Quando desejamos recuperar JSON, seja de um arquivo local ou via uma requisição http, usamos o método abaixo, que é um método de classe da classe JSONSerialization, passando como parâmetro o arquivo ***Data*** gerado a partir do JSON e as opções (***JSONSerialization.ReadinOptions***) de como o JSON será lido e convertido em objetos do Foundation (***fig. 8***).

8

```
open class func jsonObject(with data: Data, options opt: JSONSerialization.ReadingOptions = [])
    throws -> Any
```

JSON

FIAP

- Conforme as instruções da tela a seguir (***fig. 9***), iremos realizar as seguintes tarefas:
  - Criar um objeto (***dataSource***) que servirá de fonte de dados para nossa tabela.
  - Criar um método (***loadLocalJSON()***) para a leitura do nosso arquivo JSON. Este método será chamado assim que nossa view for carregada (***viewDidLoad***) e fará o parse do JSON, criando os objetos do tipo ***Movie*** e alimentando nossa lista de filmes (dataSource) que posteriormente será usada para alimentar nossa tabela.
  - Ao término, devemos chamar o método ***.reloadData()*** do objeto tableView. Este é um objeto da classe ***UITableViewControler*** (da qual nossa classe herda) que faz uma referência à nossa tabela no ***StoryBoard***.

JSON

FIAP

9

```
class MoviesTableViewController: UITableViewController {

    //Objeto que conterá o conjunto de Filmes que será usado na tableview
    var dataSource: [Movie] = []

    override func viewDidLoad() {
        super.viewDidLoad()
        loadLocalJSON()
    }

    //Método que fará a leitura (parse) do JSON e alimentará o objeto dataSource
    func loadLocalJSON() {

        //Recuperando URL onde se encontra localmente o arquivo movies.json
        if let jsonURL = Bundle.main.url(forResource: "movies", withExtension: "json") {

            //Gerando arquivo Data a partir do arquivo movies.json
            let data: Data = try! Data(contentsOf: jsonURL)

            //Deserializando o JSON e convertendo em Array de Dicionários. Usamos o objeto data
            //contendo o JSON e .ReadingOptions() como valor padrão de leitura e tratamento desse JSON
            let json = try! JSONSerialization.jsonObject(with: data, options: JSONSerialization.
                ReadingOptions()) as! [[String: Any]]

            for item in json {  //Percorrendo Array de itens do json

                //Recuperando informações do JSON e convertendo em objetos locais
                let title = item["title"] as! String
                let duration = item["duration"] as! String
                let summary = item["summary"] as! String
                let imageName = item["image_name"] as! String
                let rating = item["rating"] as! Double

                //Criando objeto Movie a partir das informações extraídas do JSON
                let movie = Movie(title: title, rating: rating, summary: summary, duration: duration,
                    imageName: imageName)
                movie.categories = item["categories"] as! [String]

                dataSource.append(movie)    //Alimentando array de Movie (dataSource)
            }
            tableView.reloadData()  //Método usado para recarregar os itens de uma tabela
        }
    }
```

UITABLEVIEWDATASOURCE/DELEGATE

UITableViewDataSource/Delegate

FIAP

UITableViewDataSource

- O protocolo ***UITableViewDataSource*** é implementado pela classe UITableViewController e é utilizado para os objetos que farão o papel de mediar o modelo de dados para uma tabela (tableView)
- O DataSource fornece a uma tableView as informações necessárias para a sua construção e modificação.
- Dentre os métodos que este protocolo fornece, estão métodos para definir a quantidade de seções que uma tabela possui, quantas células cada seção irá conter, bem como as informações que cada uma dessas células irá apresentar.
- A maneira como os métodos do DataSource nos fornece informações de qual célula estamos trabalhando é através de um objeto do tipo ***IndexPath***. Este objeto possui propriedades que contém informações de qual seção aquela célula pertence (***.section***) bem como em qual linha (***.row***) ela se encontra.
- Ao utilizarmos uma UITableViewController, os principais métodos de UITableViewDataSource já vem implementados e/ou comentados no código.

UITableViewDataSource/Delegate

FIAP

- Os métodos que obrigatoriamente iremos utilizar para utilizarmos nossa tabela são:
  - *func numberOfSections(in tableView: UITableView) -> Int*
    Solicita quantas seções a tabela irá conter
  - *func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int*
    Solicita quantas linhas (células) cada seção da tabela irá apresentar
  - *func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell*
    Solicita a célula que será apresentada em determinado indexPath (objeto que contém as informações de seção e linha de determinada célula)
- Ao criarmos a célula que será retornada no método *cellForRowAt indexPath* usamos o método *tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)*, que retorna uma célula reutilizável, aumentando assim a performance do nosso App pois cria somente o número necessário de células para a tabela.

UITableViewDataSource/Delegate

FIAP

UITableViewDelegate

- Outro protocolo que a UITableViewController implementa é o UITableViewDelegate, que definem que tal classe é delegate da tableView.
- O delegate de uma tableView é responsável, dentre outras coisas, por gerenciar seleções de células, configurar os cabeçalhos e rodapés das seções e auxilar na exclusão e reordenação das células, dentre outras coisas.
- Alguns exemplos dos métodos são:
  - *func tableView(UITableView, heightForRowAt: IndexPath)*
    Define a altura para determinada célula
  - *func tableView(UITableView, didSelectRowAt: IndexPath)*
    Informa que a célula especificada foi selecionada
  - *func tableView(UITableView, viewForHeaderInSection: Int)*
    Solicita uma view que será utilizada como cabeçalho da seção especificada
  - *func tableView(UITableView, editActionsForRowAt: IndexPath)*
    Solicita que ações serão apresentadas quando for feito o Swipe na célula

UITABLEVIEW / UITABLEVIEWCELL

UITableView / UITableViewCell

FIAP

UITableView

- É um dos principais componentes presentes no desenvolvimento de aplicações iOS.
- Uma ***UITableView*** é um objeto utilizado quando precisamos mostrar uma lista hierárquica de itens, com possibilidade de edição, seleção, reordenação, etc
- Tableviews apresentam seus elementos apenas em uma coluna, com possibilidade de scroll na vertical.
- Cada item de uma tableview é representado por uma ***UITableViewCell***. Estas células podem ser agrupadas em seções para fácil identificação
- Uma tableview precisa ter um objeto que aja como ***delegate*** (responsável por executar ações de configurações e reagir a eventos do usuário) e também um objeto que aja como ***dataSource*** (para configurar a fonte de dados que alimentará a tabela)
- O objeto que servirá como dataSource deve implementar o protocolo ***UITableViewDataSource***, e quem agirá como delegate deve implementar o protocolo ***UITableViewDelegate***. Ao utilizarmos uma ***UITableViewController***, ambos protocolos já são implementados pela mesma.

UITableView / UITableViewCell

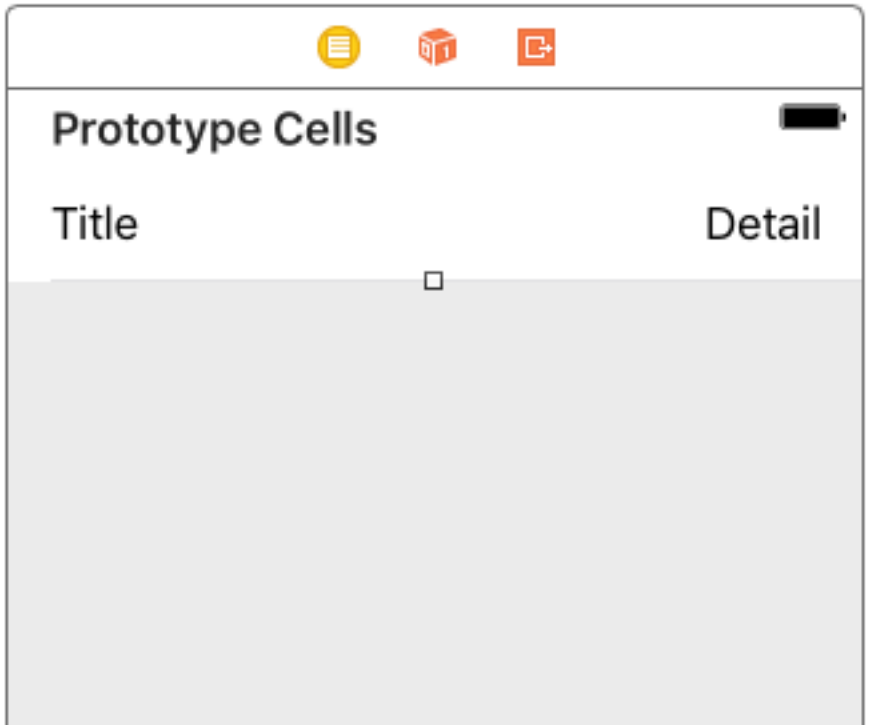
UITableViewCell

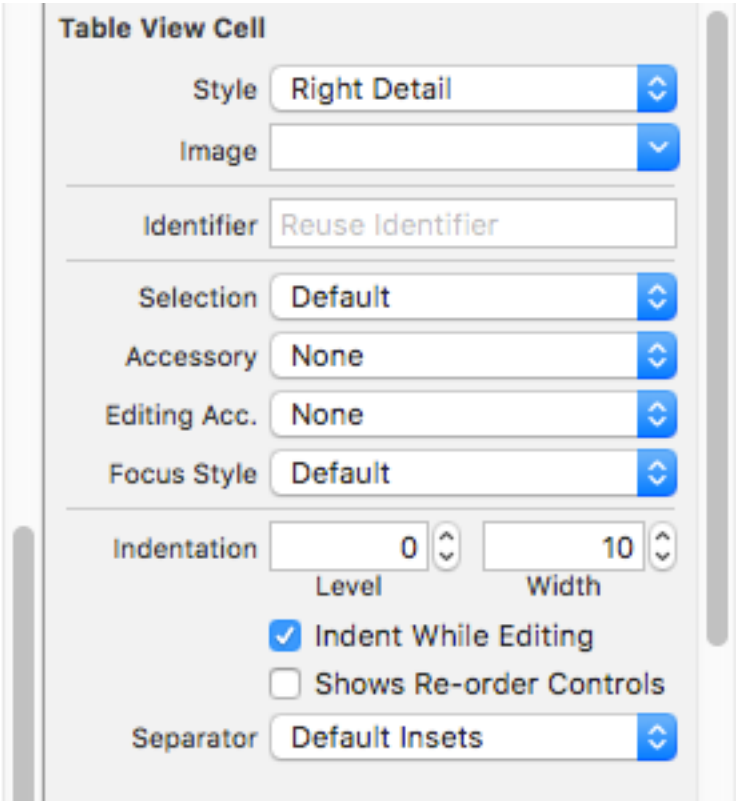
- Cada uma das células (***fig. 10***) apresentadas por uma tableview é representada pela classe ***UITableViewCell***. Esta classe possui métodos e propriedades para definir e gerenciar o conteúdo e aparência dessas células.
- Podemos criar células customizadas ou utilizar estilos pré-definidos, que já fornecem labels e imagens com estilos e posições pré-definidos, que podem ser acessados pelas propriedades ***textLabel*** (para o texto principal), ***detailTextLabel*** (informação secundária) e ***imageView*** (imagem associada à célula)
- Dentre os estilos temos
  - ***Basic***: textLabel alinhado à esquerda
  - ***Right Detail***: textLabel à esquerda e detailTextLabel à direita
  - ***Left Detail***: textLabel à esquerda e detailTextLabel ao lado
  - ***Subtitle***: textLabel à esquerda e detailTextLabel abaixo
- Para trabalharmos com células, precisamos definir seu ***Identifier***, que será um rótulo associado àquele célula protótipo e será usado na criação ou reutilização da mesma. Uma tableview pode ter várias células protótipo, com estilos diferentes. Cada uma precisa possuir um Identifier próprio e único.

UITableView / UITableViewCell

FIAP

10





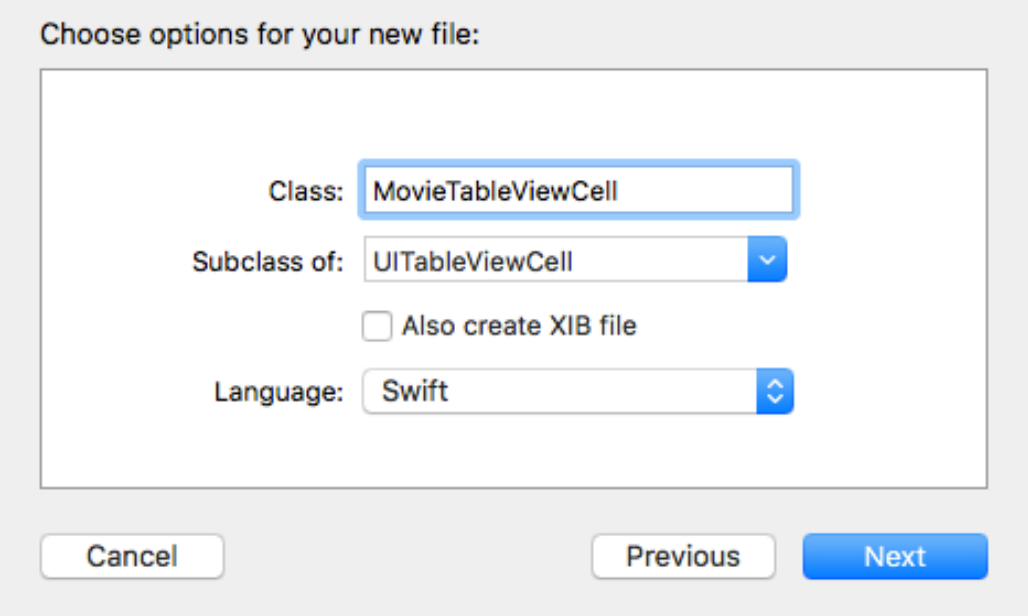
UITableView / UITableViewCell

FIAP

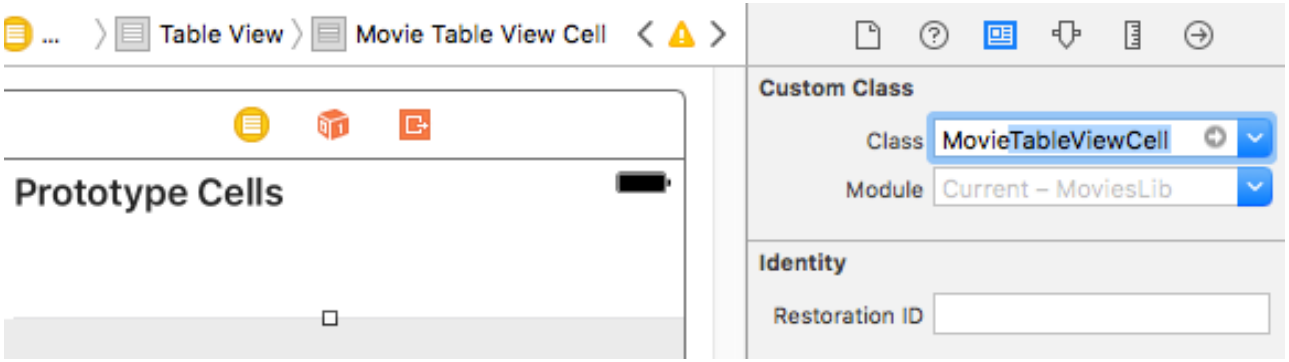
- Em nosso projeto, não iremos utilizar nenhum dos estilos pré-definidos para nossa célula. Vamos criar uma célula com estilo personalizado e para isso precisamos criar uma classe que herde de ***UITableviewCell***. Crie essa classe e chame-a de ***MovieTableViewCell*** (***fig. 10***).
- No StoryBoard, selecione a célula protótipo da nossa tableview e associe a classe recém-criada a esta célula (***fig. 11***).
- Certifique-se que a célula esteja com seu ***Style*** definida como ***Custom*** (***fig. 12***) e modifique o ***Identifier*** desta célula para ***movieCell*** (***fig. 13***).
- Selecione a tableview e defina, no painel ***Size Inspector***, a altura de cada célula, através da propriedade ***Row Height***, como 106 (***fig. 14***).
- Selecione a célula protótipo e altere sua cor de fundo para preto, alterando a propriedade Background (***fig. 15***).

UITableView / UITableViewCell

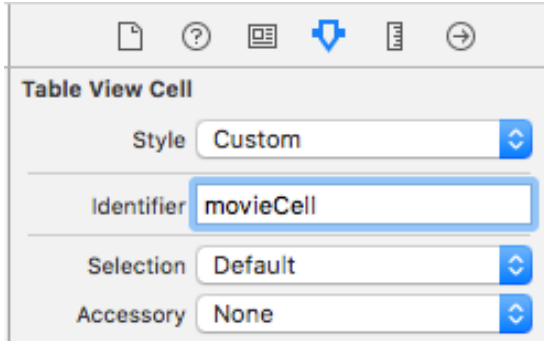
11



12

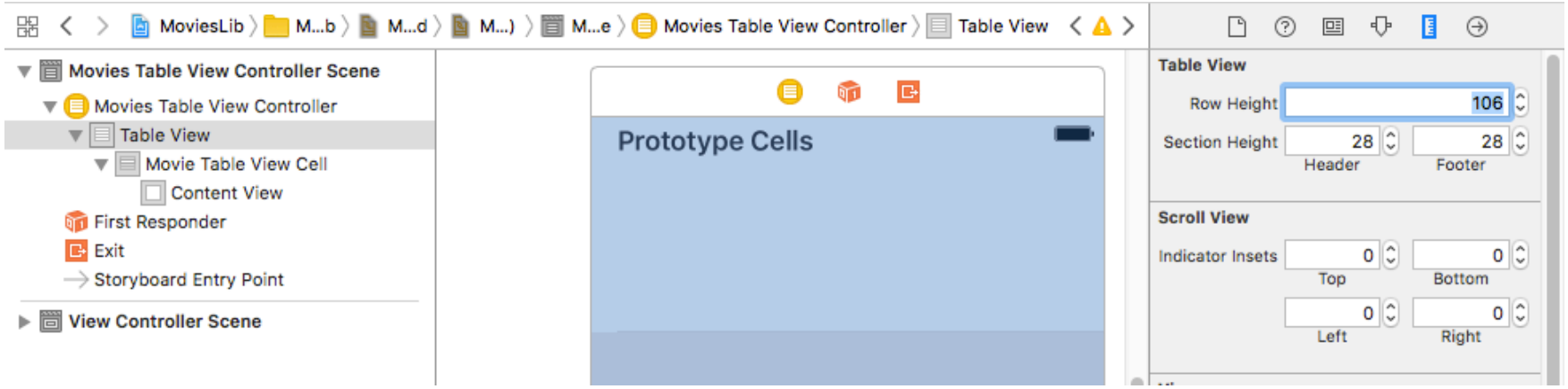


13

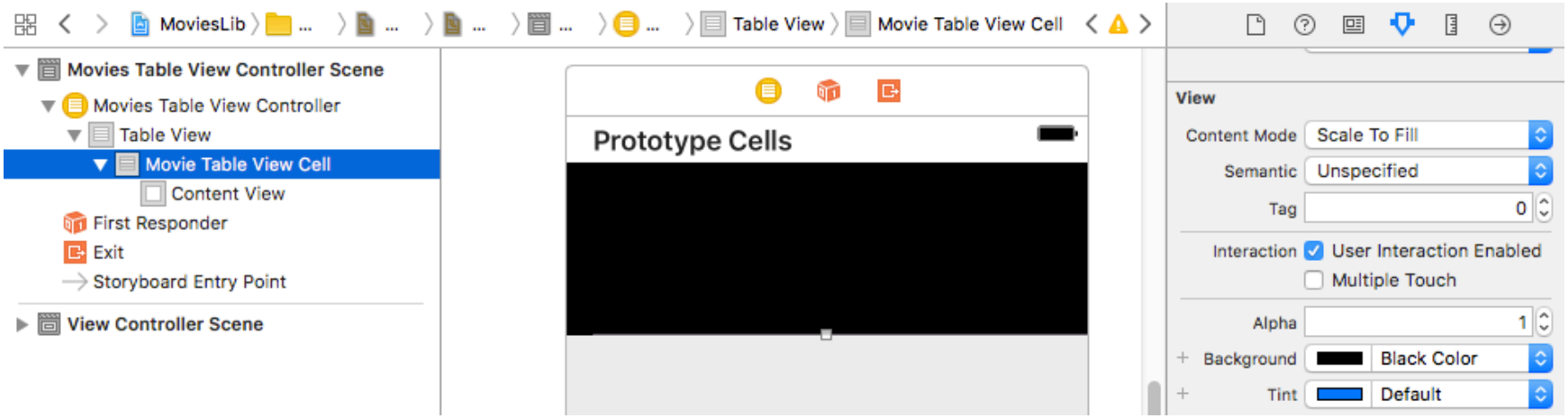


UITableView / UITableViewCell

14



15

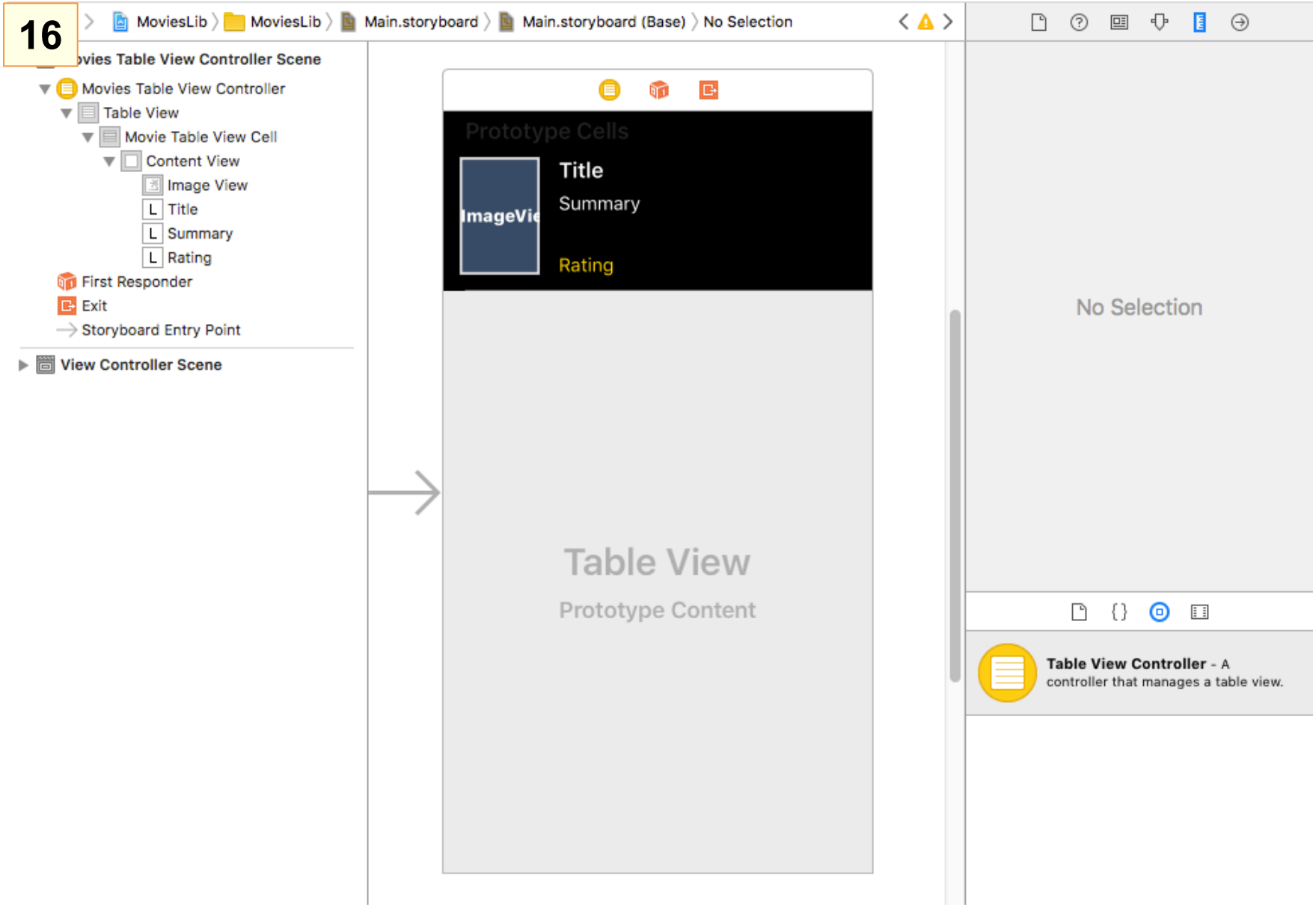


UITableView / UITableViewCell

FIAP

- Insira uma ***UIImageView*** dentro da célula e configure seu frame para ***x***: **12**, ***y***: **6**, ***Width***: **60**, ***Height***: **80**. Defina seu ***Content Mode*** para **Scale To Fill** e marque ***Clip To Bounds***.
- Insira uma ***UILabel*** com o texto **Title**, defina sua **Color** para **Branco** e ***Font*** para **System**, **Semibold**, **16**. Seu frame ficará ***x***: **86**, ***y***: **6**, ***Width***: **224**, ***Height***: **20**.
- Insira outra ***UILabel*** com o texto **Summary**, defina sua ***Color*** para Branco e ***Font*** para **System**, **Regular**, **14**. Seu frame ficará ***x***: **86**, ***y***: **32**, ***Width***: **224**, ***Height***: **17**.
- Adicione uma última ***UILabel*** com o texto **Rating**, ***Color*** para **#FFCE00** e ***Font*** para **System**, **Regular**, **14**. Seu frame ficará ***x***: **86**, ***y***: **78**, ***Width***: **224**, ***Height***: **17**.
- Selecione a tableView e altere sua ***Background*** para **Preto**.
- No final, sua tela ficará como na ***figura 16***.

UITableView / UITableViewCell



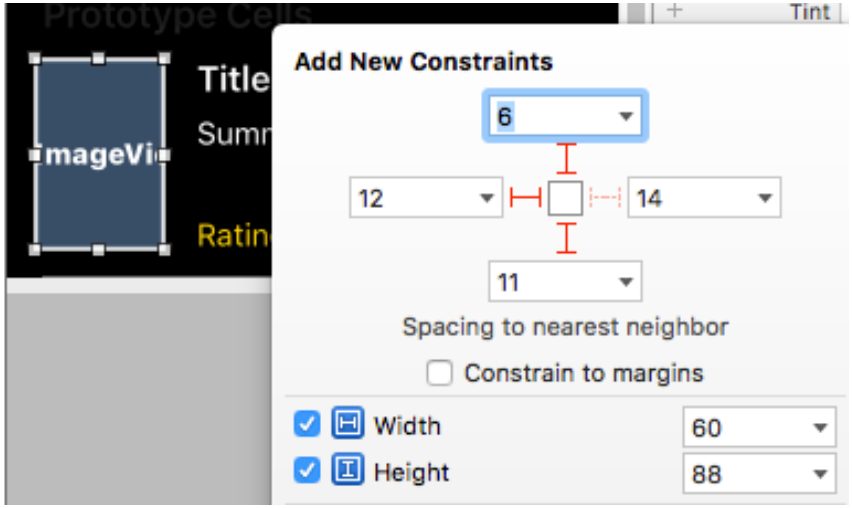
UITableView / UITableViewCell

FIAP

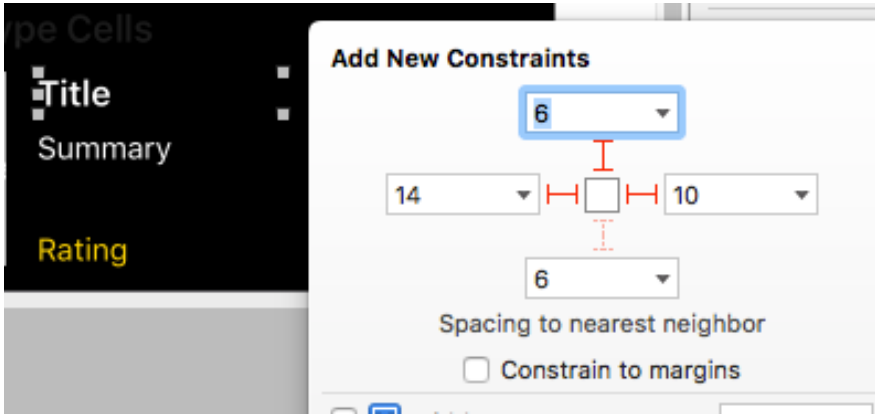
- Adicione constraints na UIImageView conforme a ***figura 17***, na label Title conforme a ***figura 18***, label Summary conforme a ***figura 19*** e label Rating conforme a ***figura 20***.
- Selecione a contraints de bottom da Summary e altere seus valores de modo que o valor de ***Constant*** fique **8** e ***Priority*** fique **250** (***fig. 21***). Com isso, o conteúdo de Summary ficará com uma distância de 8 da label Rating e, por causa de seu grau de prioridade baixo, não entrará em conflito caso este valor precise ser maior (em um cenário onde Summary tenha apenas 1 linha). Assim Rating continuará alinhado na base da célula e Summary estará alinhado no topo, logo abaixo de Title, mantendo uma distância de Rating maior que 8.
- Selecione a constraint de bottom da UIImageView e altere o seu valor de ***Relation*** para **Greater Than or Equal** (***fig. 22***). Desse modo, poderemos ter células com tamanho maior que o tamanho definido porém nunca menor, evitando assim esconder a imagem.

UITableView / UITableViewCell

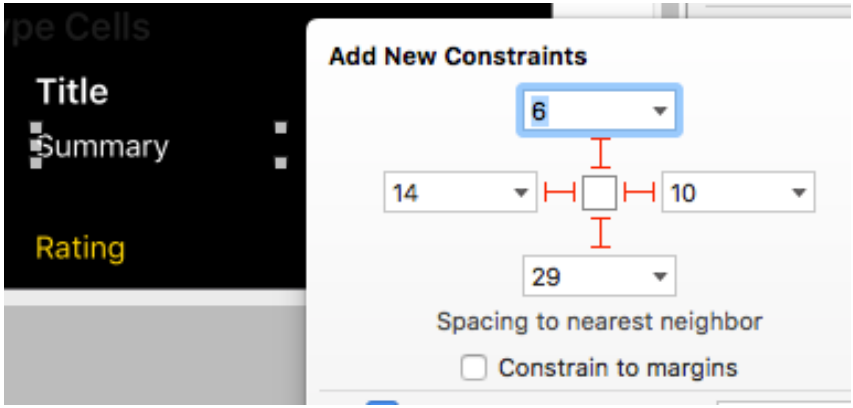
17



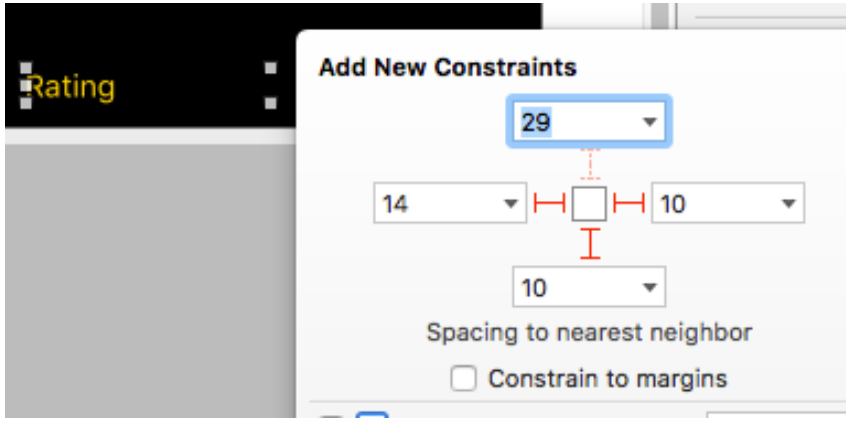
18



19

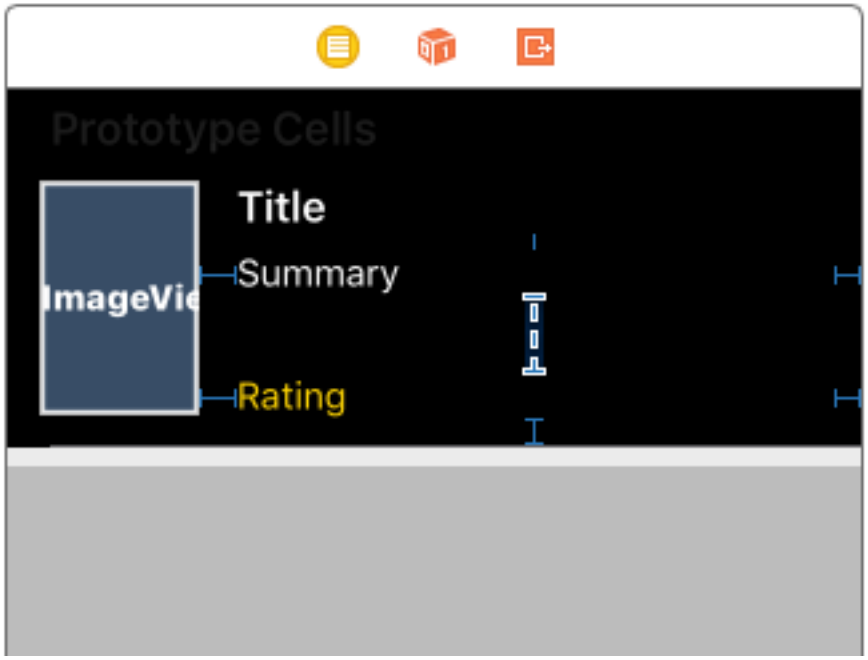


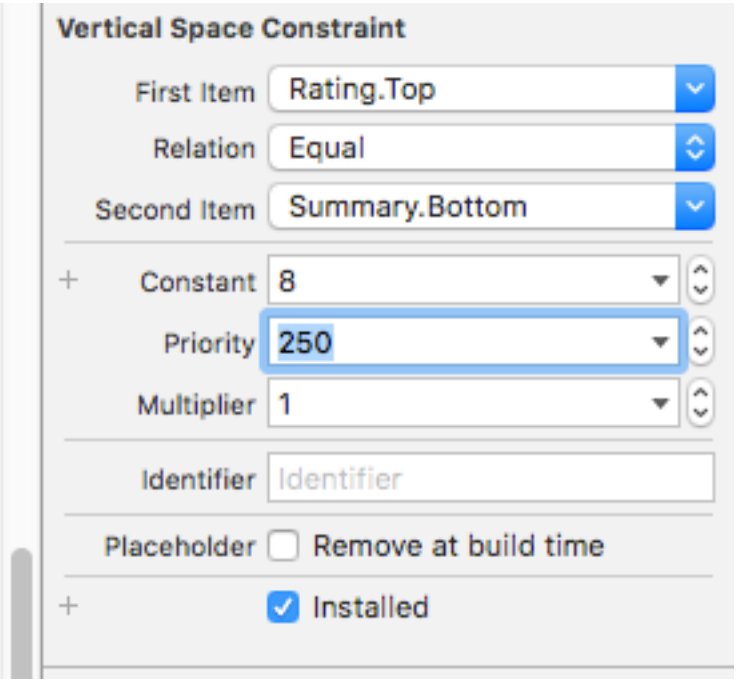
20



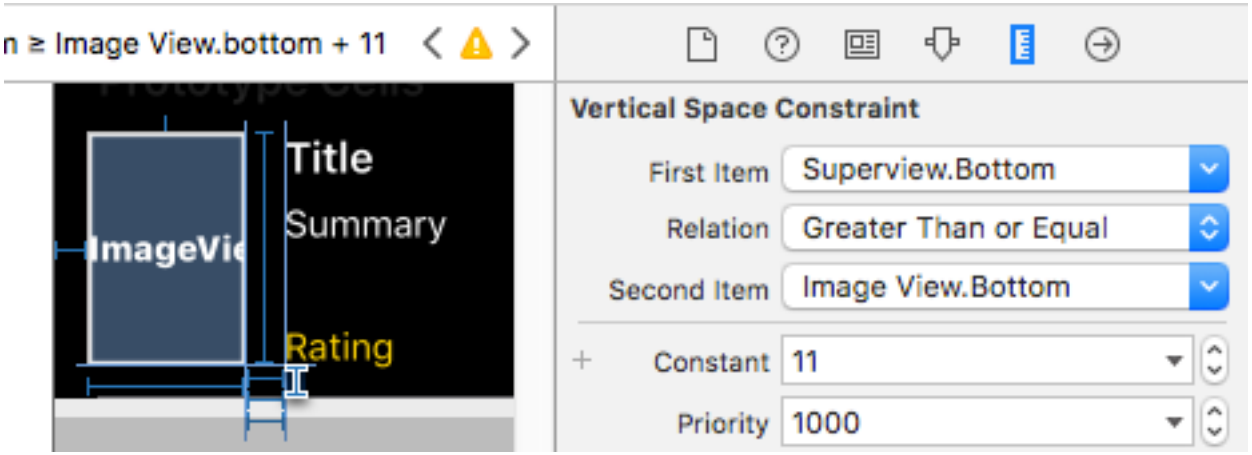
UITableView / UITableViewCell

21





22



UITableView / UITableViewCell

FIAP

- Crie os seguintes ***IBOutlets*** para os elementos da células. Ao entrar no modo ***Assistant Editor***, para garantir que a classe ***MovieTableViewCell*** apareça no lado direito, com a tecla ***option*** apertada, clique sobre o arquivo ***MovieTableViewCell.swift***.

- *@IBOutlet weak var ivPoster: UIImageView* para a **UIImageView**
- *@IBOutlet weak var lbTitle: UILabel!* para a label **Title**
- *@IBOutlet weak var lbSummary: UILabel!* para a label **Summary**
- *@IBOutlet weak var lbRating: UILabel!* para a label **Rating**

UITableView / UITableViewCell

FIAP

- Para que as células fiquem com seus tamanhos variáveis, além da implementação das constraints na parte anterior, precisamos implementar dentro de ***viewDidLoad*** as propriedades ***estimatedRowHeight*** e ***rowHeight*** da nossa tableview conforme a ***figura 23***.

23

```
override func viewDidLoad() {
    super.viewDidLoad()
    loadLocalJSON() //Alimentando nosso dataSource

    tableView.estimatedRowHeight = 106  //Definindo um tamanho base para o cálculo do tamanho final
    tableView.rowHeight = UITableViewAutomaticDimension //Definindo que o tamanho será dinâmico
}
```

- Faça as implementações nos métodos do dataSource da nossa classe ***MoviesTableViewController*** conforme a ***figura 23***. Não esqueça de descomentar o trecho de código que implementa o método "***cellForRowAt***" e alterar seus valores, atentando para o identifier do método "***dequeueReusableCell***".

UITableView / UITableViewCell

FIAP

24

```
// MARK: - Table view data source

//Método que define a quantidade de seções de uma tableView
override func numberOfSections(in tableView: UITableView) -> Int {
    return 1
}

//Método que define a quantidade de células para cada seção de uma tableView
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    //Retornamos o total de itens no nosso dataSource
    return dataSource.count
}

//Método que define a célula que será apresentada em cada linha
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
    UITableViewCell {

    //Definimos o identifier que usamos em nossa célula (movieCell)
    //Fazemos o cast para MovieTableViewCell para que possamos acessar os
    //IBOutlets criados
    let cell = tableView.dequeueReusableCell(withIdentifier: "movieCell", for: indexPath) as!
        MovieTableViewCell

    //Atribuindo os valores de acordo com os dados recuperados de cada Movie
    //Recuperamos o Movie usando a propriedade row do indexPath da célula em questão
    cell.ivPoster.image = UIImage(named: dataSource[indexPath.row].imageSmall)
    cell.lbTitle.text = dataSource[indexPath.row].title
    cell.lbRating.text = "\(dataSource[indexPath.row].rating)"
    cell.lbSummary.text = dataSource[indexPath.row].summary

    return cell
}
```

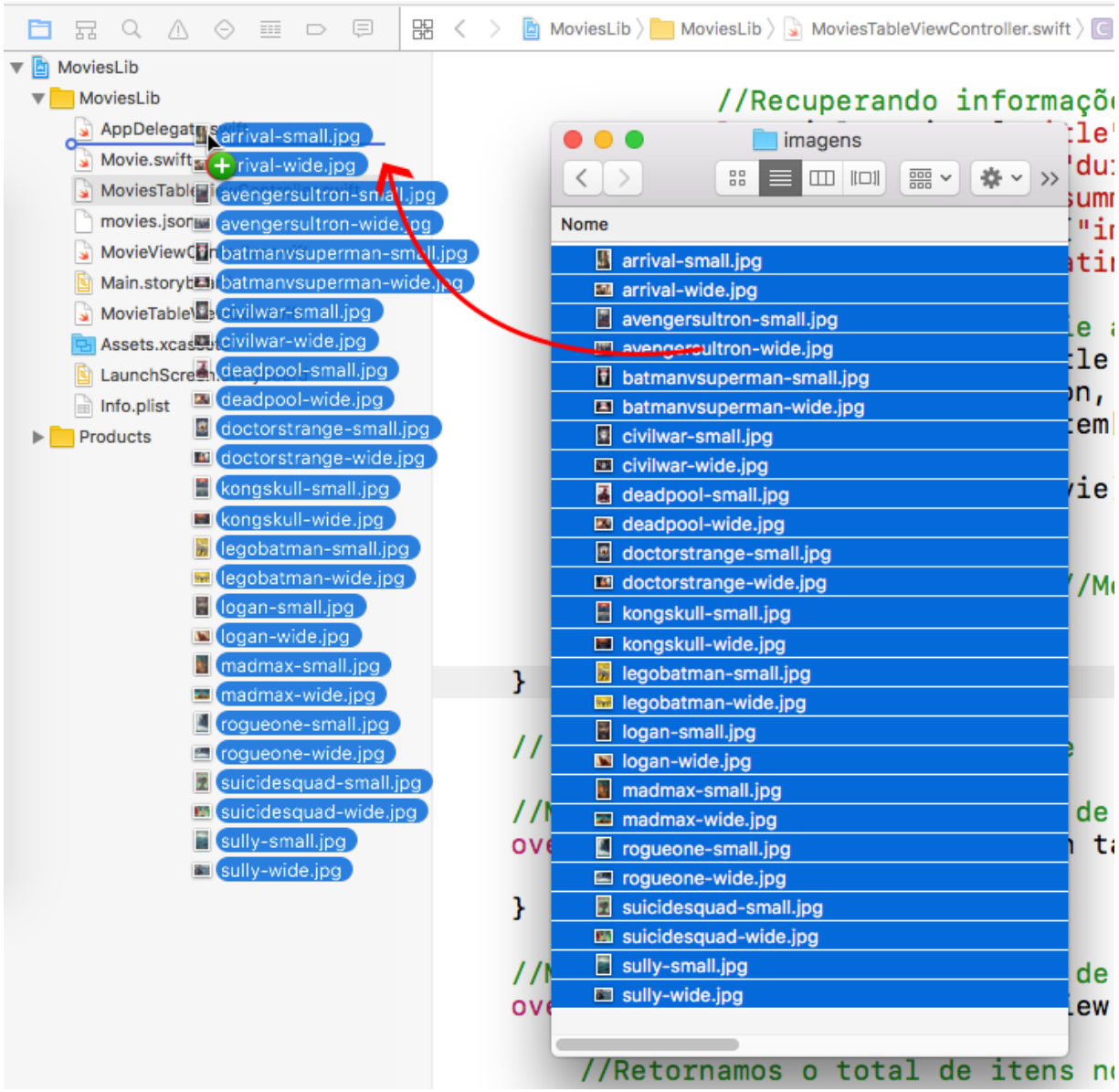
UITableView / UITableViewCell

FIAP

- Precisamos agora inserir as imagens em nosso projeto. Selecione os arquivos da pasta imagens e insira no projeto ***(fig. 25)***.
- Não esqueça de marcar as opções ***Copy items if needed*** e ***Add to targets:***.
- Agora iremos inserir na tableView uma mensagem que irá aparecer caso não tivéssemos filmes. Para isso, crie e configure uma ***UILabel*** no método ***viewDidLoad*** (***fig. 26***).
- No método *numberOfRowsInSection*, do caso o count do nosso dataSource seja 0, esta label será inserida como ***backgroundView*** da tableView. Do contrário, setamos como ***nil*** (***fig. 27***).
- Para que nossa tabela não fique com as linhas que separam as células quando não tiverem filmes, adicione uma ***UIView*** como sendo footer da tabela (***fig. 28***) e logo após, defina a altura dessa viewpara 0 (***fig. 29***). Dessa forma, esa view não irá aparecer mas servirá para esconder as linhas caso nossa tabela tenha poucos ou nenhum filme.

UITableView / UITableViewCell

25



UITableView / UITableViewCell

FIAP

26

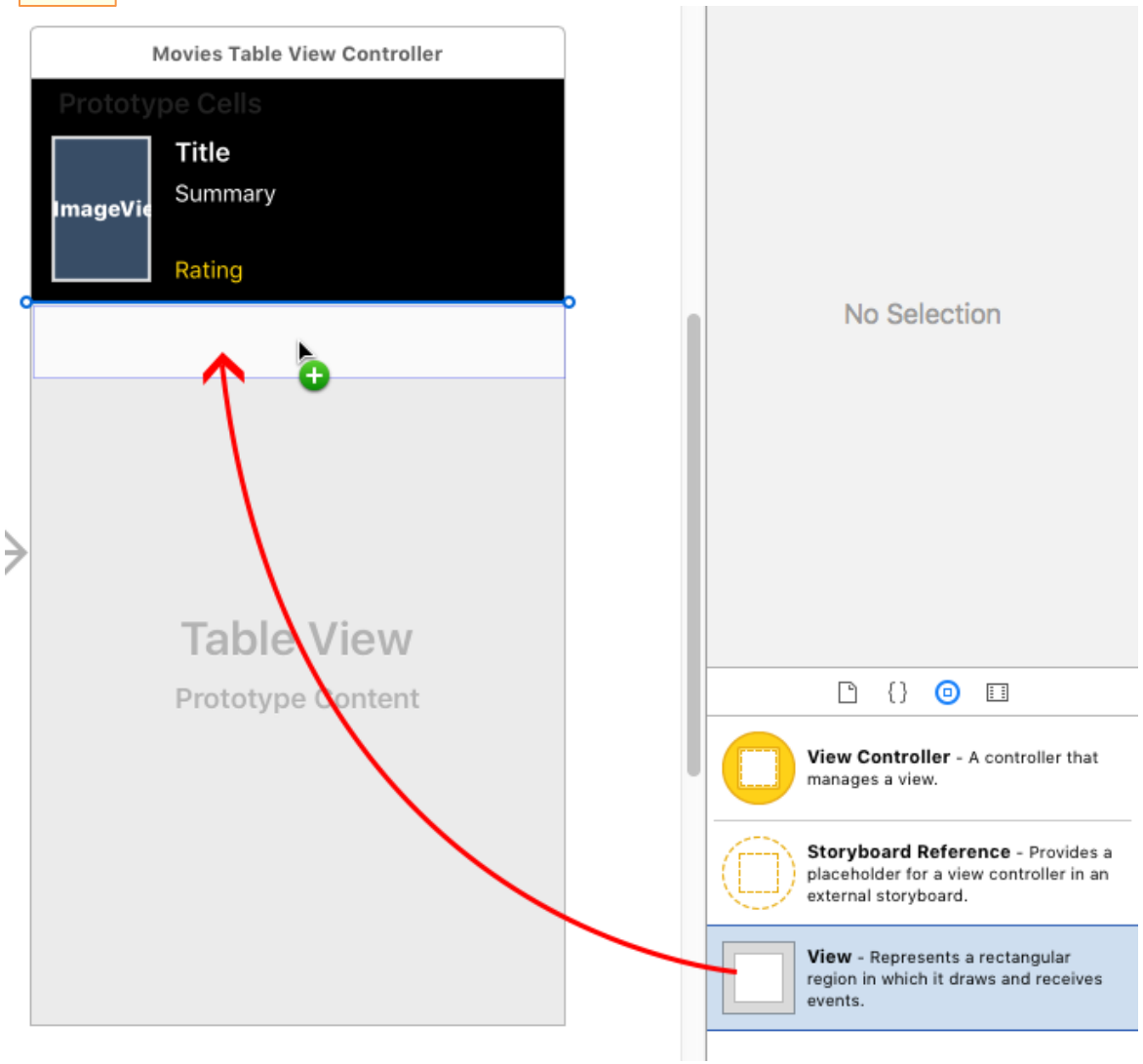
```
class MoviesTableViewController: UITableViewController {

    //Objeto que conterá o conjunto de Filmes que será usado na tableview
    var dataSource: [Movie] = []

    //Criando nossa label que será a backgroundView da tabela
    var label = UILabel(frame: CGRect(x: 0, y: 0, width: 200, height: 22))

    override func viewDidLoad() {
        super.viewDidLoad()
        loadLocalJSON() //Alimentando nosso dataSource

        tableView.estimatedRowHeight = 106  //Definindo um tamanho base para o cálculo do tamanho final
        tableView.rowHeight = UITableViewAutomaticDimension //Definindo que o tamanho será dinâmico

        //Definindo os valores das propriedades da lavel
        label.text = "Sem filmes"
        label.textAlignment = .center
        label.textColor = .white
    }
```

27

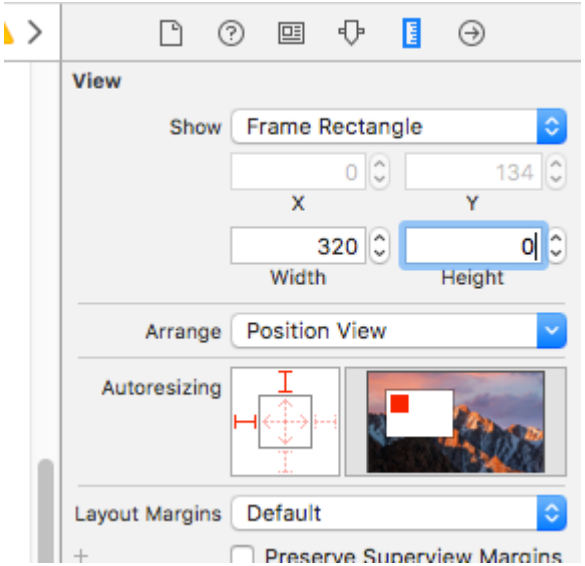
```
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

    //Caso nosso dataSource seja 0, teremos a label aparecendo.
    tableView.backgroundView = dataSource.count == 0 ? label : nil

    return dataSource.count //Retornamos o total de itens no nosso dataSource
}
```

UITableView / UITableViewCell

28



29



UITableView / UITableViewCell

FIAP

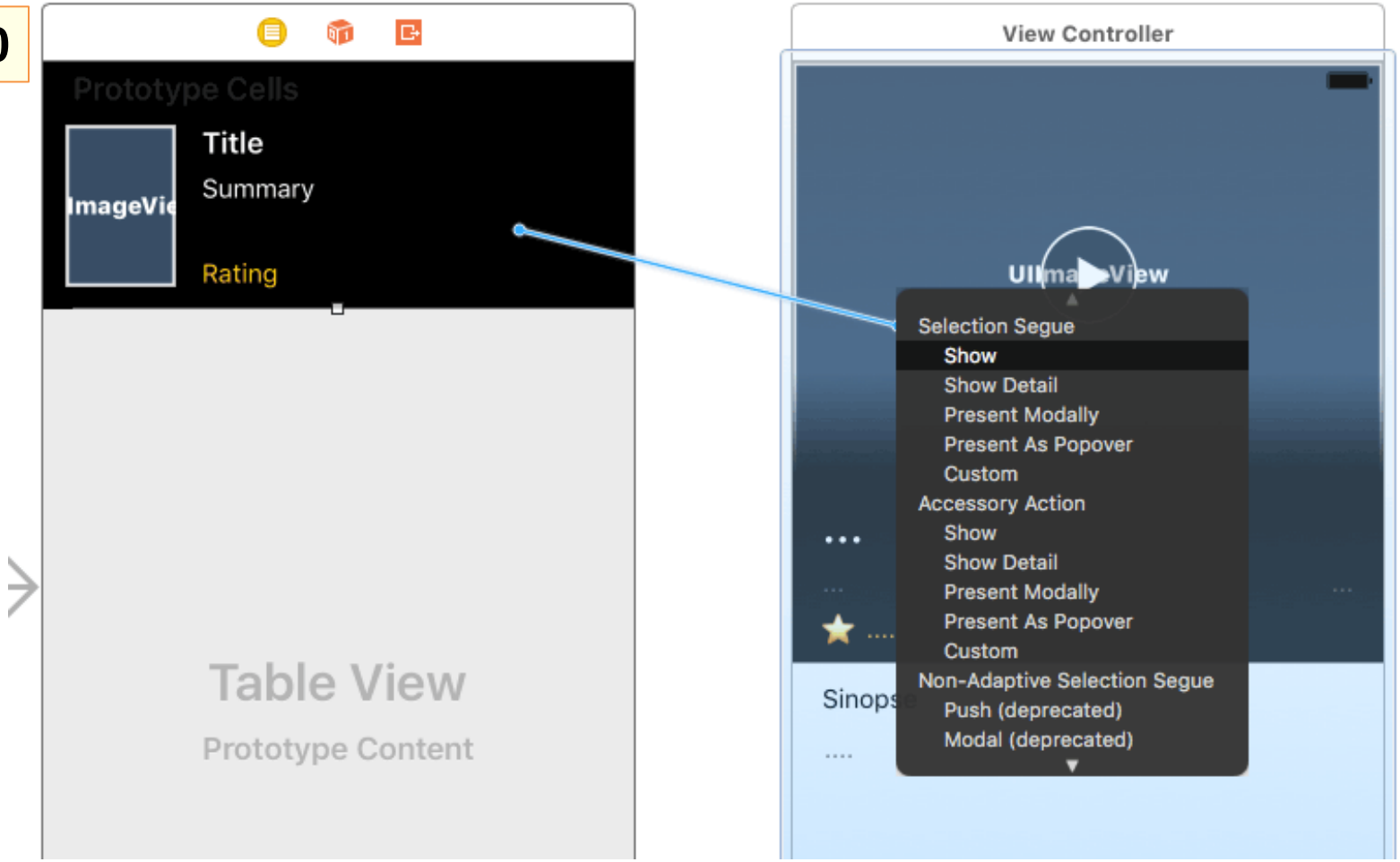
- Associe a classe ***MovieViewController*** à respectiva ViewController no StoryBoard (a que possui os detalhes do filme). Todos os IBOutlets já estão criados e serão associados. Devemos agora passar as informações do filme escolhido para a MovieViewController. Crie uma ***segue*** do tipo ***show*** da célula até a ***MovieViewController*** (***fig. 30***).

- Na classe ***MoviesTableViewController***, implemente o método *prepare(for segue: UIStoryboardSegue, sender: Any?)* e envie o filme selecionado conforme abaixo.

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    let vc = segue.destination as! MovieViewController
    vc.movie = dataSource[tableView.indexPathForSelectedRow!.row]
}
```

- Na classe ***MovieViewController***, crie a variável ***movie*** que irá receber o filme selecionado e implemente o código para alimentar os ***IBOutlets*** com as informações do filme, bem como o método *touchesBegan* para podermos voltar à tela clicando em alguma área da tela anterior (***fig. 31***).

UITableView / UITableViewCell



UITableView / UITableViewCell

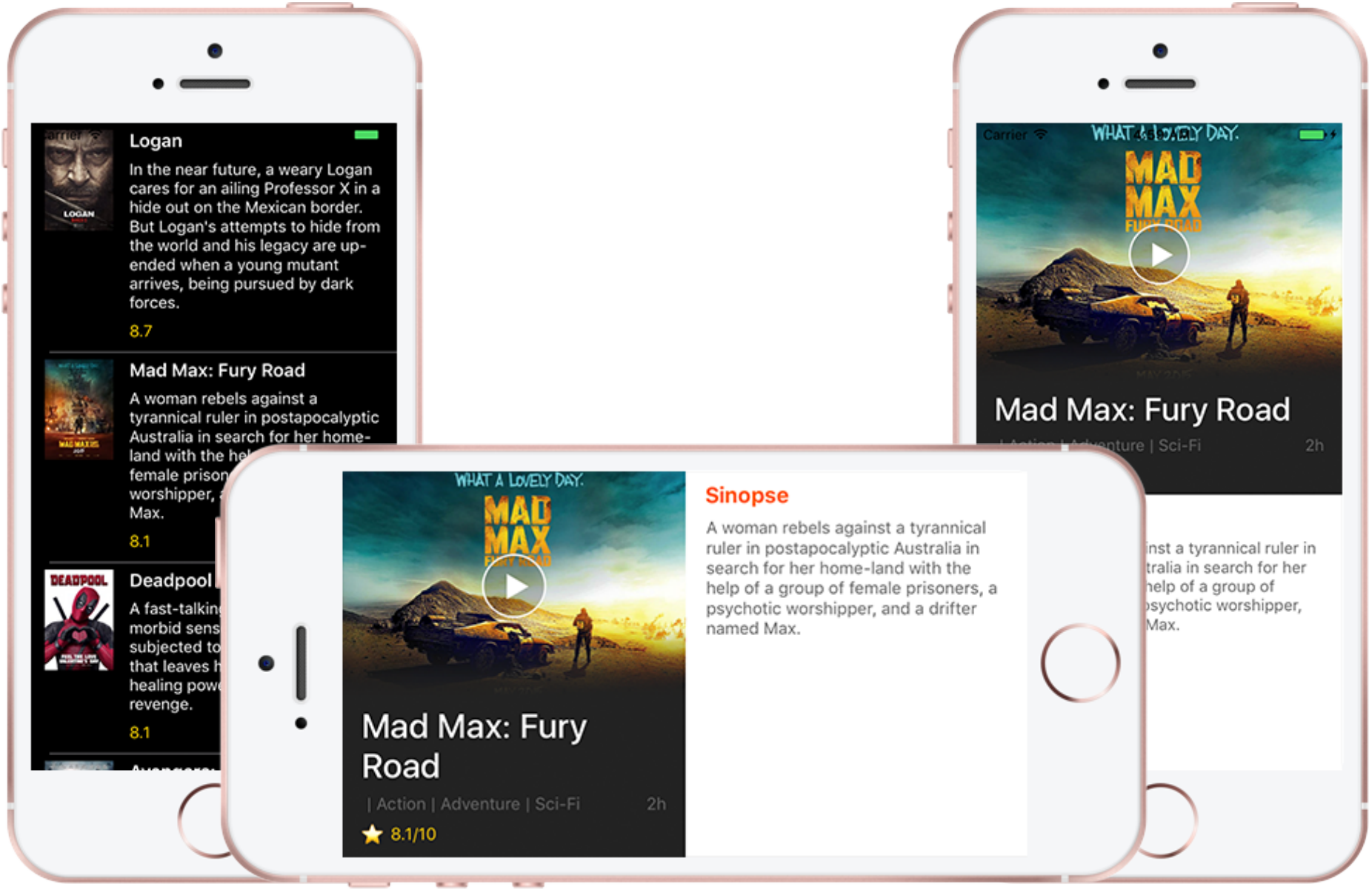
FIAP

31

```
class MovieViewController: UIViewController {

    // MARK: IBOutlets
    @IBOutlet weak var ivPoster: UIImageView!
    @IBOutlet weak var lbTitle: UILabel!
    @IBOutlet weak var lbGenre: UILabel!
    @IBOutlet weak var lbDuration: UILabel!
    @IBOutlet weak var lbScore: UILabel!
    @IBOutlet weak var tvSinopsis: UITextView!
    @IBOutlet weak var lcButtonX: NSLayoutConstraint!

    //Variável que receberá o filme selecionado na tabela
    var movie: Movie!

    // MARK: Super Methods
    override func viewDidLoad() {
        super.viewDidLoad()

        //Alimentando as IBOutlets com as informações dos filmes
        ivPoster.image = UIImage(named: movie.imageWide)
        lbTitle.text = movie.title
        lbGenre.text = movie.categoriesDescription
        lbDuration.text = movie.duration
        lbScore.text = "⭐ \(movie.rating)/10"
        tvSinopsis.text = movie.summary
    }

    //Dessa forma, podemos voltar à tela anterior
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        dismiss(animated: true, completion: nil)
    }
}
```

UITableView / UITableViewCell

FIAP



FIAP

MBA+