



PROF. RENÊ XAVIER

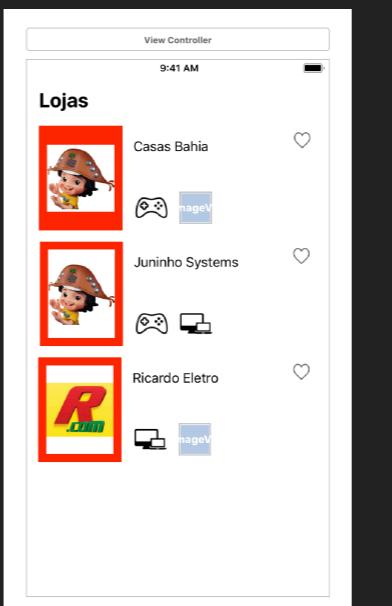
**DESENVOLVIMENTO PARA IOS 11 COM
SWIFT 4**

ONDE ENCONTRAR O MATERIAL?

[HTTPS://GITHUB.COM/RENEFX/IOS-2018-01](https://github.com/renefx/ios-2018-01)

GITHUB
ALÉM DO TRADICIONAL BLACKBOARD DO IESB

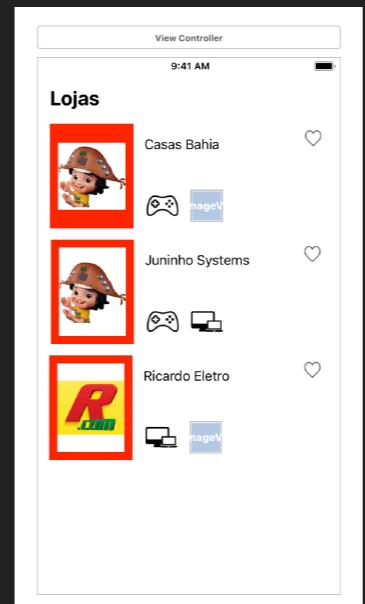
CHEGA DE CRIAR VÁRIOS BAHIANINHOS



SE VAMOS COLOCAR UM SCROLL, VAMOS USAR UMA...



ScrollView Provides a mechanism to display content that is larger than the size of the application's window.



COMO UMA SCROLLVIEW DEVE SER USADA



O QUE VAMOS FAZER HOJE?

AGENDA

- ▶ UITableView e UITableViewController
- ▶ Células personalizadas
- ▶ Prática



TABLEVIEW CONTROLLER OU TABLEVIEW?



Table View Controller - A controller that manages a table view.



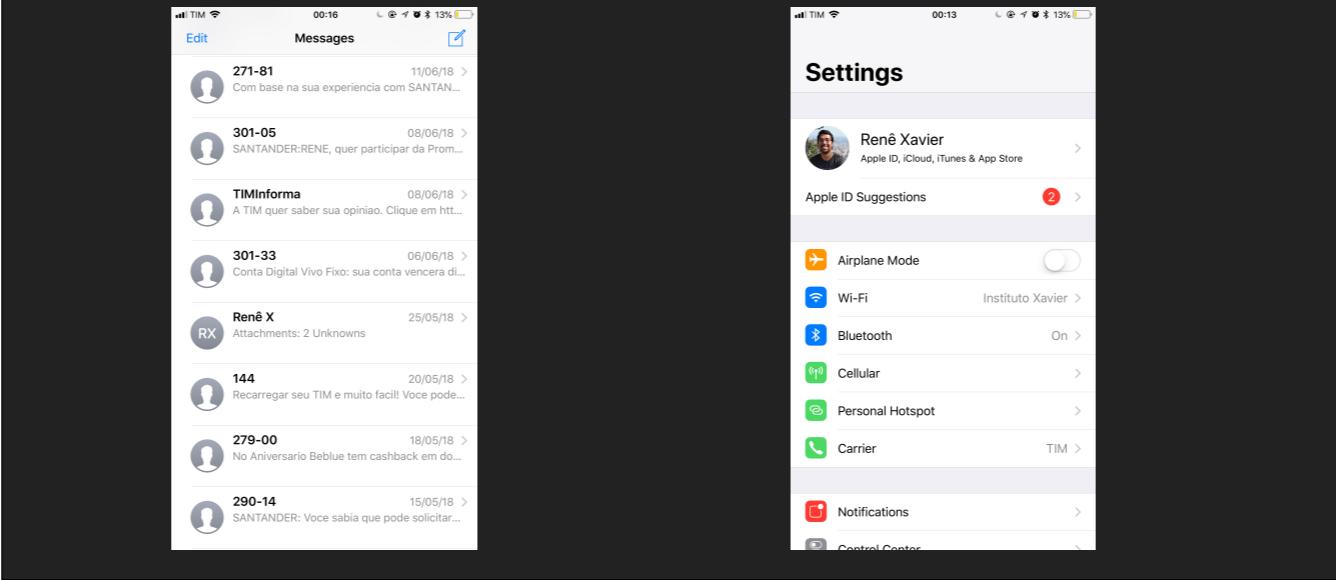
Table View - Displays data in a list of plain, sectioned, or grouped rows.



TABLEVIEW CONTROLLER

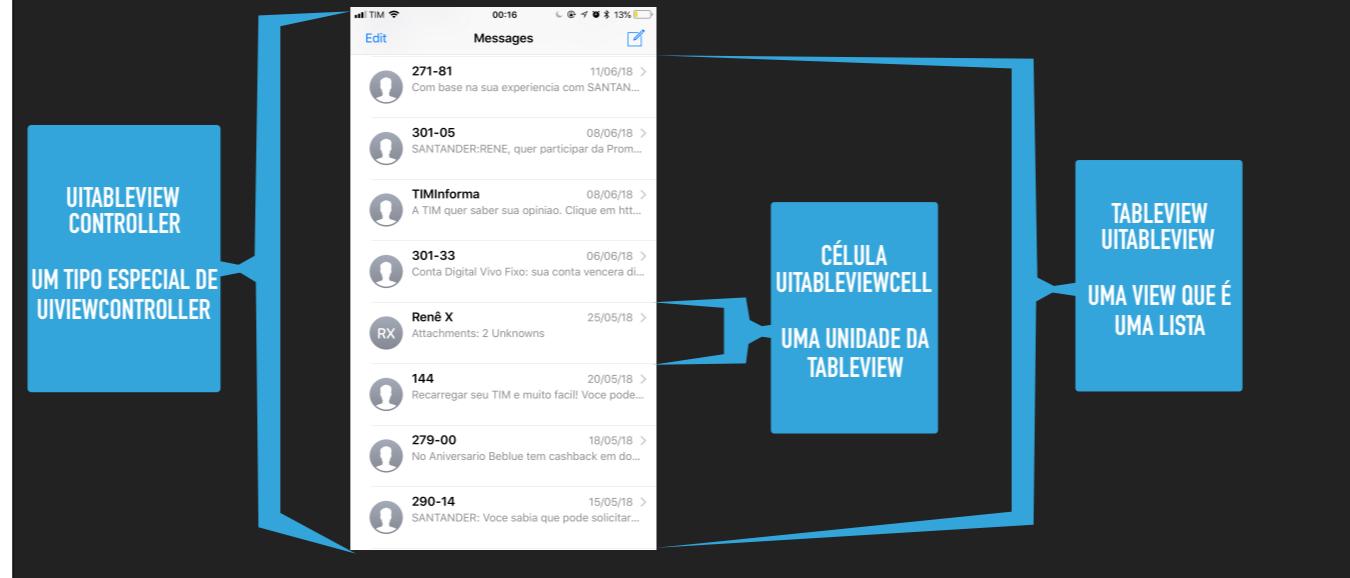
TABLEVIEW CONTROLLER

► Uma tela completa por uma lista (TableView)



TABLEVIEW CONTROLLER

► Vamos a sua anatomia:



TABLEVIEW CONTROLLER

COM OU SEM SEÇÕES!?

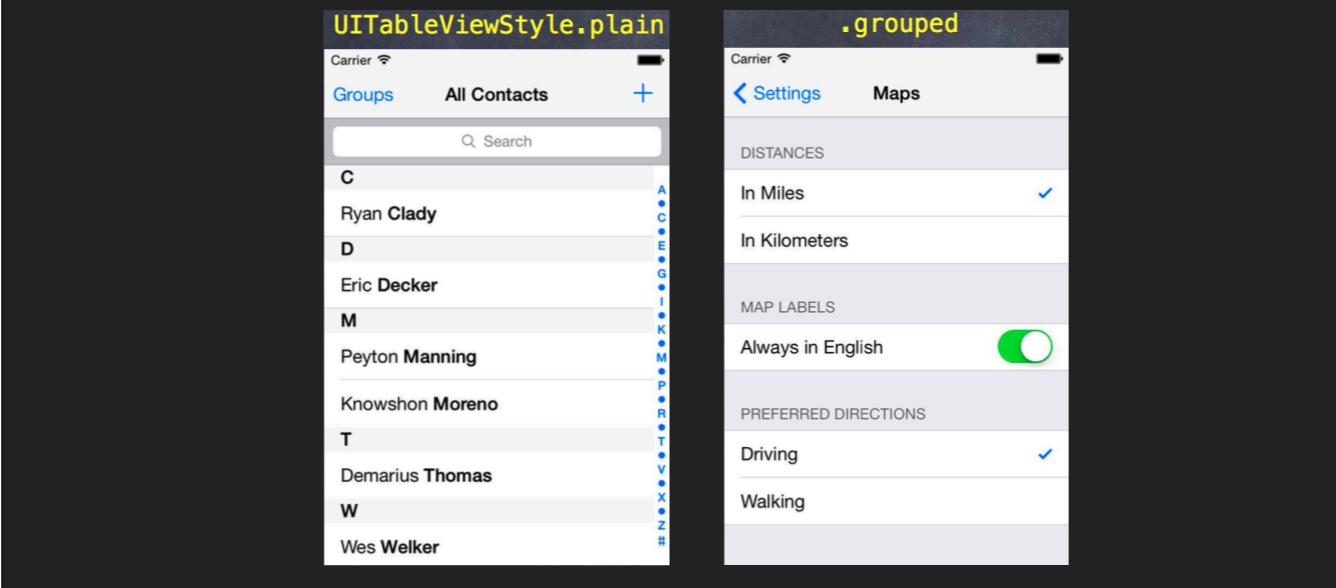
SEM
SEÇÕES



COM
SEÇÕES



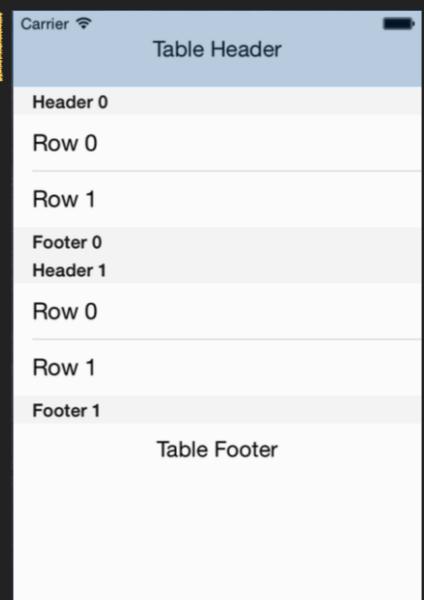
TABLEVIEW CONTROLLER



TABLEVIEW CONTROLLER

PLAIN STYLE

CABEÇALHO DA TABELA
(TABLE HEADER)



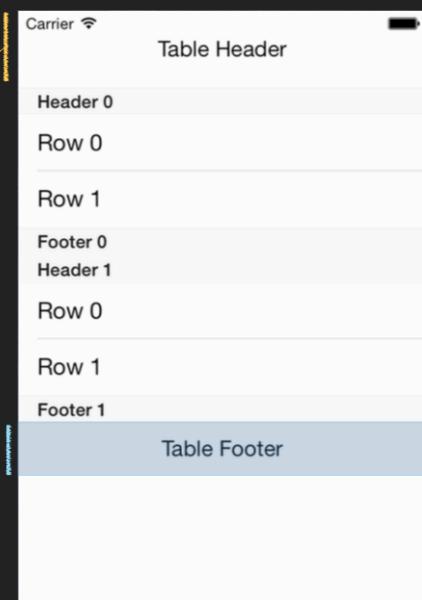
var tableHeaderView: UIView

TABLEVIEW CONTROLLER

PLAIN STYLE

CABEÇALHO DA TABELA
(TABLE HEADER)

RODAPÉ DA TABELA
(TABLE FOOTER)



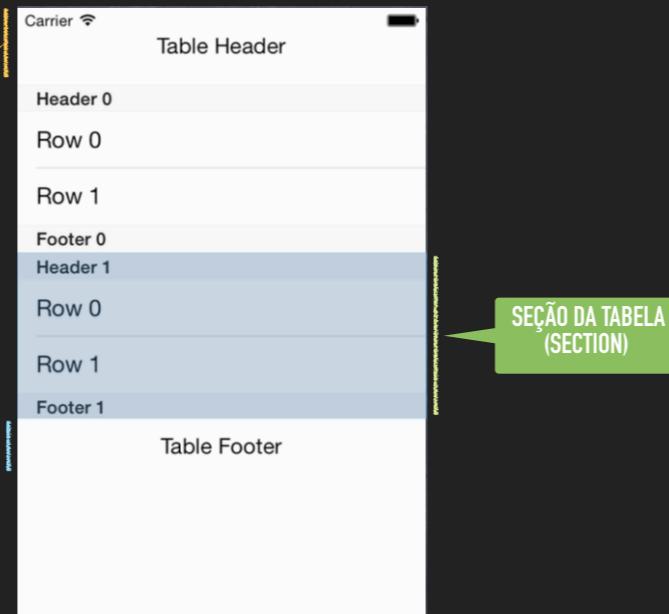
```
var tableFooterView: UIView
```

TABLEVIEW CONTROLLER

PLAIN STYLE

CABEÇALHO DA TABELA
(TABLE HEADER)

RODAPÉ DA TABELA
(TABLE FOOTER)



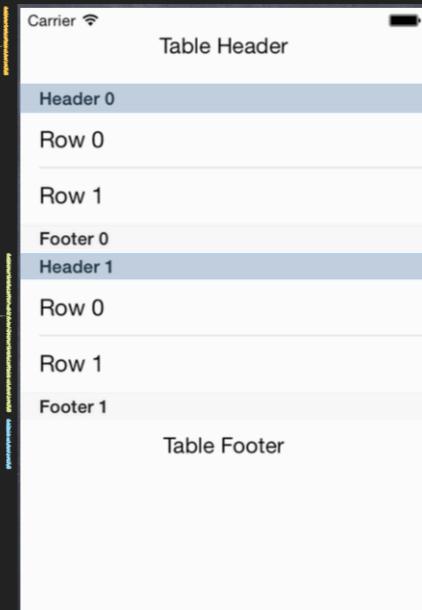
TABLEVIEW CONTROLLER

PLAIN STYLE

CABEÇALHO DA TABELA
(TABLE HEADER)

SEÇÃO DA TABELA
(SECTION)

RODAPÉ DA TABELA
(TABLE FOOTER)



CABEÇALHOS DA SEÇÃO
(SECTION HEADER)

TABLEVIEW CONTROLLER

PLAIN STYLE

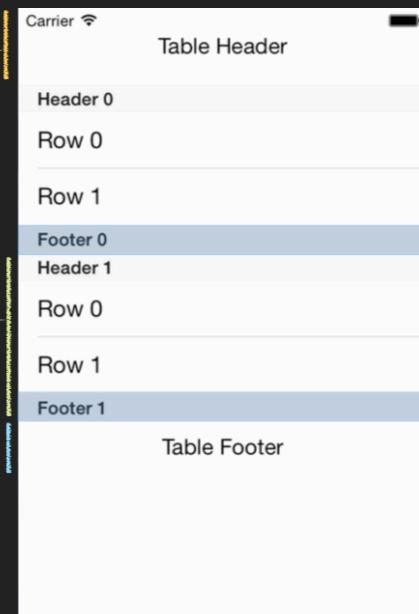
CABEÇALHO DA TABELA
(TABLE HEADER)

SEÇÃO DA TABELA
(SECTION)

RODAPÉ DA TABELA
(TABLE FOOTER)

CABEÇALHOS DA SEÇÃO
(SECTION HEADER)

RODAPÉS DA SEÇÃO
(SECTION FOOTER)



TABLEVIEW CONTROLLER

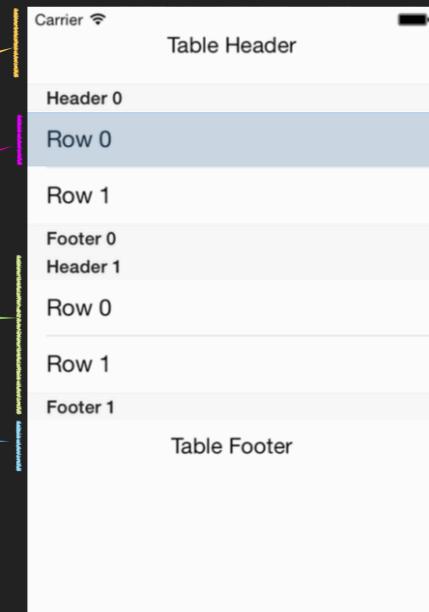
PLAIN STYLE

CABEÇALHO DA TABELA
(TABLE HEADER)

CÉLULA DA TABELA
(TABLE CELL)

SEÇÃO DA TABELA
(SECTION)

RODAPÉ DA TABELA
(TABLE FOOTER)



CABEÇALHOS DA SEÇÃO
(SECTION HEADER)

RODAPÉS DA SEÇÃO
(SECTION FOOTER)

TABLEVIEW CONTROLLER

GROUPED STYLE

CABEÇALHO DA TABELA
(TABLE HEADER)

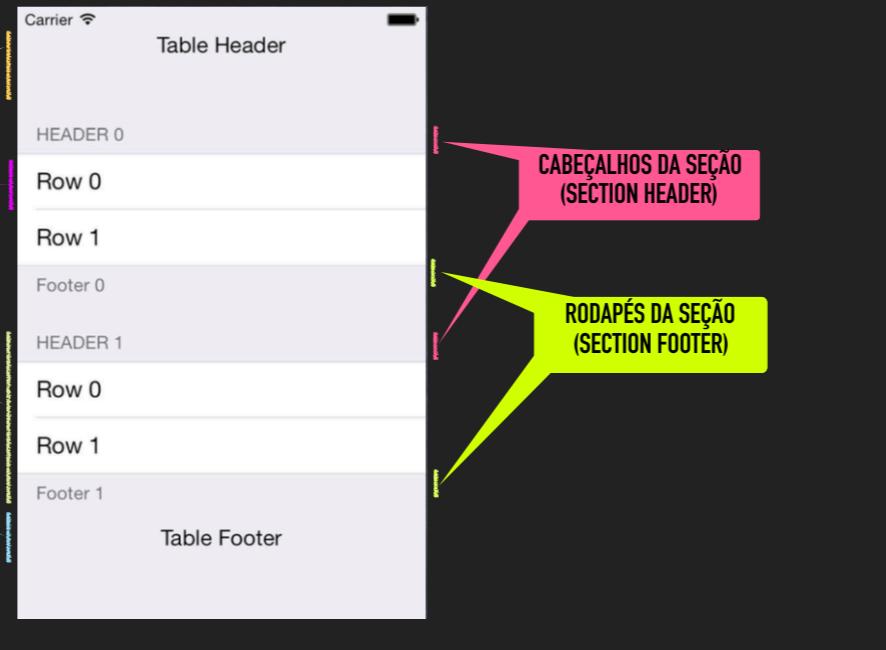
CÉLULA DA TABELA
(TABLE CELL)

SEÇÃO DA TABELA
(SECTION)

RODAPÉ DA TABELA
(TABLE FOOTER)

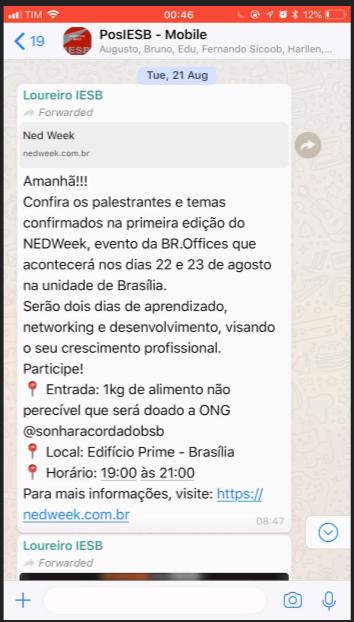
CABEÇALHOS DA SEÇÃO
(SECTION HEADER)

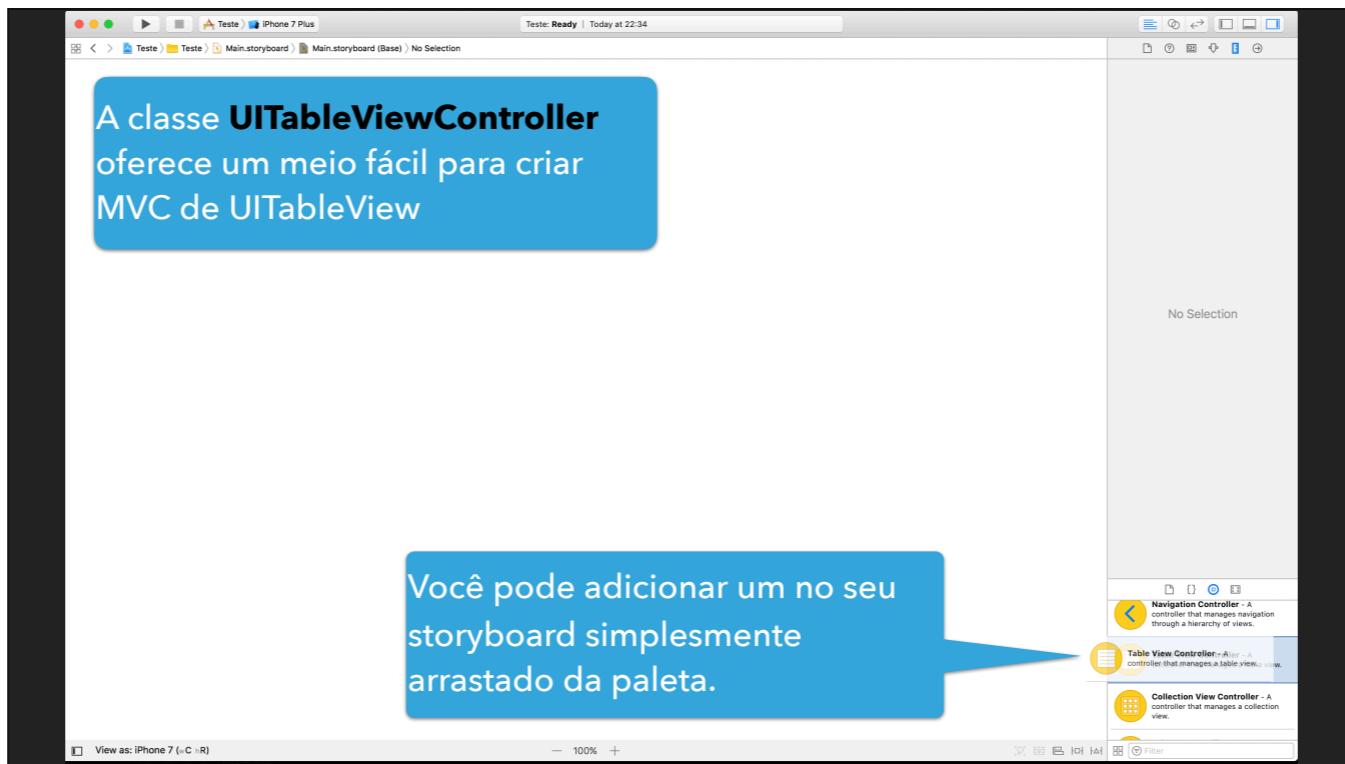
RODAPÉS DA SEÇÃO
(SECTION FOOTER)

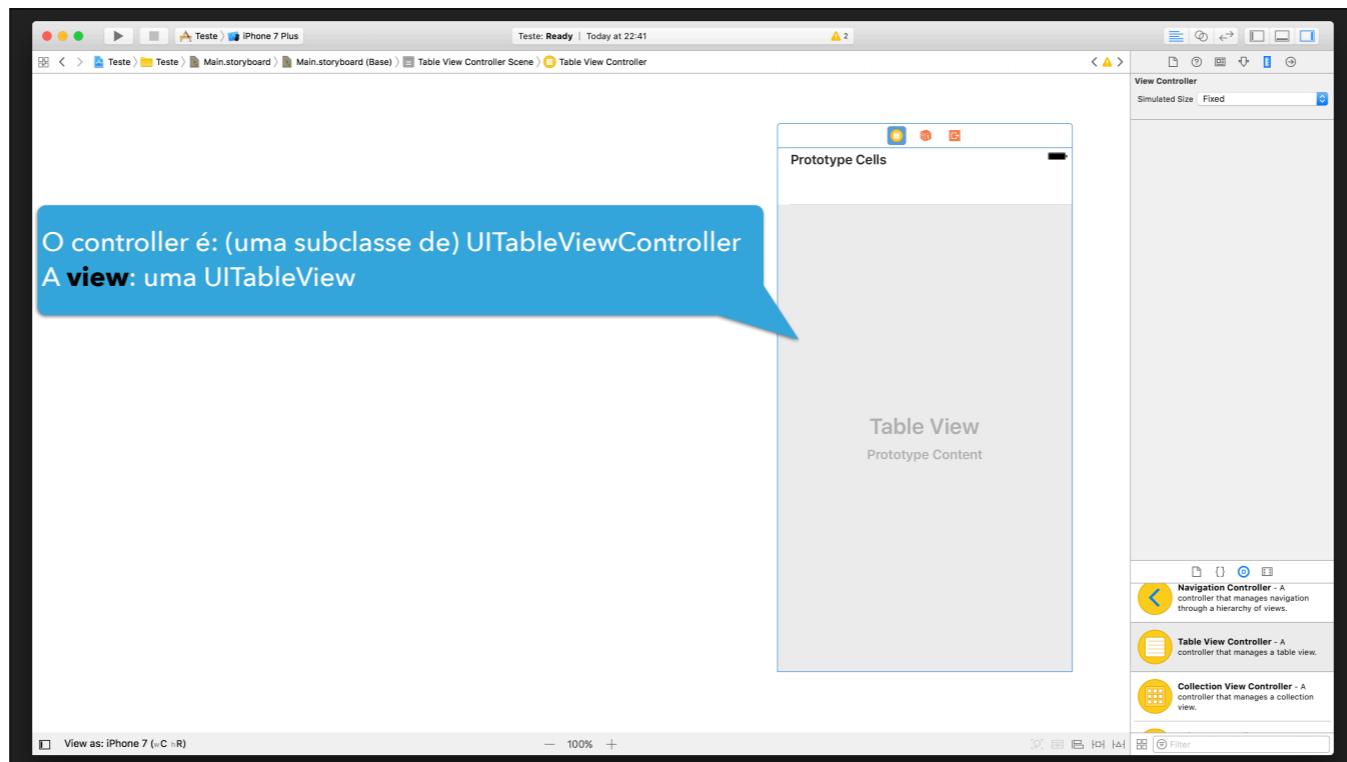


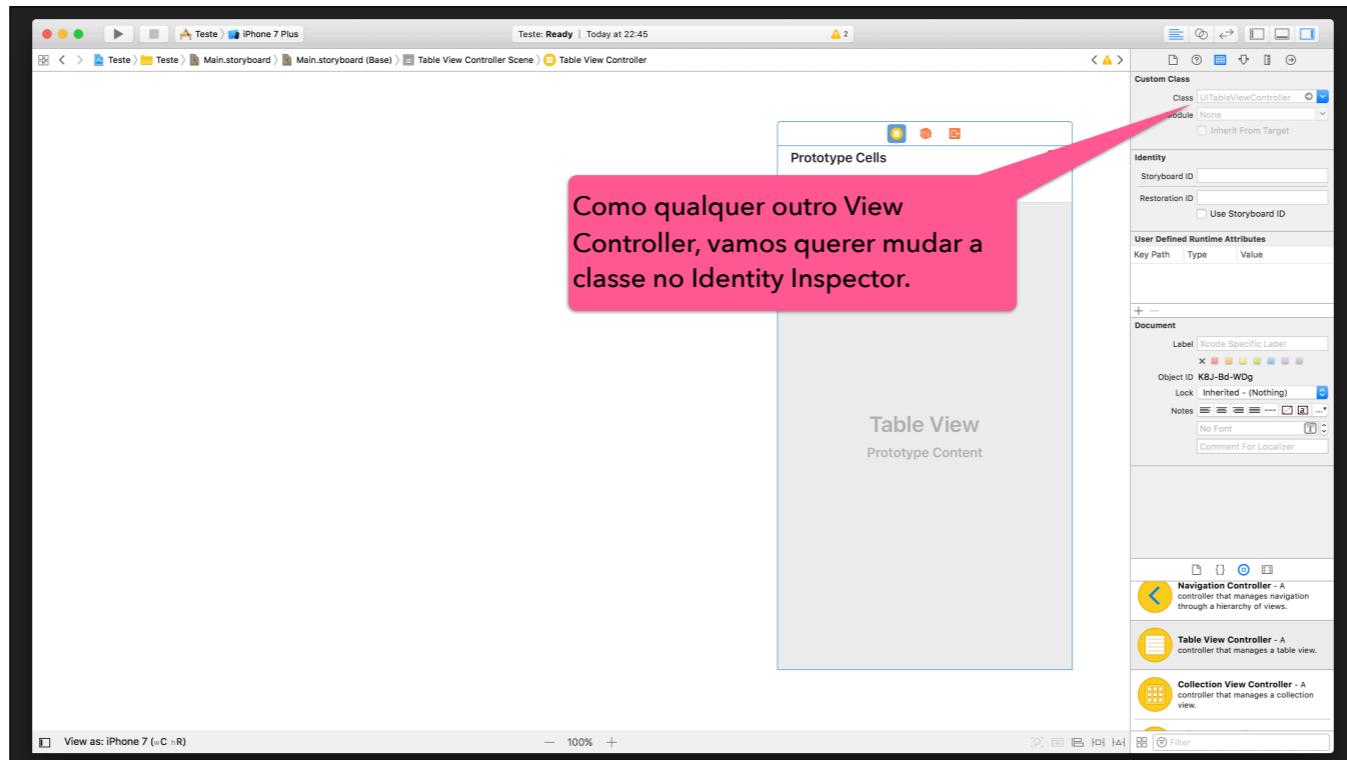
**ONDE EU JÁ VI
UMA TABLEVIEW
CONTROLLER
GROUPED?**

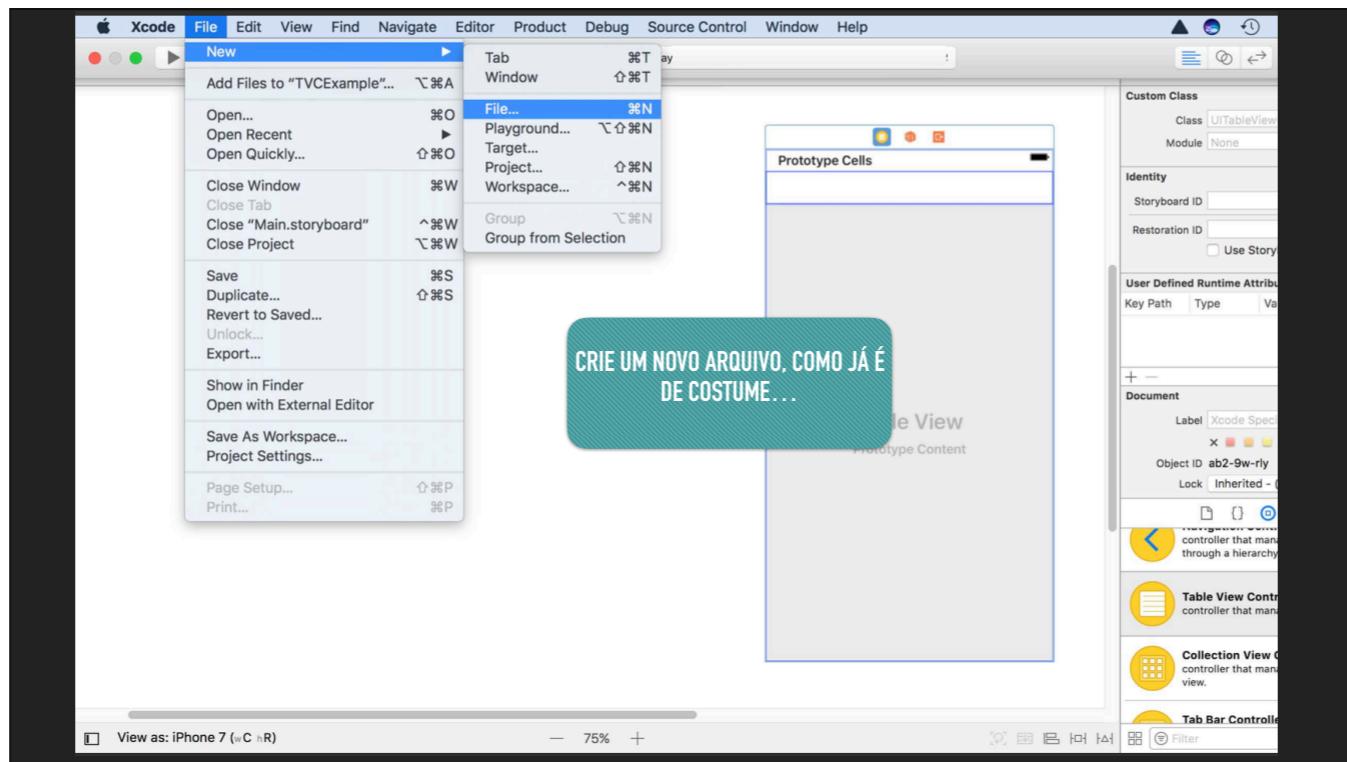
ONDE EU JÁ VI UMA TABLEVIEW CONTROLLER GROUPED?

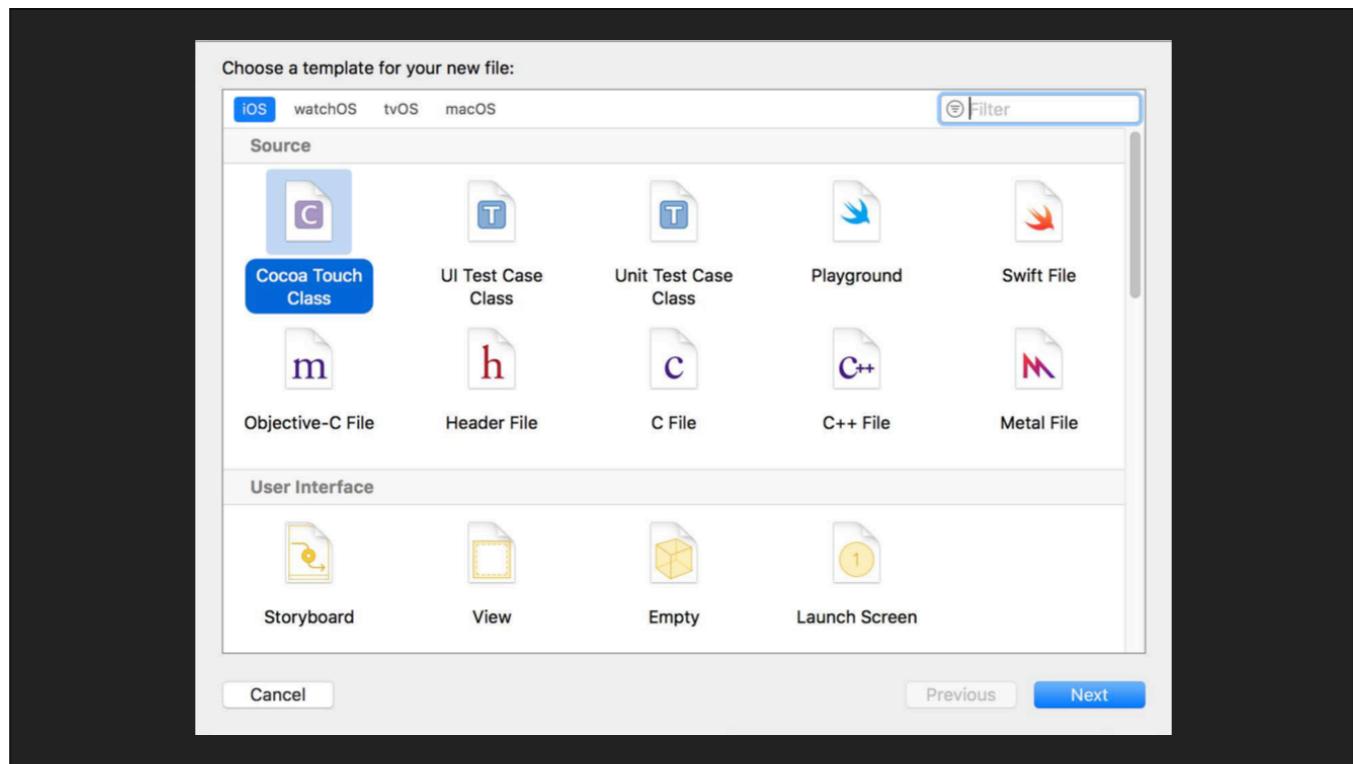


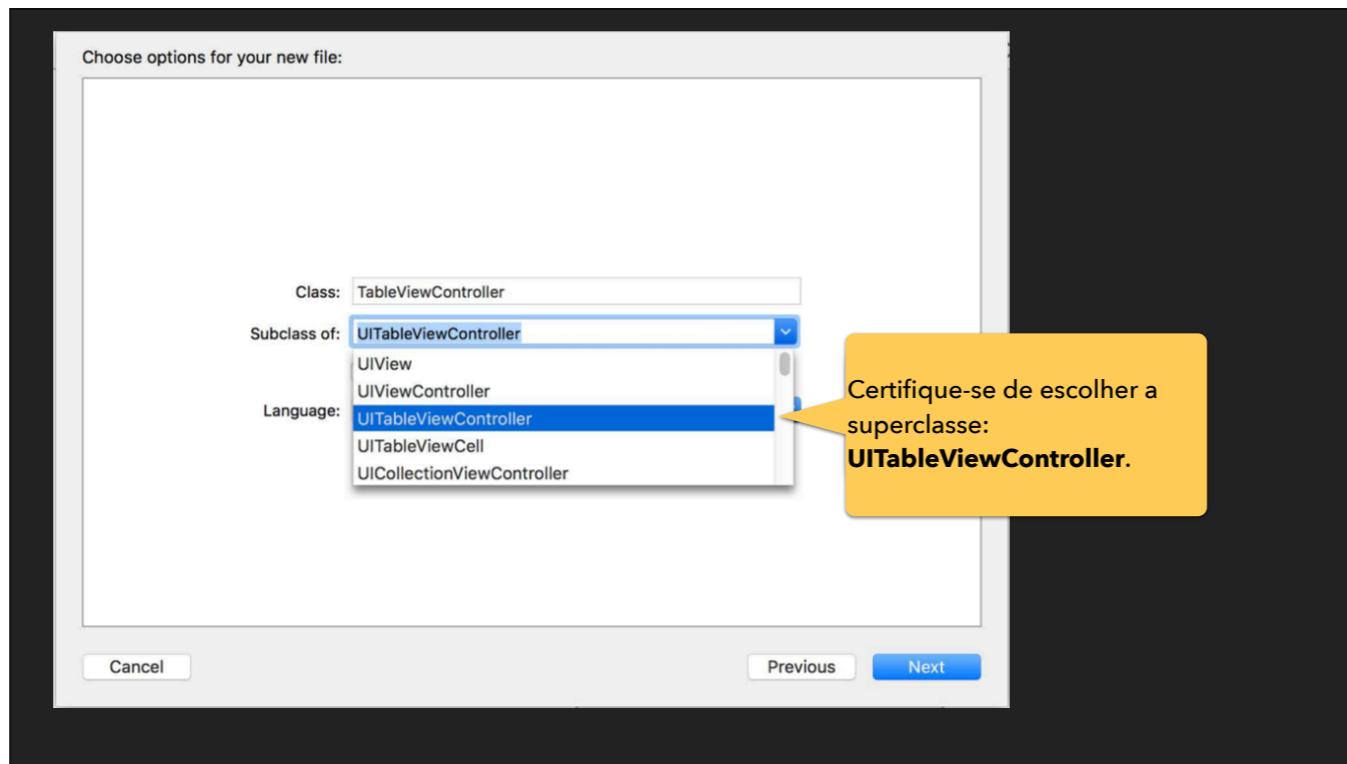


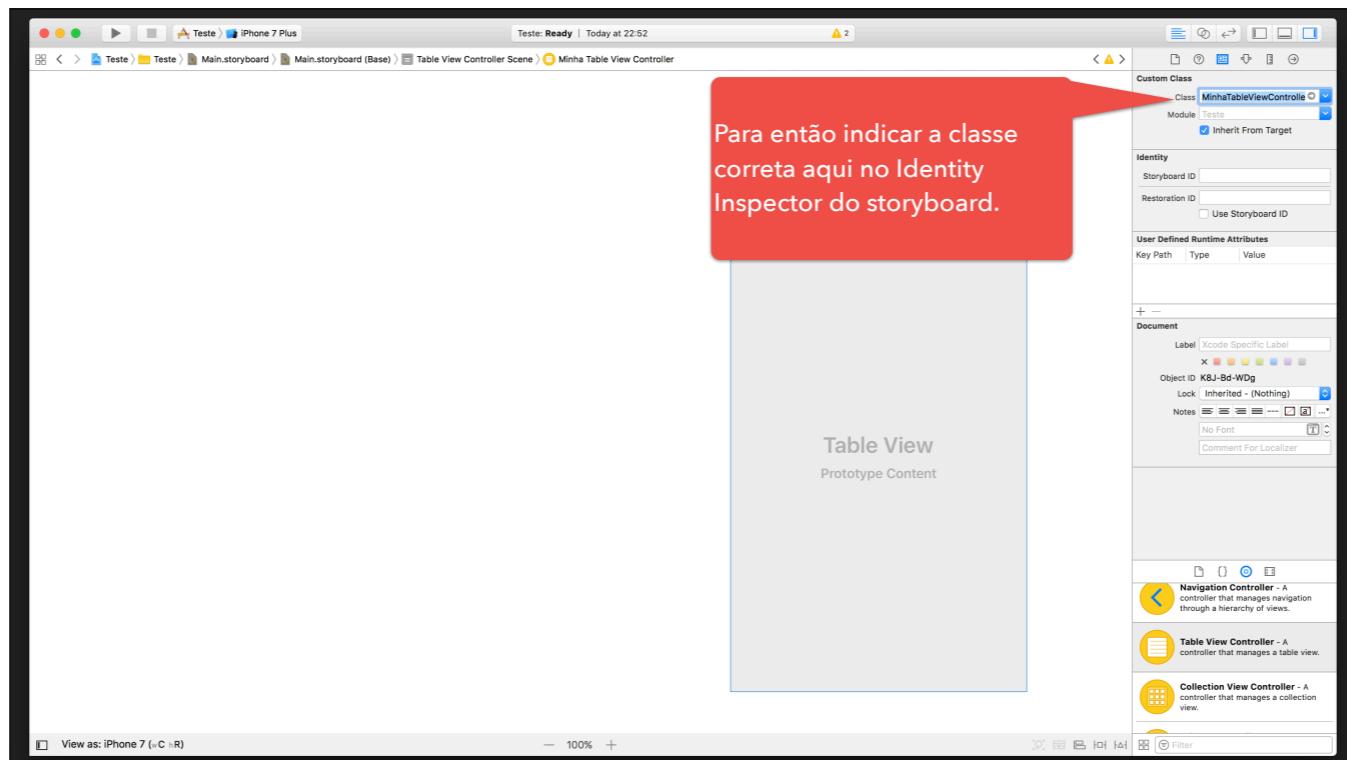


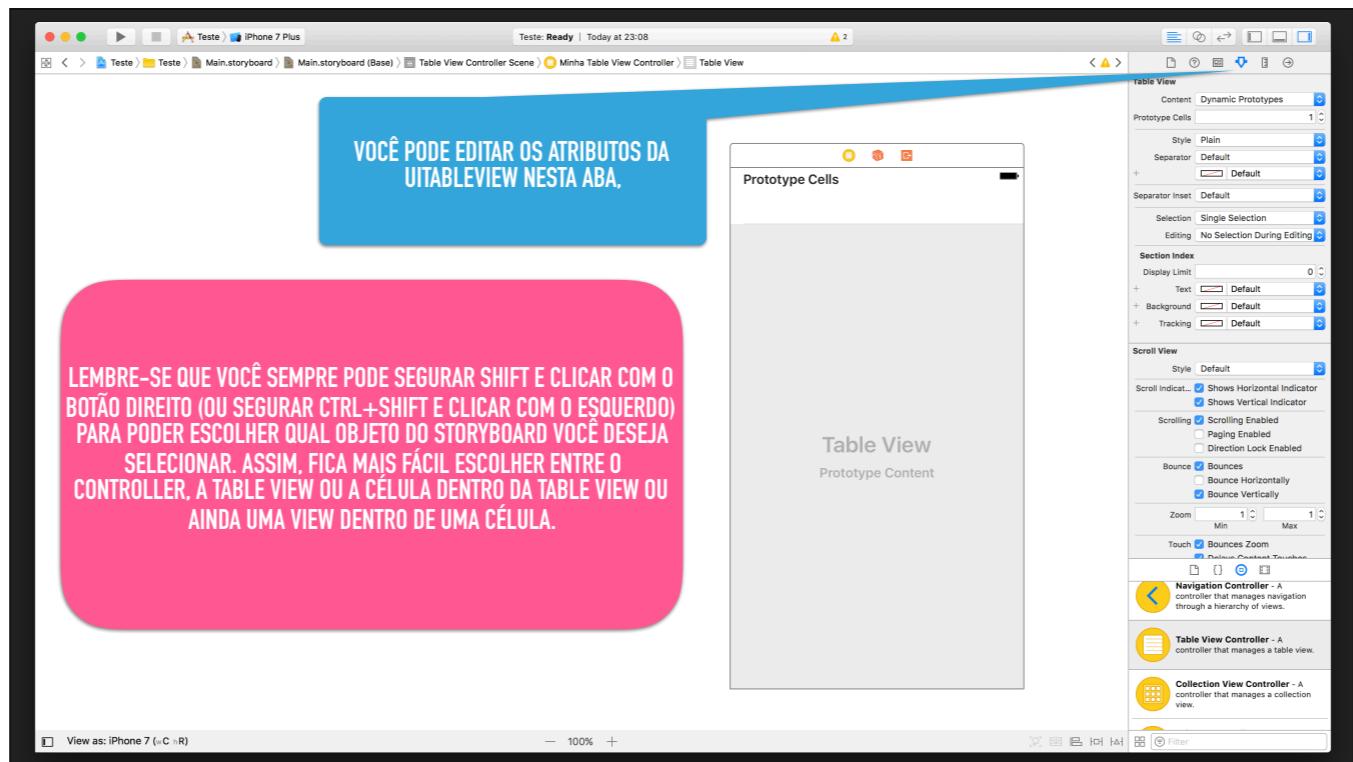


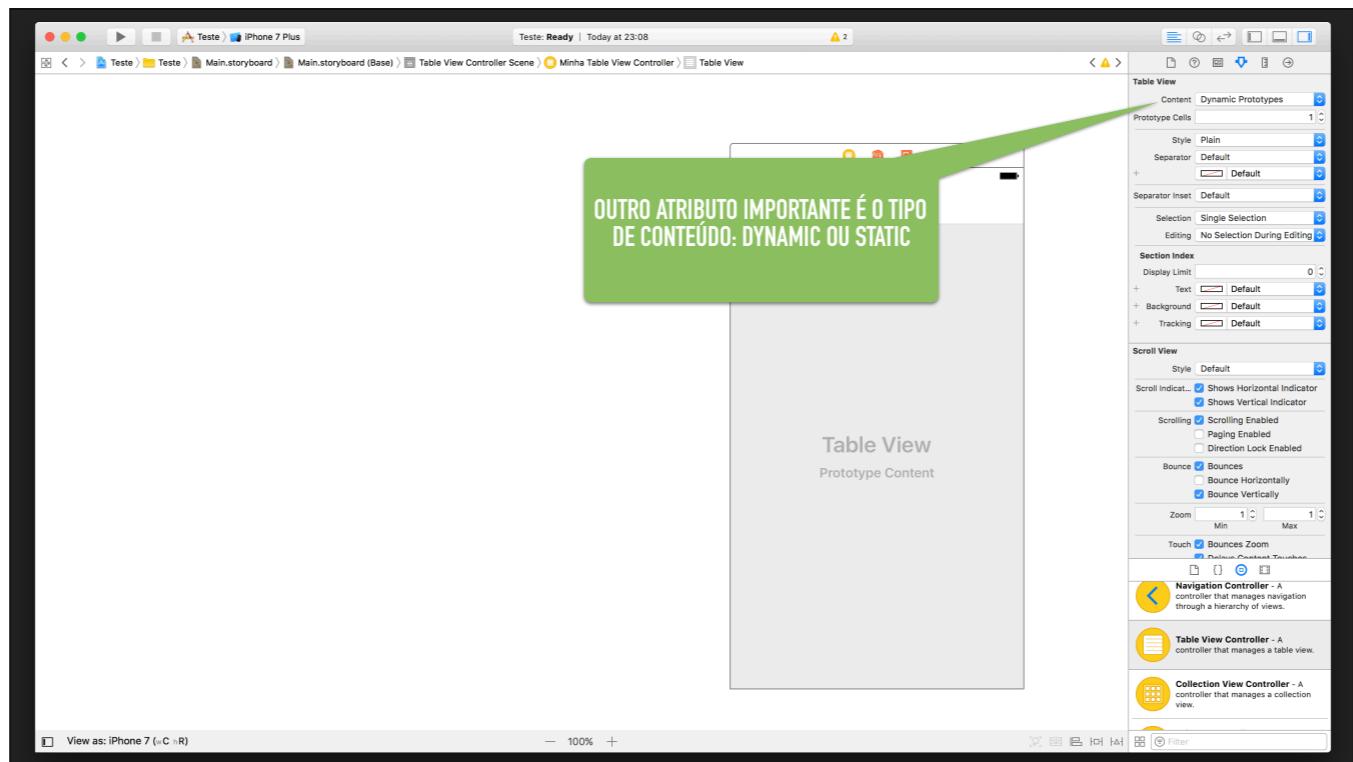












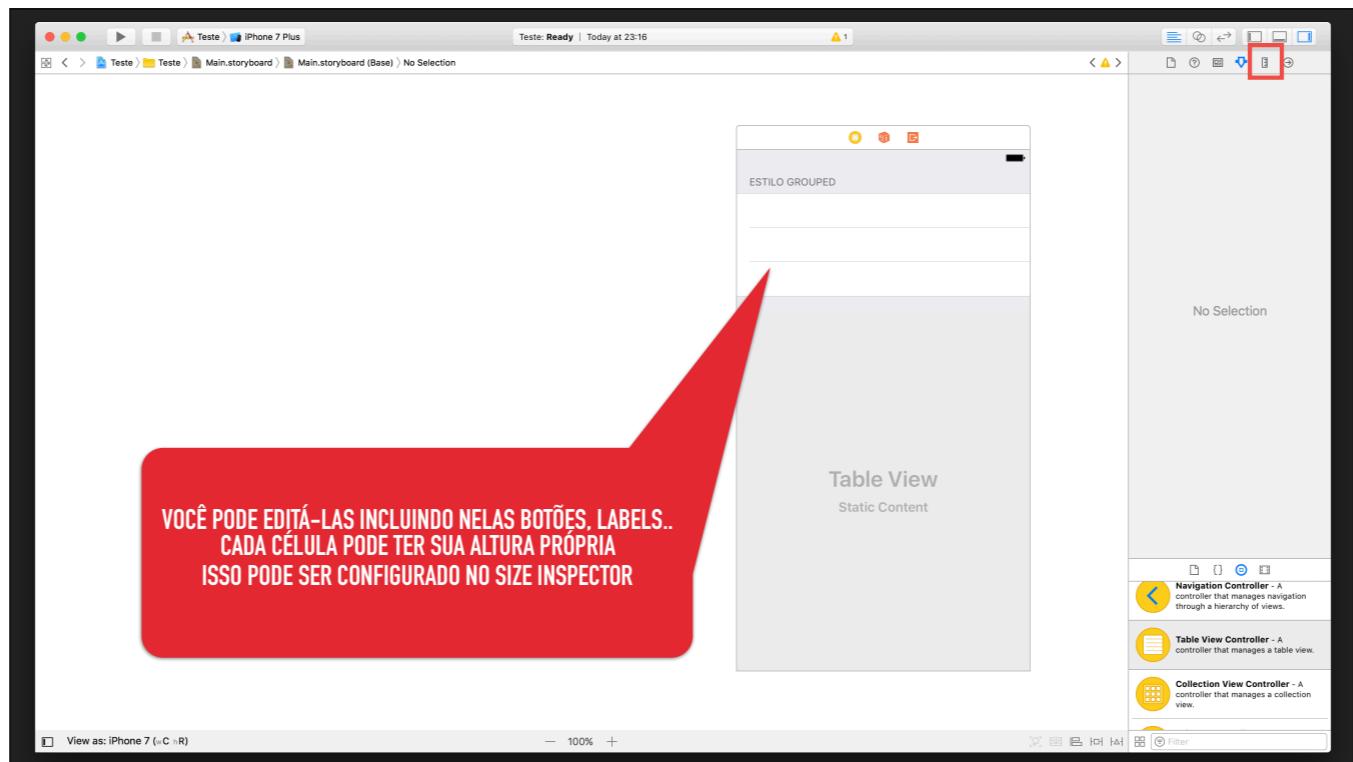
TABLEVIEW CONTROLLER

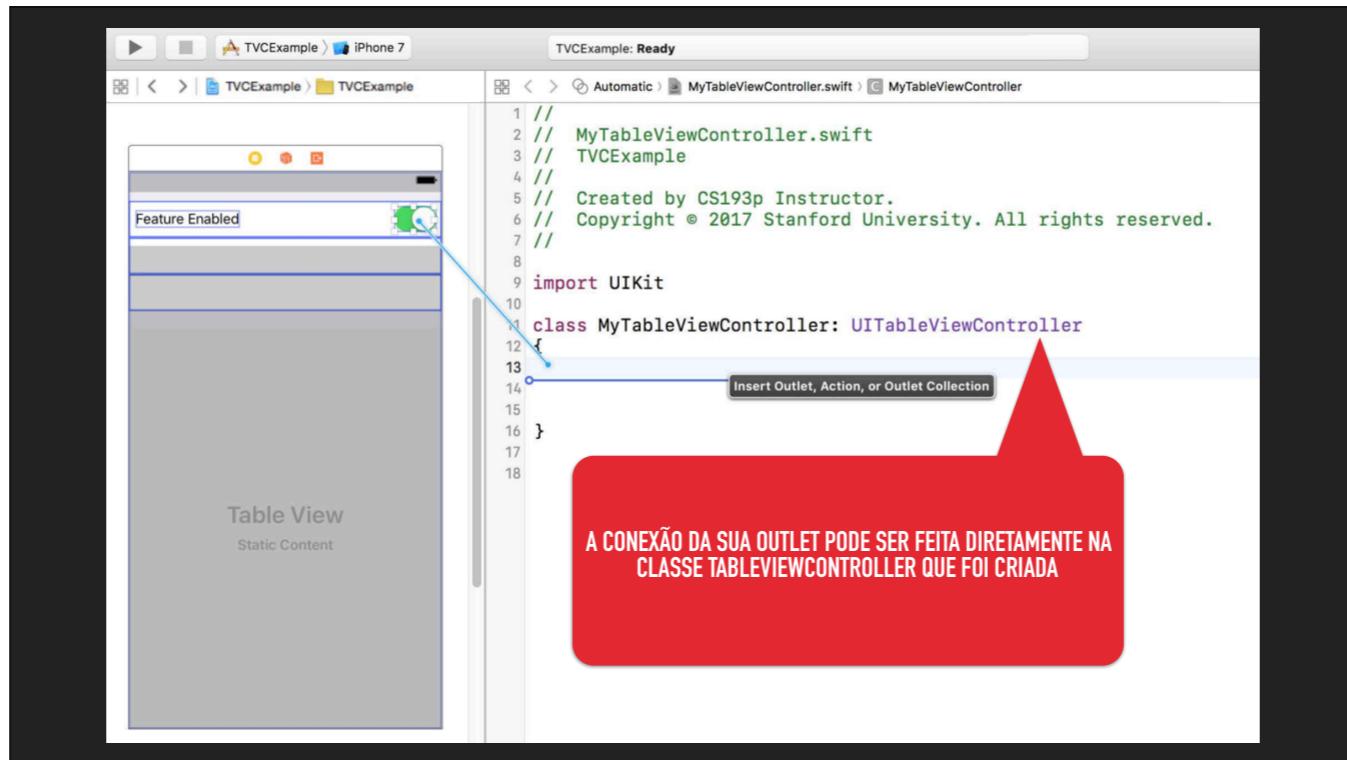
**STATIC CELLS
OU
DYNAMIC
PROTOTYPES?**

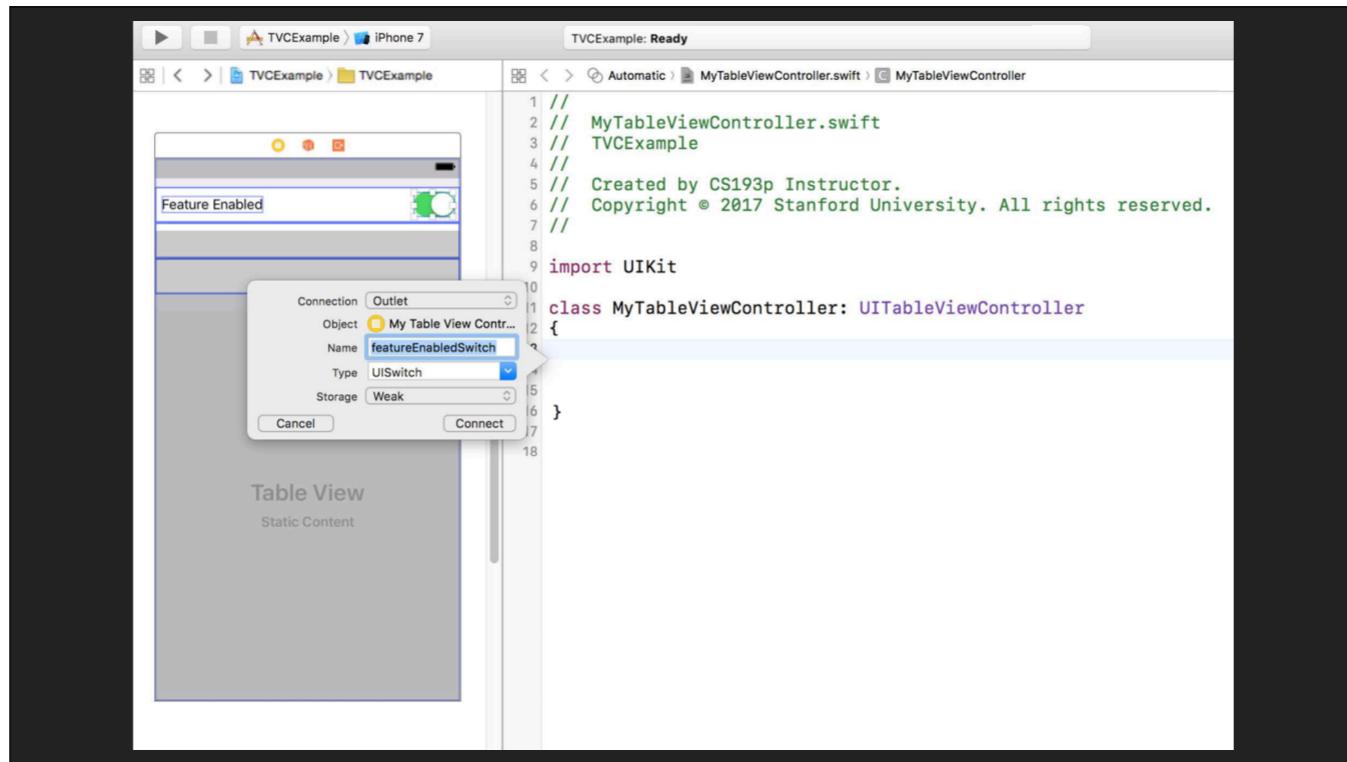
TABLEVIEW CONTROLLER

Static Cells:

- ▶ Servem para criar listas que você já sabe quantas células ela vai ter
- ▶ Muito usadas para configurações
- ▶ Podem fazer ligação diretamente do storyboard com o código
- ▶ Só existem para TableViewController, não para TableView



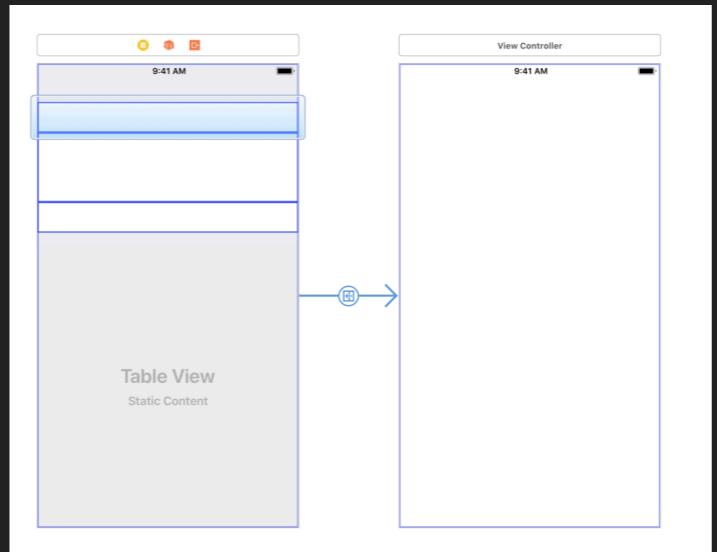




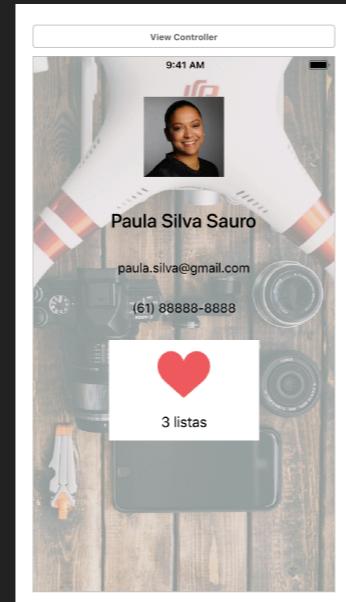
The screenshot shows the Xcode interface with two main panes. The left pane displays a storyboard for an iPhone 7 device. The storyboard contains a single view controller with a title bar labeled "Feature Enabled". Below the title bar is a green circular icon with a white letter "C". The main content area of the view controller is titled "Table View" and "Static Content". The right pane shows the code for the "MyTableViewController.swift" file. The code is as follows:

```
1 // MyTableViewController.swift
2 // TVCEExample
3 // Created by CS193p Instructor,
4 // Copyright © 2017 Stanford University. All rights reserved.
5 //
6 import UIKit
7
8 class MyTableViewController: UITableViewController
9 {
10
11    @IBOutlet weak var featureEnabledSwitch: UISwitch!
12
13
14
15
16 }
17
18
```

Outra coisa
interessante é
que conseguimos
criar segue a
partir da
TableViewCell



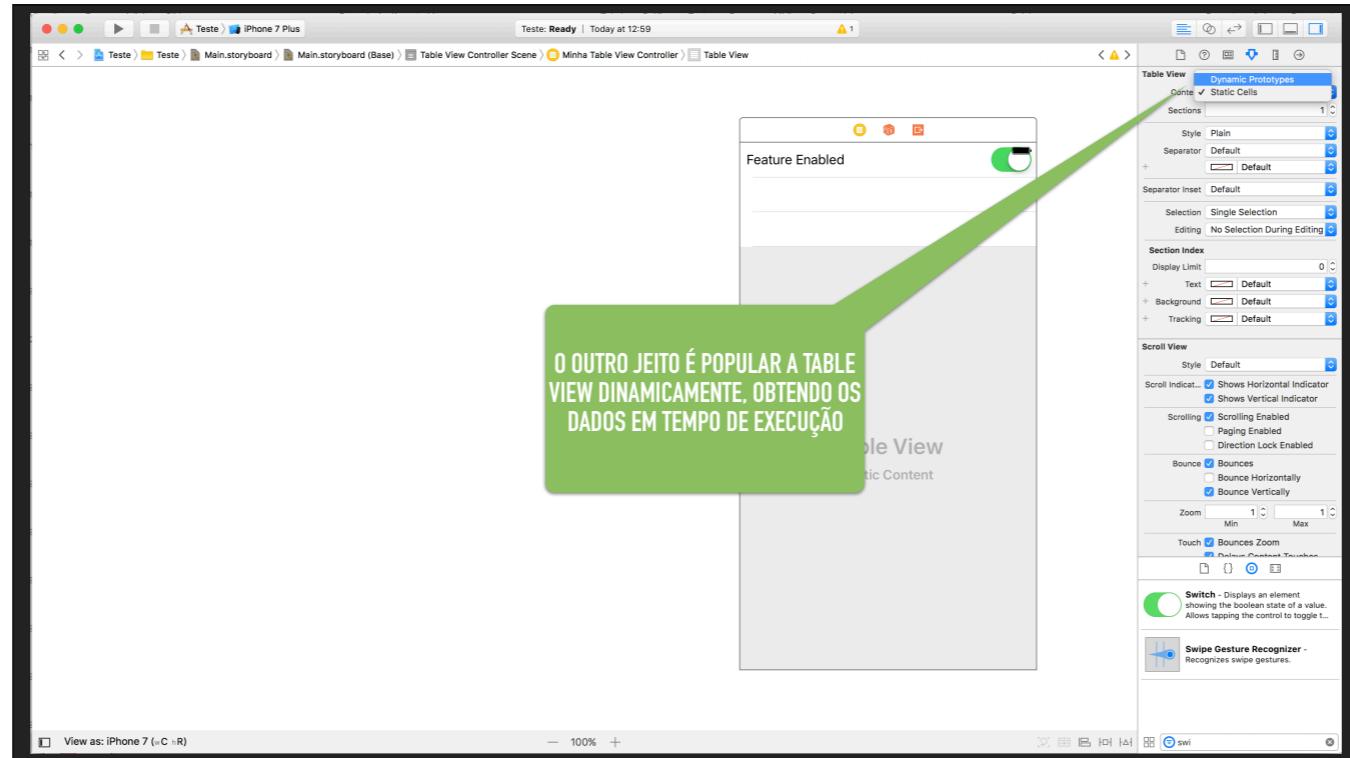
VAMOS AJUSTAR NOSSO PROJETO

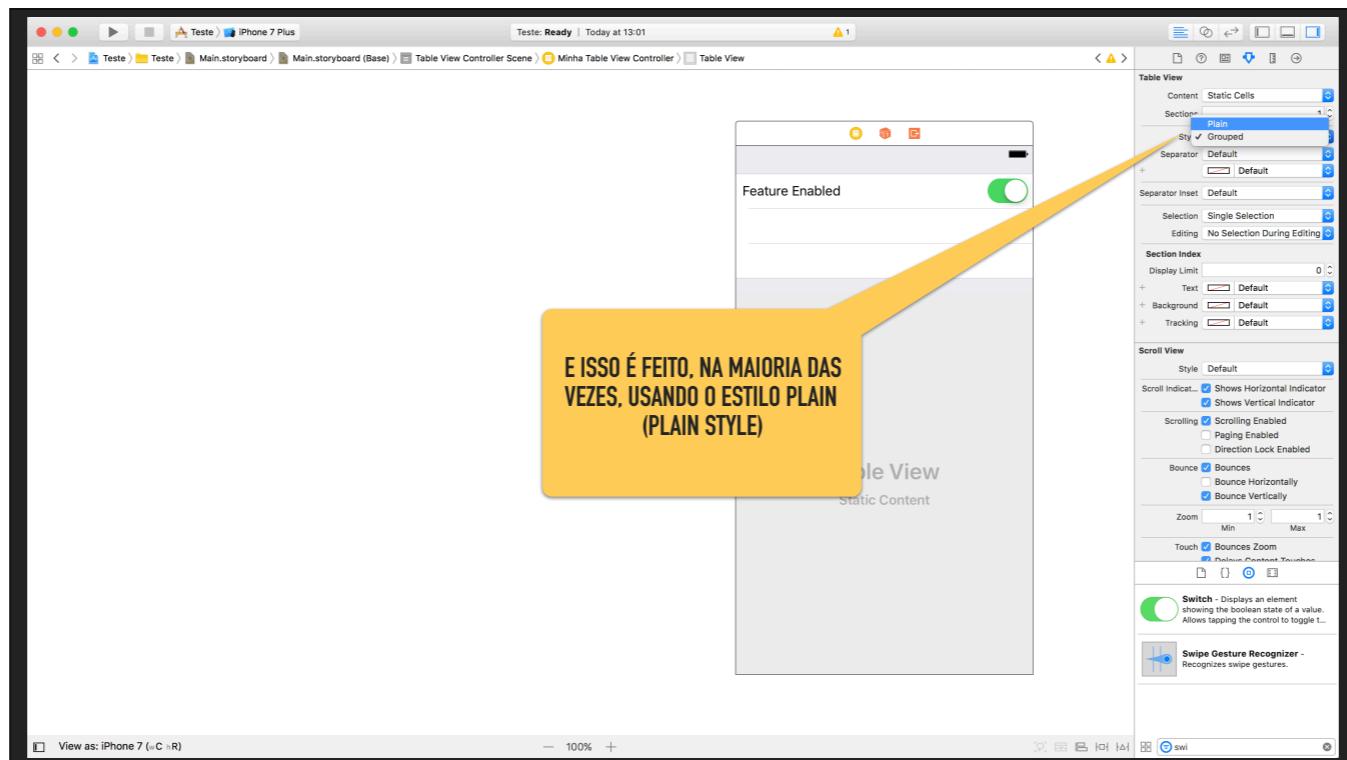
A registration form titled 'View Controller' with a close button 'X' at the top right. It features a circular profile placeholder with a blue silhouette. Below it are four input fields with red borders: 'Nome', 'Email', 'Senha', and 'Telefone'. At the bottom is a red button labeled 'CADASTRE-SE'.

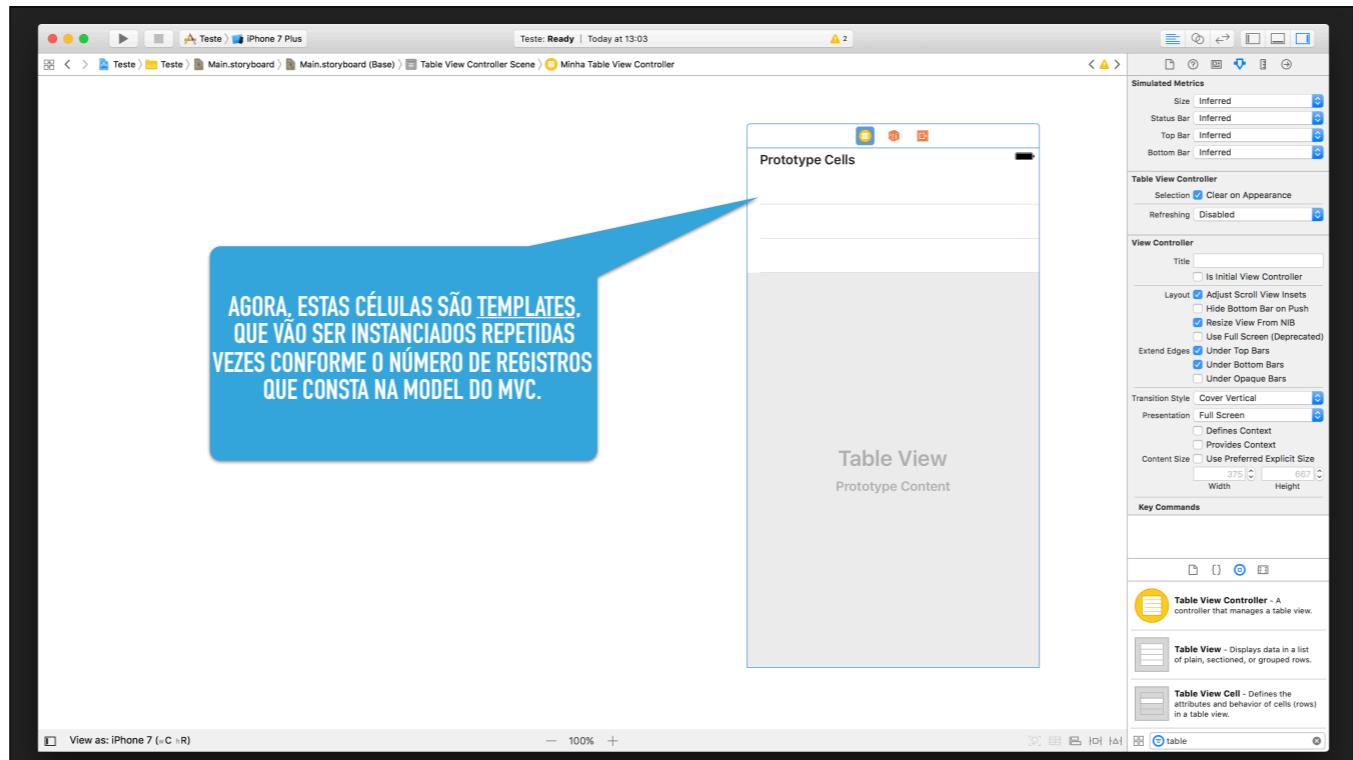
TABLEVIEW CONTROLLER

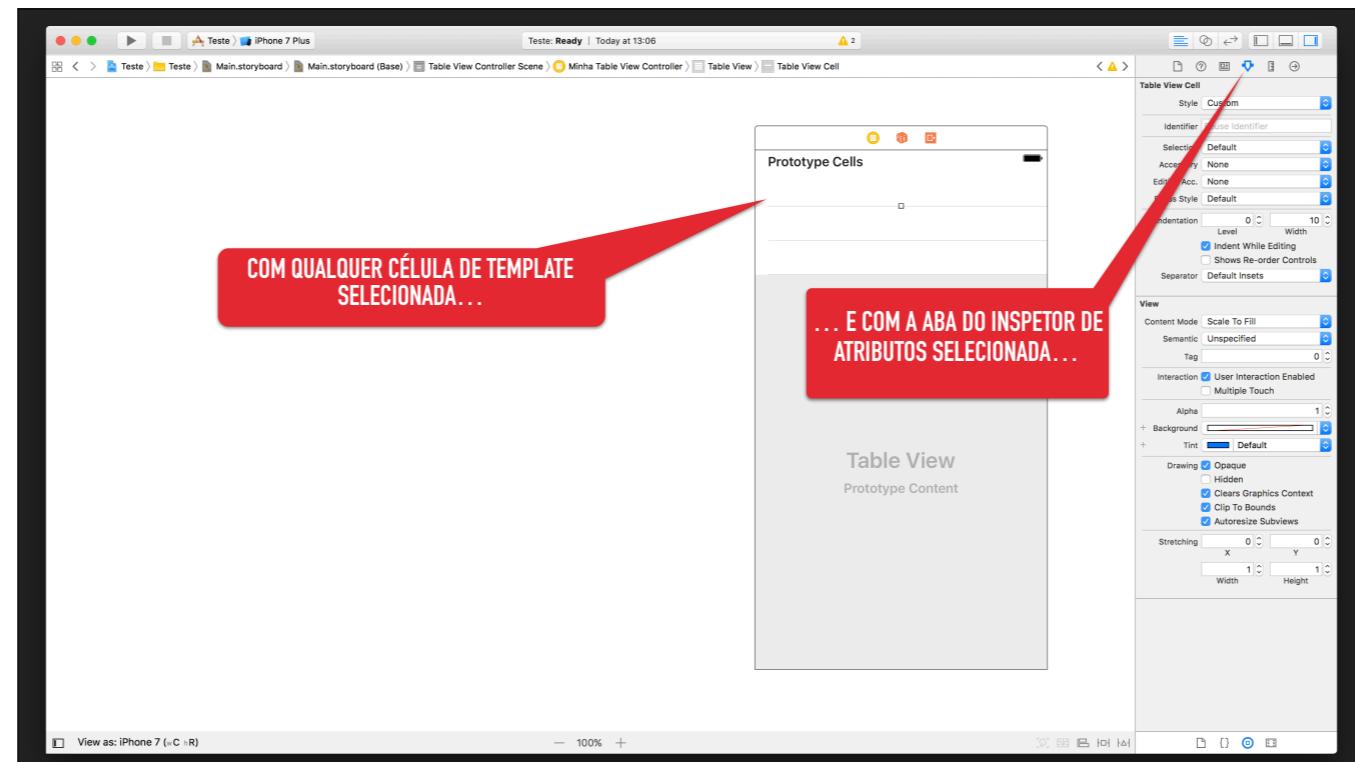
Dynamic Prototypes:

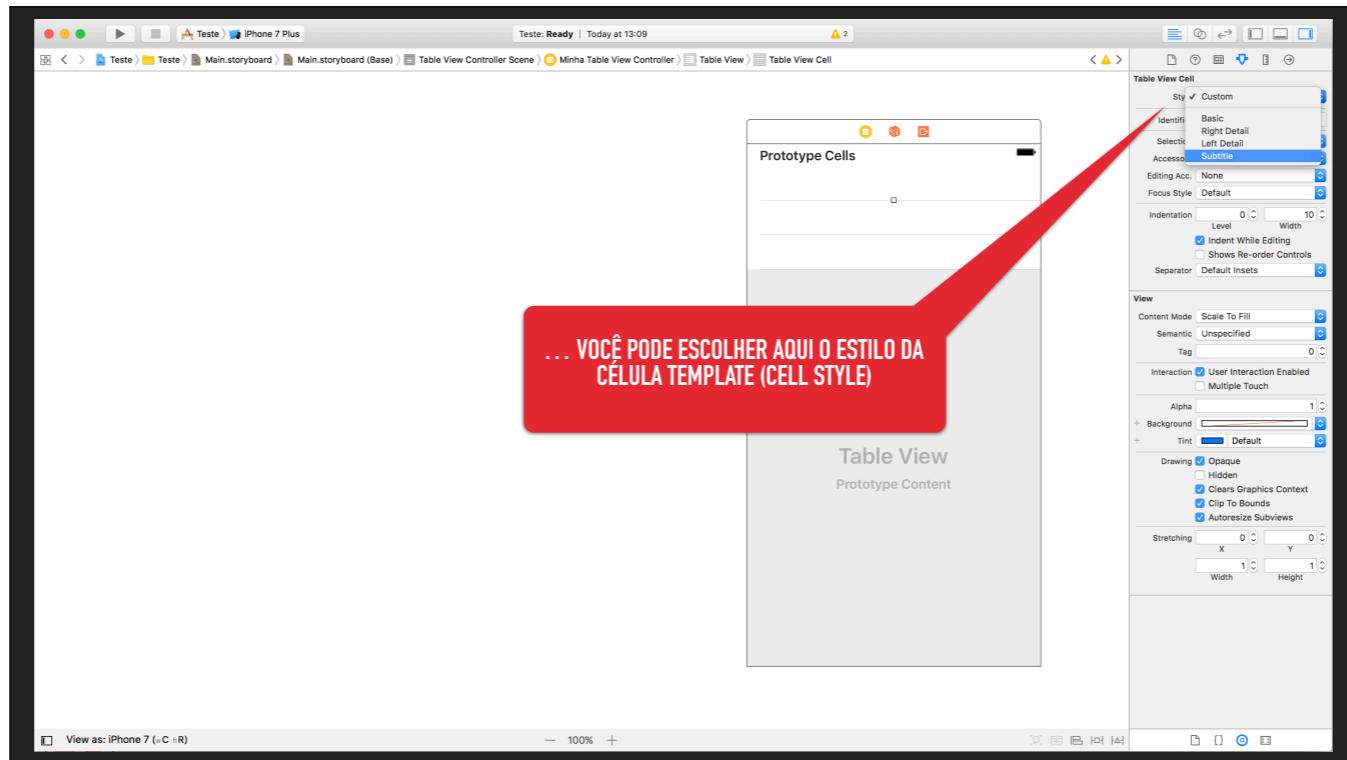
- ▶ Servem para criar listas que você não sabe quantas células ela vai ter
- ▶ Muito usadas para exibir informações vindas do servidor
- ▶ Não podem fazer ligação diretamente do storyboard com o código
- ▶ Existem tanto para a TableViewController, quanto para a TableView
- ▶ Cada célula funciona como um template a ser preenchido. Vão ter a mesma forma, porém informações diferentes
- ▶ É possível ter mais de um template de célula





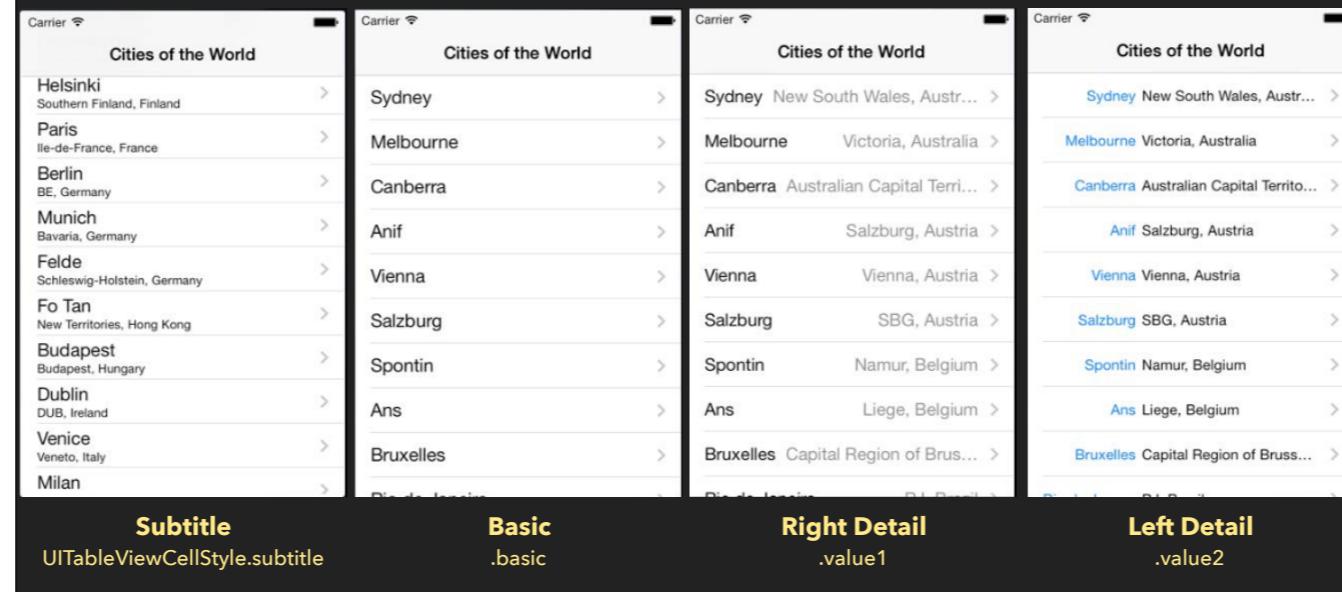


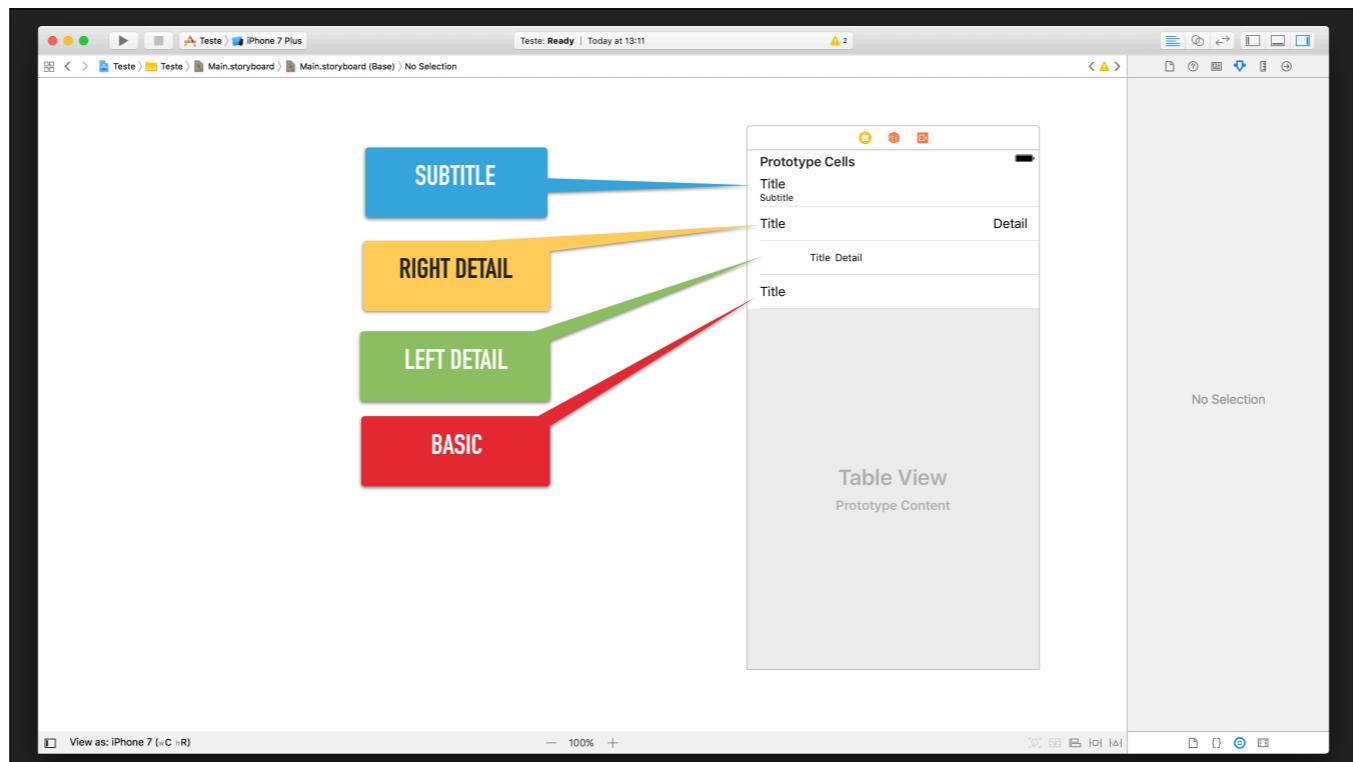


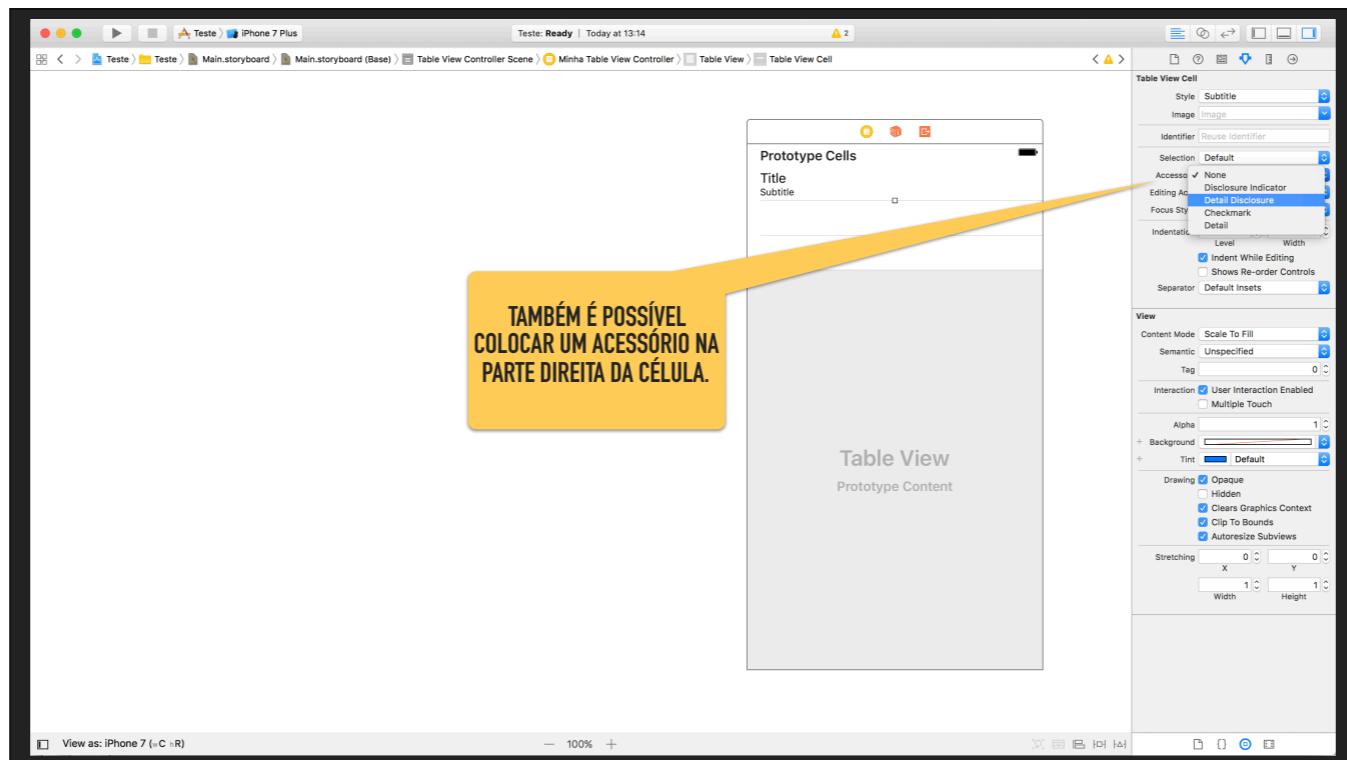


CADA UM NO SEU QUADRADO

TIPOS DE CÉLULA PRÉ-DEFINIDOS (CELL STYLE)







TABLEVIEW CONTROLLER

SE EU NÃO POSSO CONECTAR MEUS ELEMENTOS DA CÉLULA COM O CÓDIGO VIA OUTLETS, COMO EU FAÇO?

Nossa UITableViewcontroller é uma classe muito especial

Ela é como a UIViewController, só que com mais funcionalidades

Você notou que quando ela foi criada, ela já veio com código comentado?

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

override func numberOfSections(in tableView: UITableView) -> Int {
    // Warning: Incomplete implementation, return the number of sections
    return 0
}

override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // Warning: Incomplete implementation, return the number of rows
    return 0
}

/*
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "reusableIdentifier", for: indexPath)
    // Configure the cell...
    return cell
}
*/

// Override to support conditional editing of the table view.
override func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool {
    // Return false if you do not want the specified item to be editable.
    return true
}

/*
// Override to support editing the table view.
override func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCellEditingStyle, forRowAt indexPath: IndexPath) {
    if editingStyle == .delete {
        // Delete the row from the data source
        tableView.deleteRows(at: [indexPath], with: .fade)
    } else if editingStyle == .insert {
        // Create a new instance of the appropriate class, insert it into
        // the array, and add a new row to the table view
    }
}
*/
```

TABLEVIEW CONTROLLER

ELES SÃO DELEGATES!

A UITableView Controller delega para nós implementarmos esses métodos.

Se não implementarmos, ela exibe uma TableView vazia

Vamos dar uma olhada em um método, o `cellForRowAt:`

```
tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
```

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

override func numberOfSections(in tableView: UITableView) -> Int {
    // Warning: Incomplete implementation, return the number of sections
    return 0
}

override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // Warning: Incomplete implementation, return the number of rows
    return 0
}

/*
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "reusableIdentifier", for: indexPath)
    // Configure the cell...
    return cell
}
*/

// Override to support conditional editing of the table view.
override func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool {
    // Return false if you do not want the specified item to be editable.
    return true
}

/*
// Override to support editing the table view.
override func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCellEditingStyle, forRowAt indexPath: IndexPath) {
    if editingStyle == .delete {
        // Delete the row from the data source
        tableView.deleteRows(at: [indexPath], with: .fade)
    } else if editingStyle == .insert {
        // Create a new instance of the appropriate class, insert it into
        // the array, and add a new row to the table view
    }
}
*/
```

TABLEVIEW CONTROLLER

PROVENDO UMA UIVIEW PARA DESENHAR CADA CÉLULA

- ▶ A UIView que você vai fornecer para cada célula precisa ser subclasse de UITableViewCell (que, por sua vez, é subclasse de UIView);
- ▶ Somente as células que estiverem visíveis irão possuir uma instância dessa classe;
- ▶ Isso significa, por outro lado, que as instâncias de UITableViewCell são **reutilizadas** à medida que células aparecem e desaparecem;
- ▶ Como é de se imaginar, isso tem complicadores para cenários multithread, então seja cuidadoso!

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
    let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)  
    // Configure the cell...  
    return cell  
}
```

TABLEVIEW CONTROLLER

PROVENDO UMA UIVIEW PARA DESENHAR CADA CÉLULA

- ▶ O indexPath é uma estrutura de dados que recebemos com a informação da tela que vamos desenhar;
- ▶ As propriedades mais importantes são:
 - ▶ indexPath.section: indica qual section está sendo criada
 - ▶ indexPath.row: indica qual a linha da section está sendo crianda

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
    let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)  
    // Configure the cell...  
    return cell  
}
```

TABLEVIEW CONTROLLER

PROVENDO UMA UIVIEW PARA DESENHAR CADA CÉLULA

- ▶ Essa cell, já contém as properties que queremos acessar:
 - ▶ `textLabel` - é o título
 - ▶ `detailTextLabel` - é o detail/subtitle
- ▶ Vamos preencher:

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
    let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)  
    // Configure the cell...  
    cell.textLabel?.text = "Titulo"  
    cell.detailTextLabel?.text = "Detalhe"  
    return cell  
}
```

TABLEVIEW CONTROLLER

PROVENDO UMA UIVIEW PARA DESENHAR CADA CÉLULA

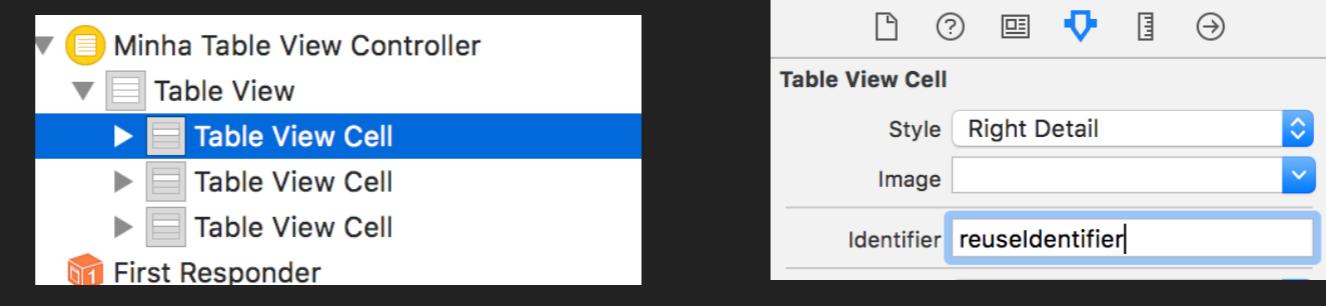
- ▶ Essa cell, pede um identifier
- ▶ Ele é definido nas propriedades das célula no storyboard

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
    let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)  
    // Configure the cell...  
    cell.textLabel?.text = "Titulo"  
    cell.detailTextLabel?.text = "Detalhe"  
    return cell  
}
```

TABLEVIEW CONTROLLER

PROVENDO UMA UIVIEW PARA DESENHAR CADA CÉLULA

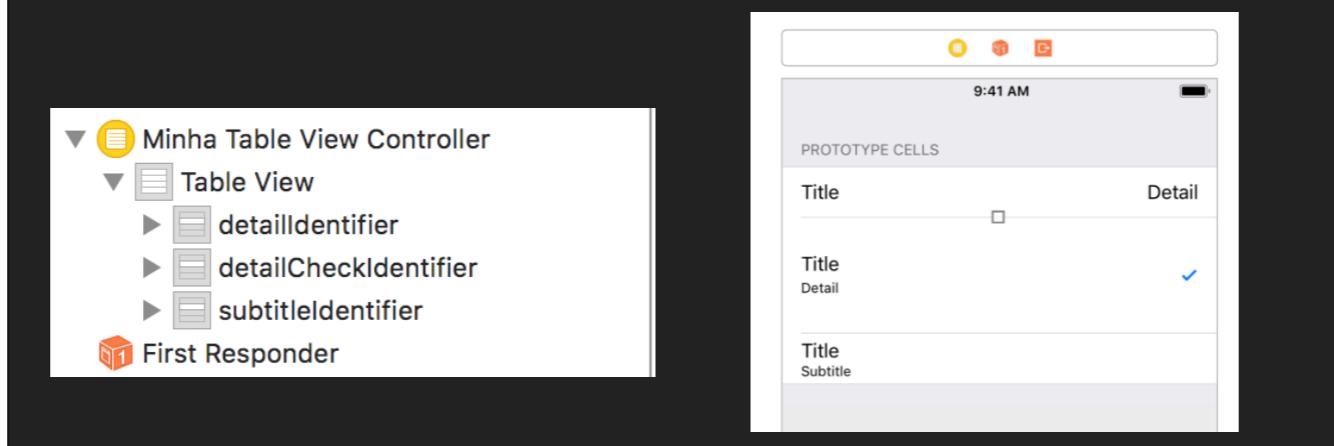
- Essa cell, pede um identifier
- Ele é definido nas propriedades das célula no storyboard
- Selecione a célula no Storyboard
- Preencha o Identifier da célula



TABLEVIEW CONTROLLER

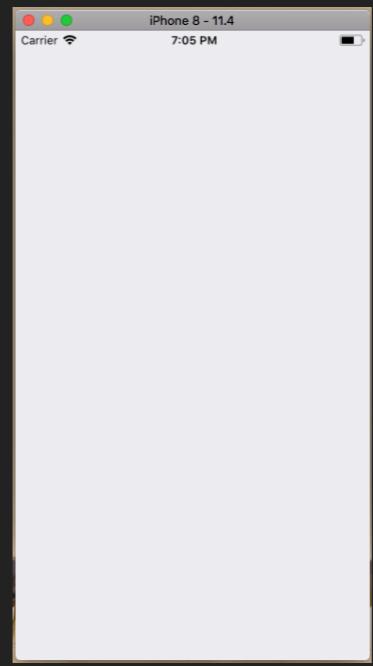
PROVENDO UMA UIVIEW PARA DESENHAR CADA CÉLULA

- ▶ Como são templates, não faz sentido terem vários templetas iguais
- ▶ Muito menos utilizarem o mesmo identificador



TABLEVIEW CONTROLLER

EXECUTEM



TABLEVIEW CONTROLLER

APRENDEMOS A ENTREGAR UMA UIVIEW PARA DESENHAR CADA CÉLULA

- ▶ Quantas células devem ser desenhadas?
- ▶ Quantas sections tem?

```
override func numberOfSections(in tableView: UITableView) -> Int {  
    // #warning Incomplete implementation, return the number of sections  
    return 0  
}  
  
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
    // #warning Incomplete implementation, return the number of rows  
    return 0  
}
```

TABLEVIEW CONTROLLER

- ▶ Estamos sobreescrevendo o método `numberOfRowsInSection`, se removermos, ele considera como return 1
- ▶ Estamos sobreescrevendo o método `cellForRowAt`, se removermos, ele considera como return 0

```
override func numberOfSections(in tableView: UITableView) -> Int {  
    // #warning Incomplete implementation, return the number of sections  
    return 0  
}  
  
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
    // #warning Incomplete implementation, return the number of rows  
    return 0  
}
```

TABLEVIEW CONTROLLER

EXERCÍCIO



► Dica: indexPath

