

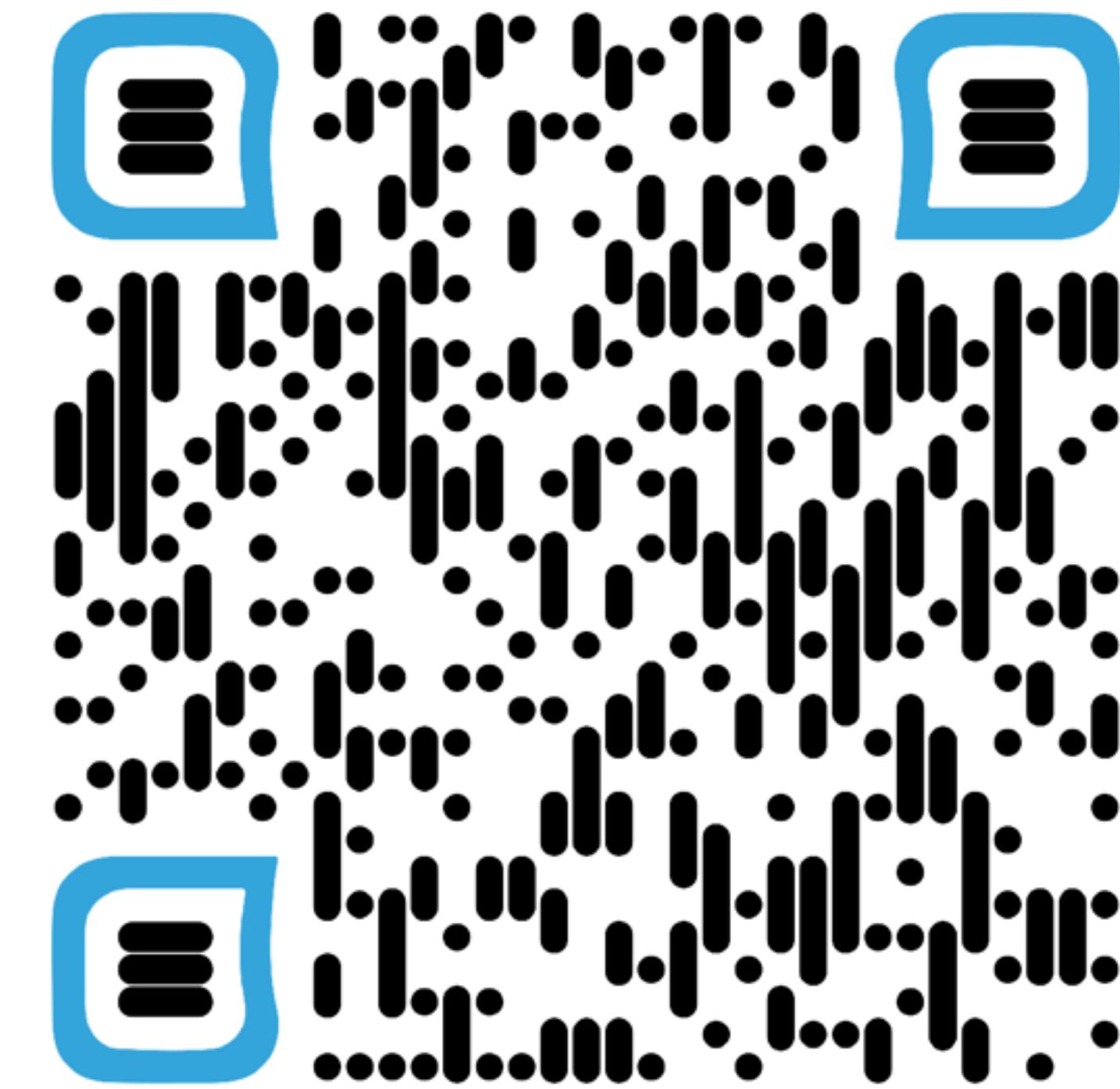


PROF. PEDRO HENRIQUE

---

# DESENVOLVIMENTO PARA IOS 11 COM SWIFT 4

GITHUB DA TURMA



[HTTPS://GIT.IO/VF50W](https://git.io/vf50w)

ONDE ENCONTRAR O MATERIAL?

# LEIA O QR CODE

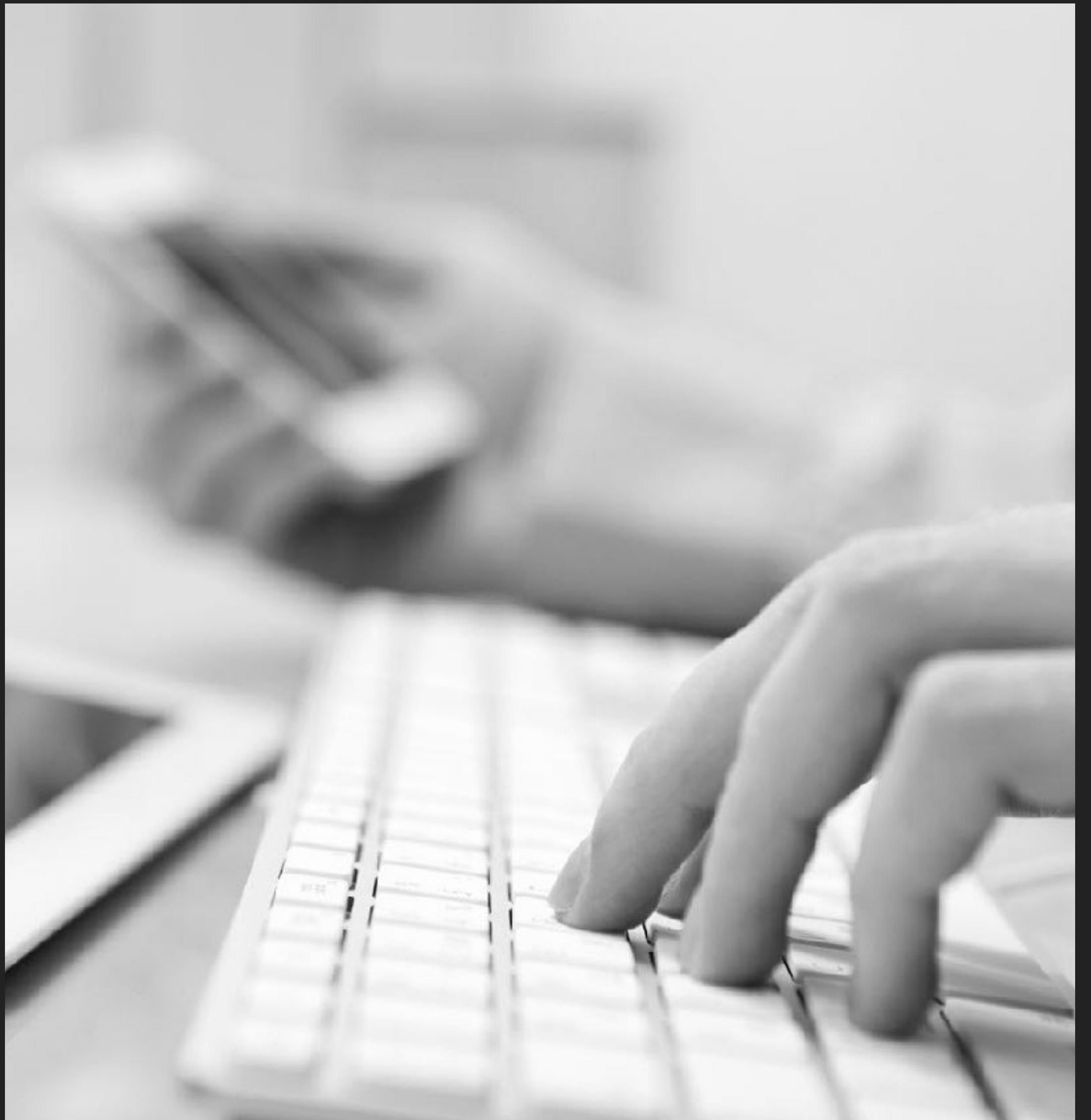
ALÉM DO TRADICIONAL BLACKBOARD DO IESB

## O QUE VAMOS FAZER HOJE?

---

### AGENDA

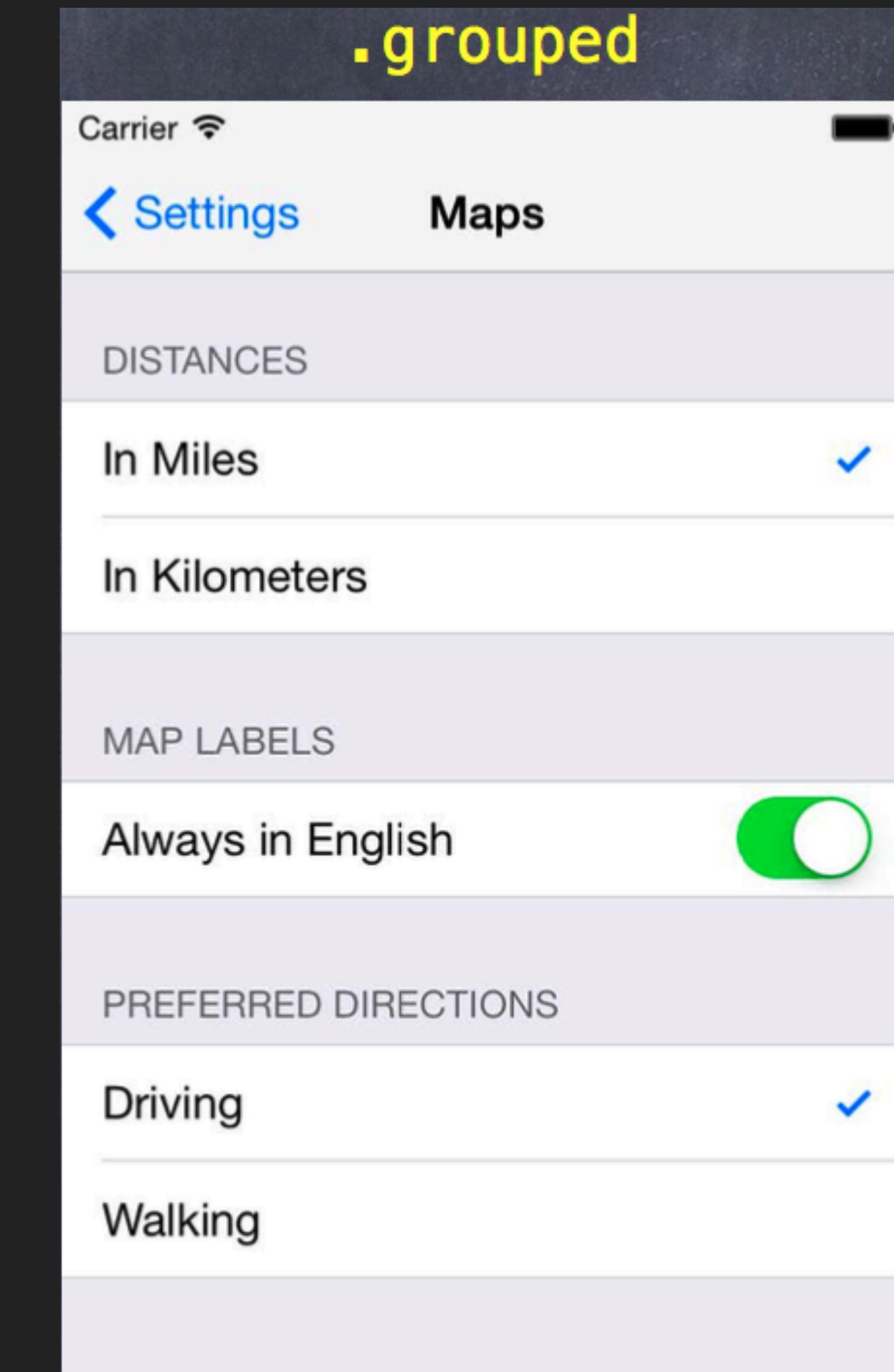
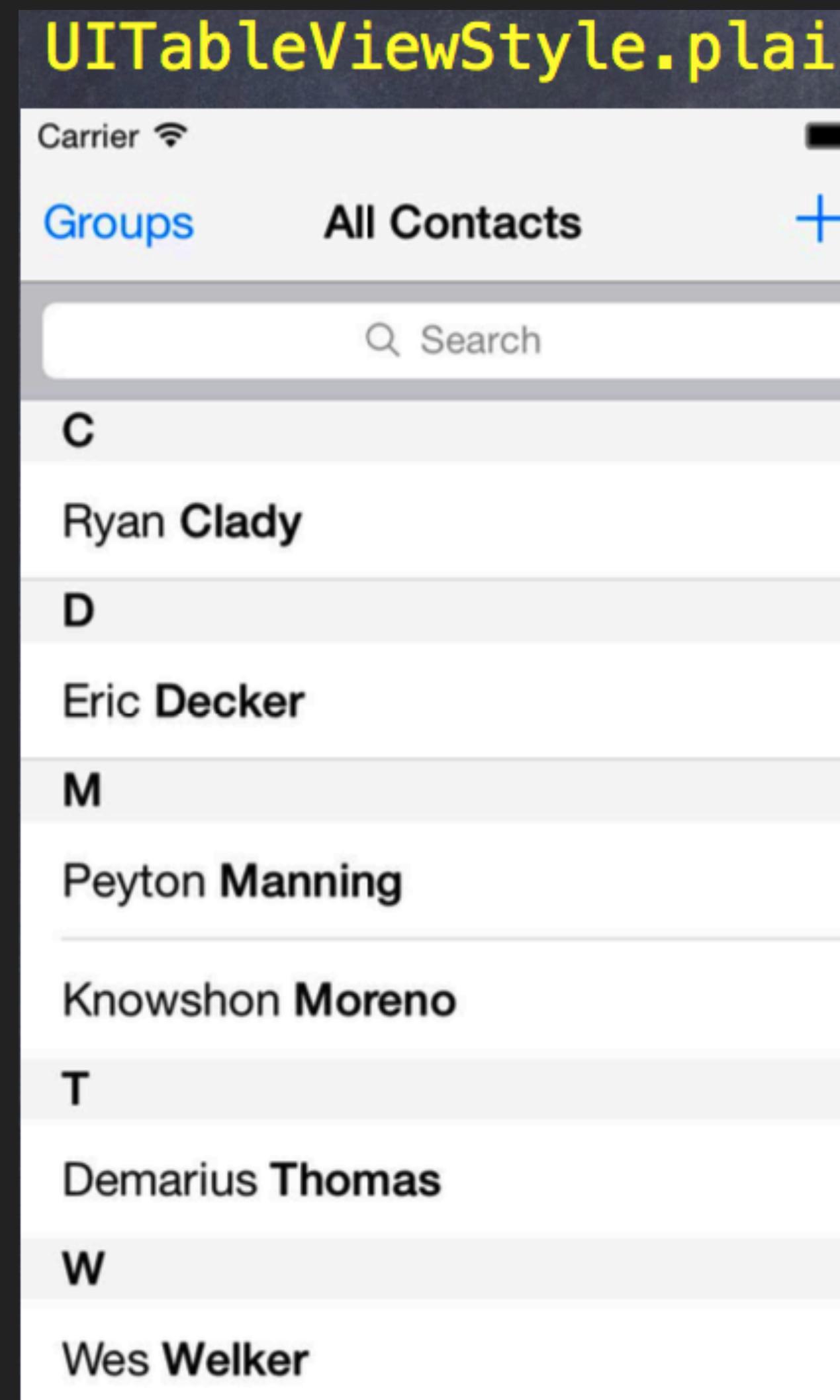
- ▶ UITableView e UITableViewController
- ▶ Células customizadas
- ▶ Prática



UMA LISTA PARA TUDO!

## UITABLEVIEW

PARA UMA LISTA DINÂMICA, COM REGISTROS QUE NÃO SÃO DETERMINADOS EM TEMPO DE PROGRAMAÇÃO



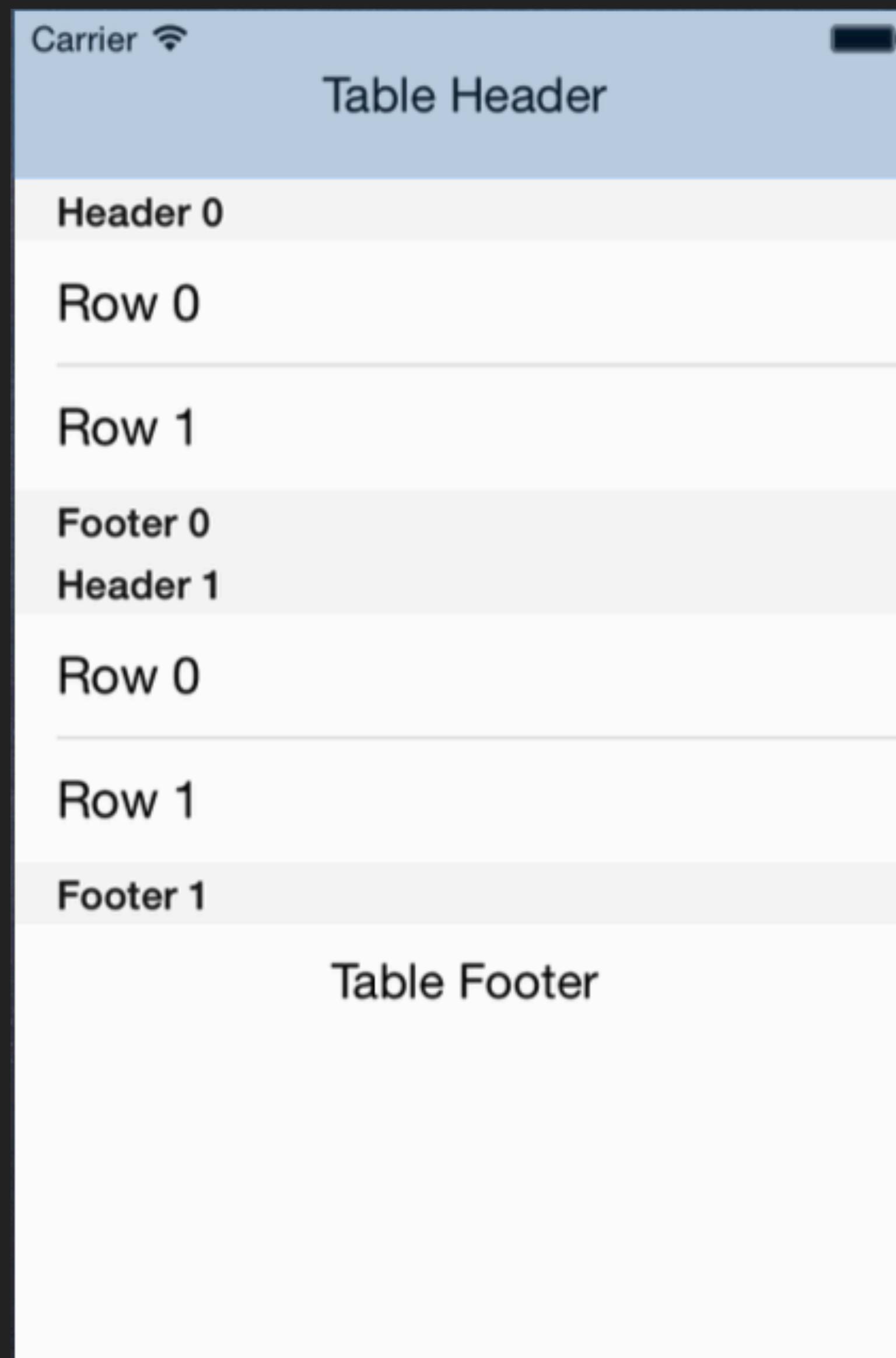
PARA UMA LISTA ESTÁTICA, COM ÍTEIS DETERMINADOS EM TEMPO DE PROGRAMAÇÃO

# UITABLEVIEW

---

## PLAIN STYLE

CABEÇALHO DA TABELA  
(TABLE HEADER)



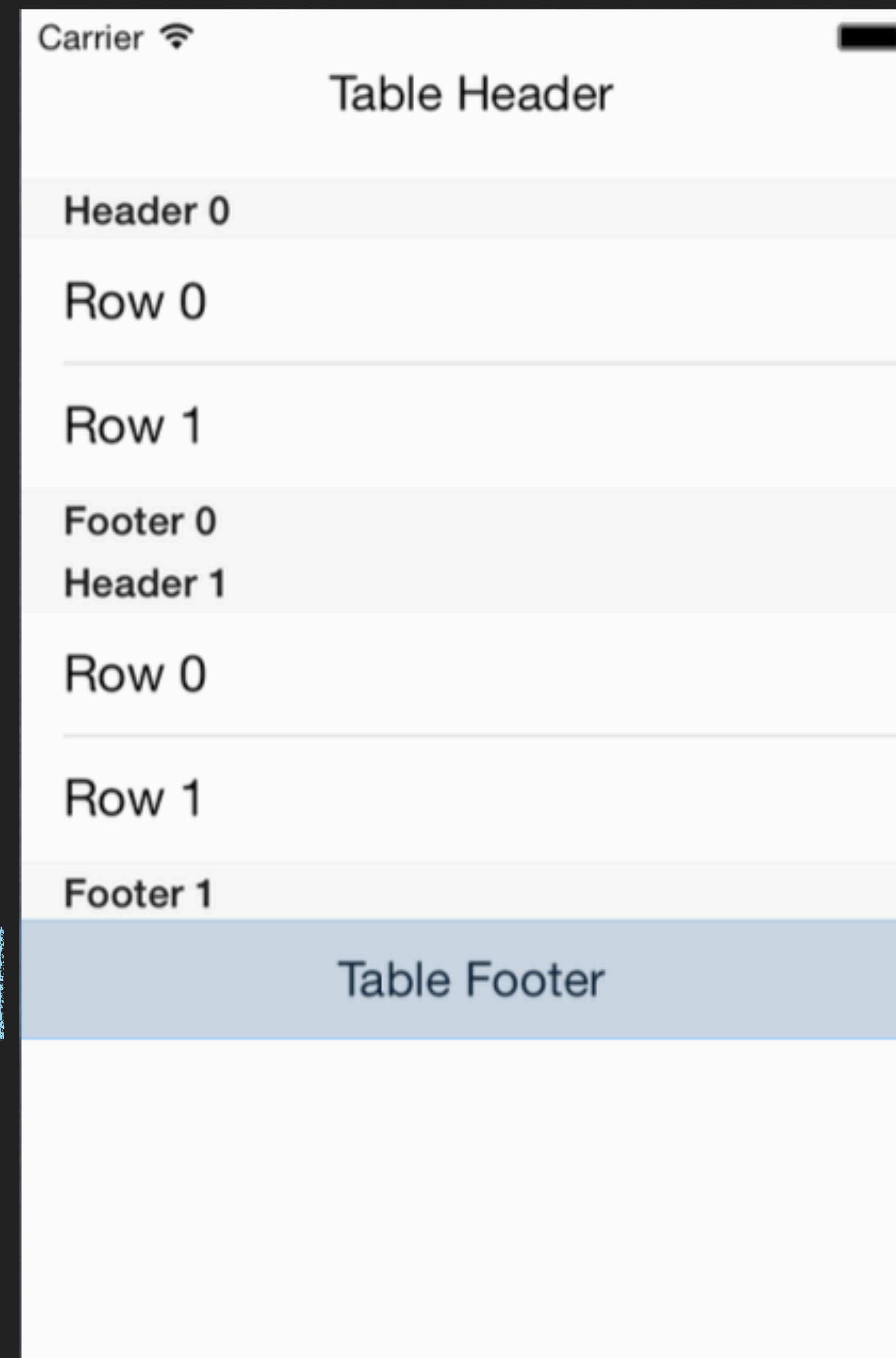
var tableHeaderView: UIView

# UITABLEVIEW

## PLAIN STYLE

CABEÇALHO DA TABELA  
(TABLE HEADER)

RODAPÉ DA TABELA  
(TABLE FOOTER)



var tableFooterView: UIView

# UITABLEVIEW

## PLAIN STYLE

CABEÇALHO DA TABELA  
(TABLE HEADER)

RODAPÉ DA TABELA  
(TABLE FOOTER)



SEÇÃO DA TABELA  
(SECTION)

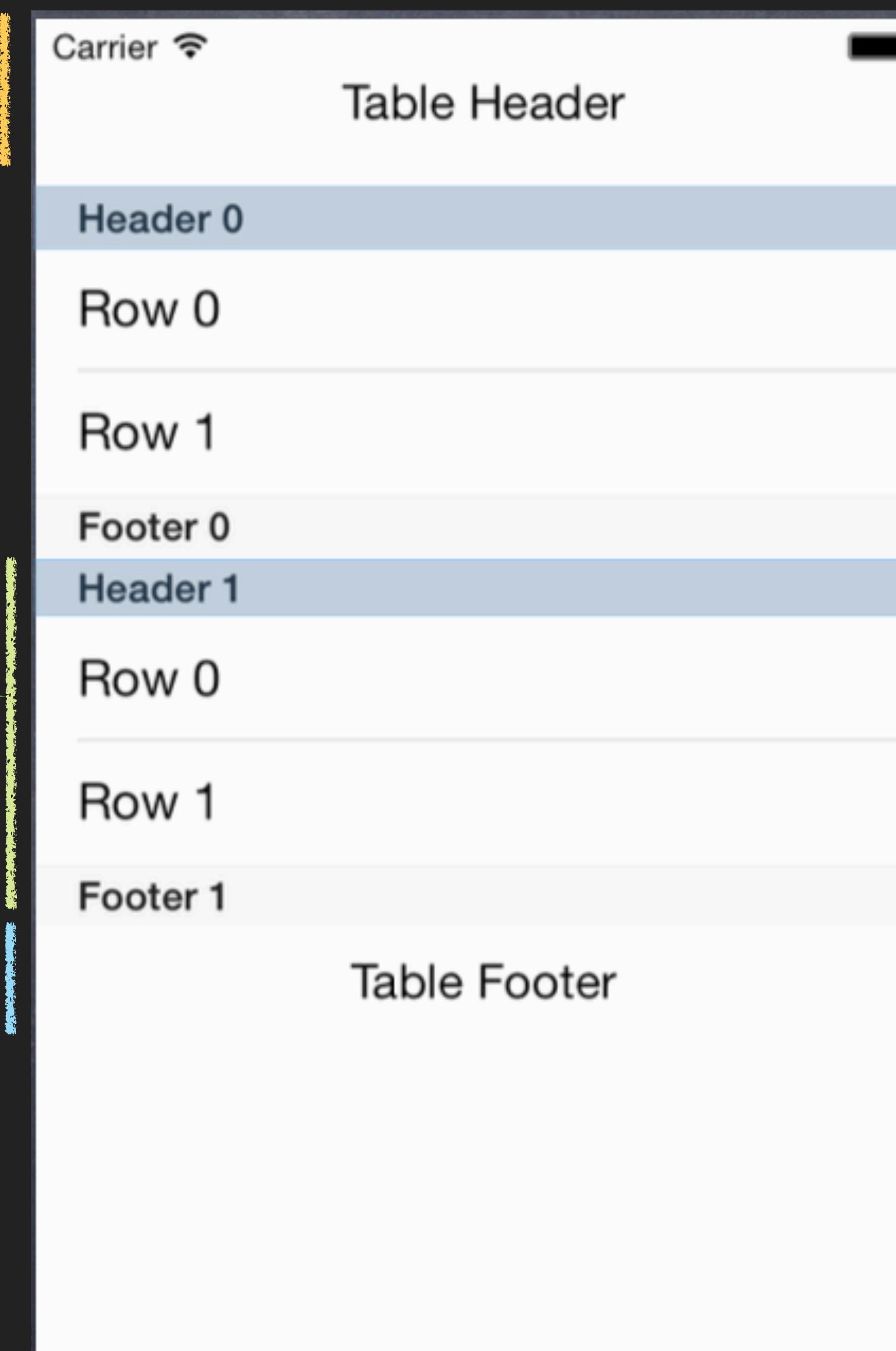
# UITABLEVIEW

## PLAIN STYLE

CABEÇALHO DA TABELA  
(TABLE HEADER)

SEÇÃO DA TABELA  
(SECTION)

RODAPÉ DA TABELA  
(TABLE FOOTER)



CABEÇALHOS DA SEÇÃO  
(SECTION HEADER)

Método no UITableViewDataSource - `tableView(UITableView, titleForHeaderInSection: Int) -> String`

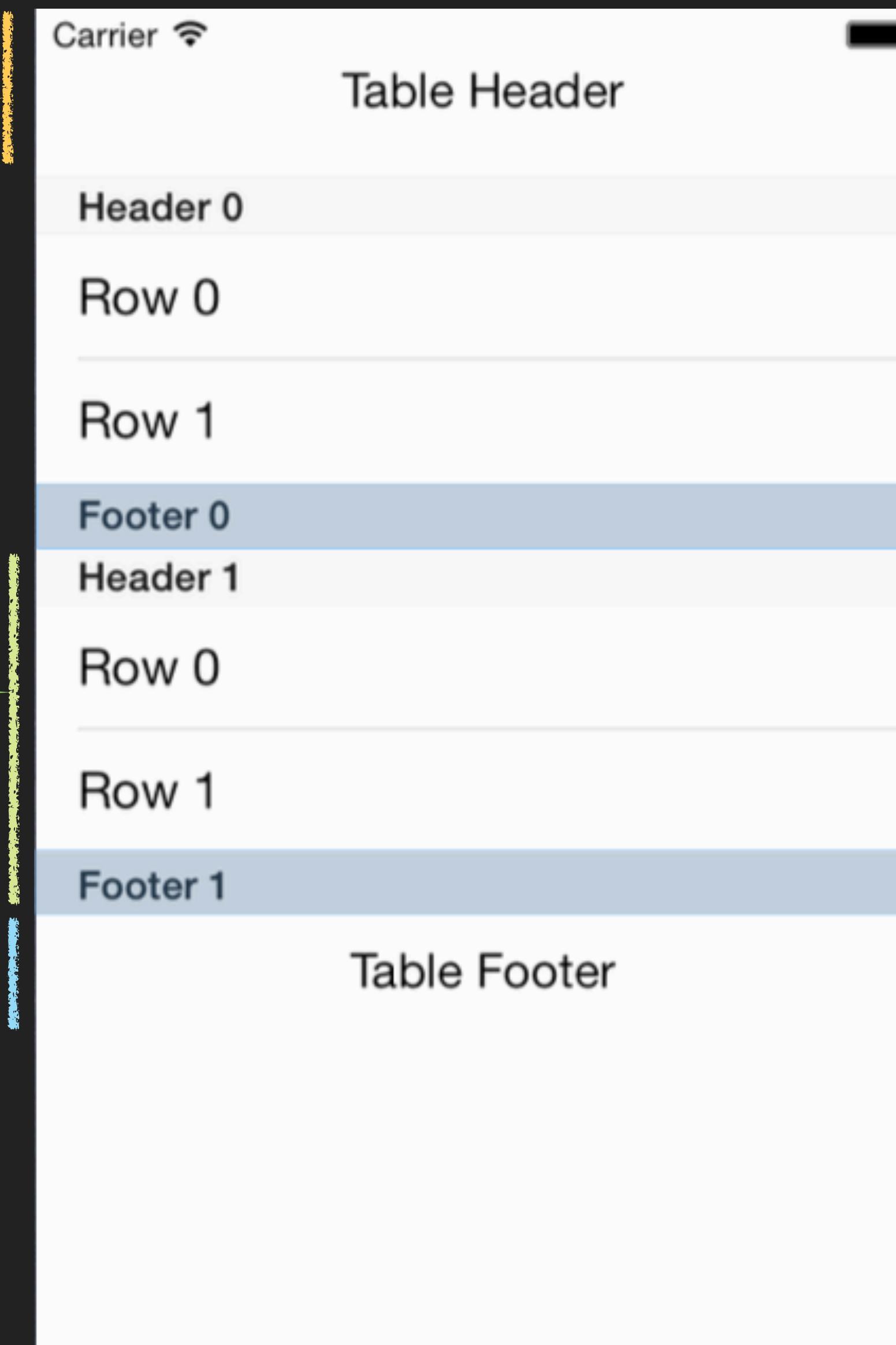
# UITABLEVIEW

## 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)

Método no UITableViewDataSource - `tableView(UITableView, titleForFooterInSection: Int) -> String`

# UITABLEVIEW

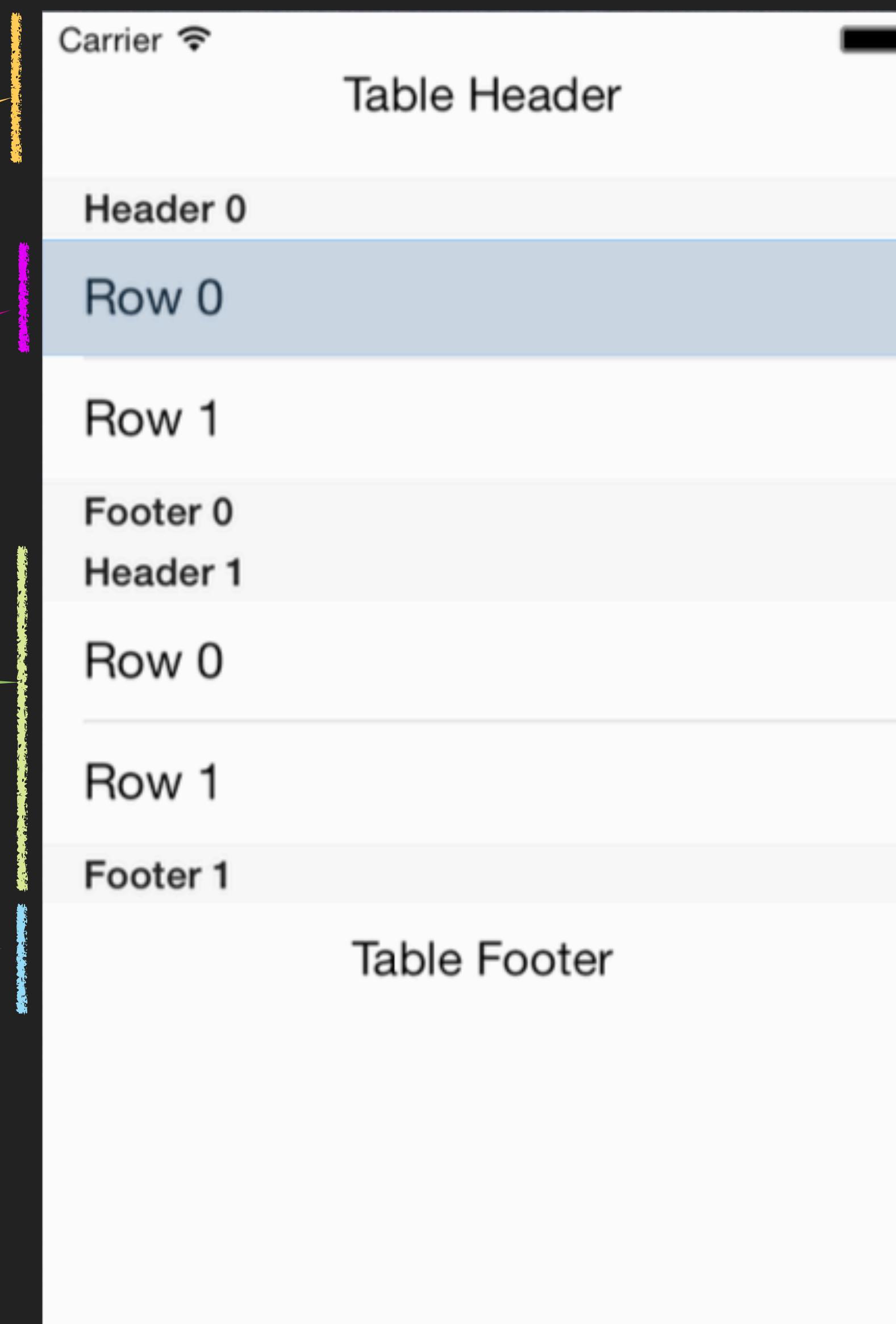
## PLAIN STYLE

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)

Método no UITableViewDataSource - `tableView(UITableView, cellForRowAt indexPath: IndexPath)`

# UITABLEVIEW

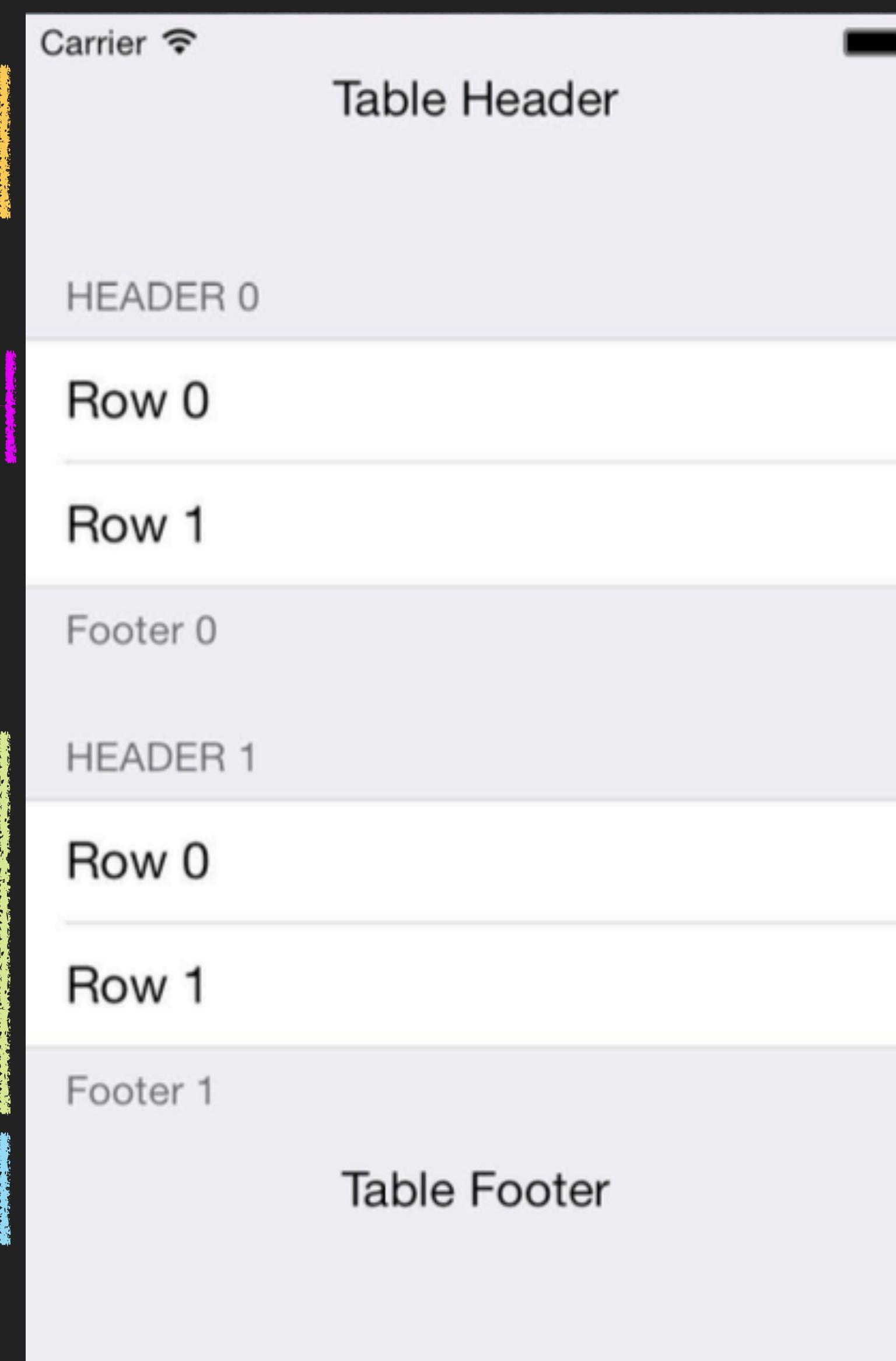
## GROUPED STYLE

CABEÇALHO DA TABELA  
(TABLE HEADER)

CÉLULA DA TABELA  
(TABLE CELL)

SEÇÃO DA TABELA  
(SECTION)

RODAPÉ DA TABELA  
(TABLE FOOTER)



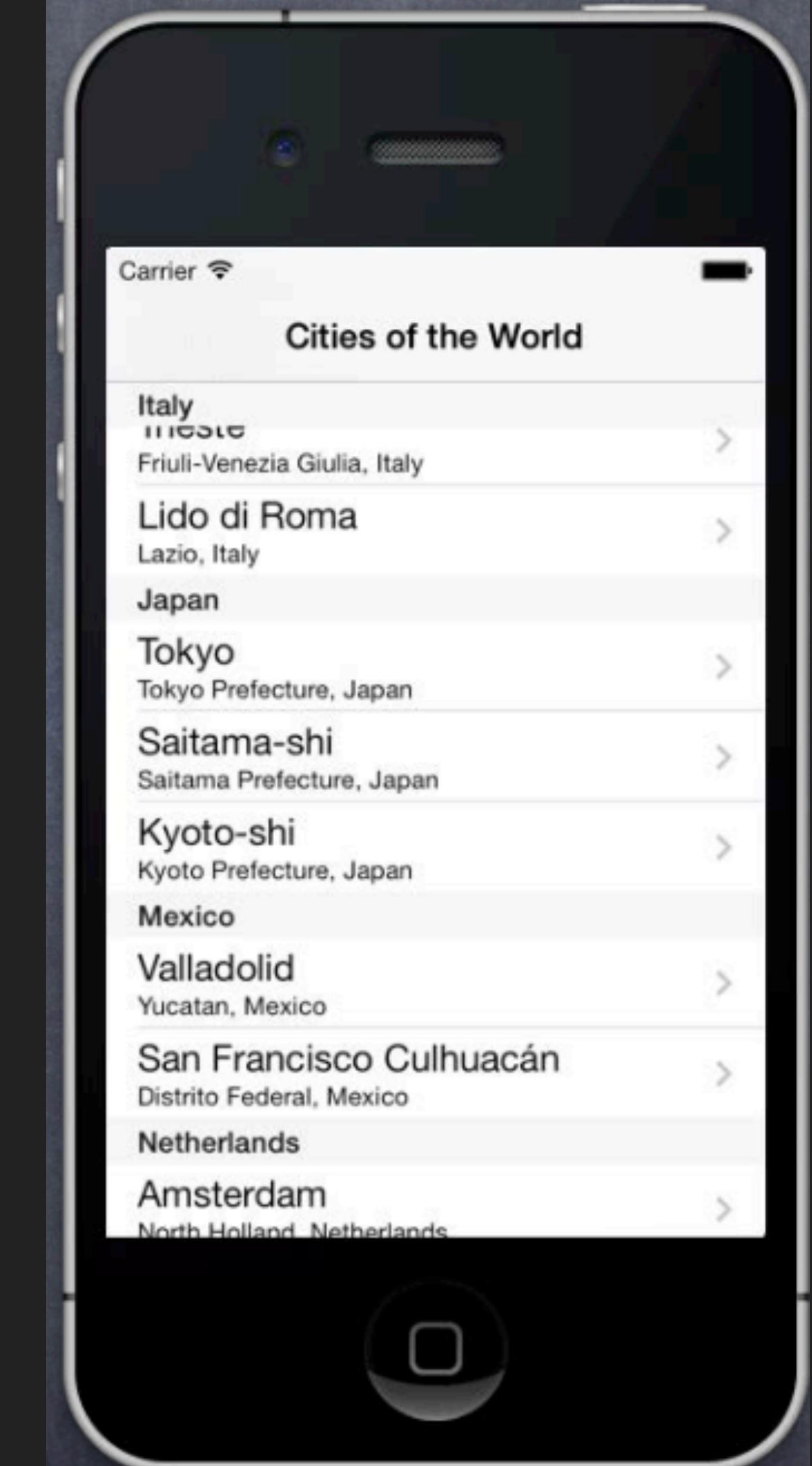
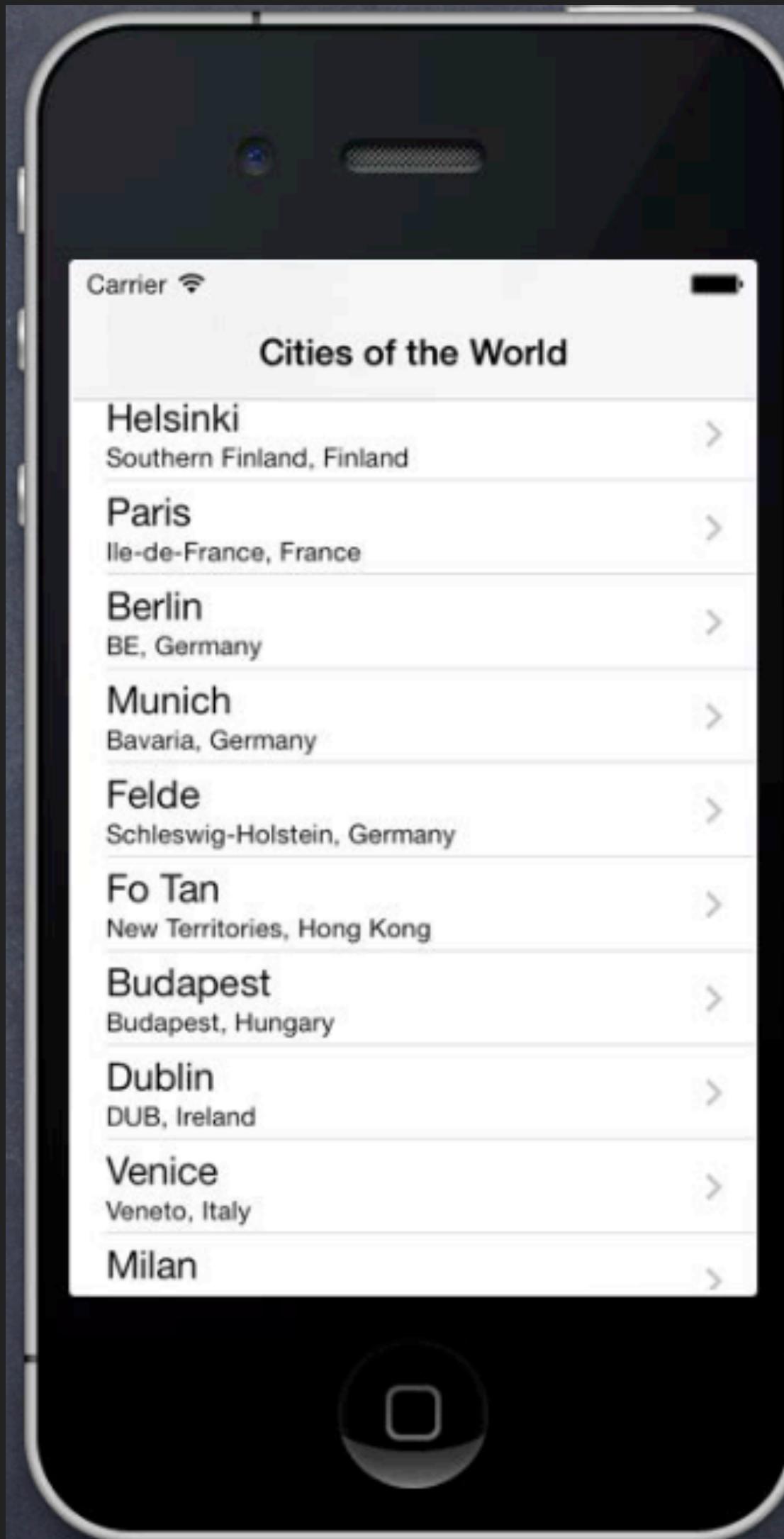
CABEÇALHOS DA SEÇÃO  
(SECTION HEADER)

RODAPÉS DA SEÇÃO  
(SECTION FOOTER)

TO BE OR NOT TO BE... THAT'S THE QUESTION!

# COM OU SEM SEÇÕES!?

SEM  
SEÇÕES



COM  
SEÇÕES

CADA UM NO SEU QUADRADO

## TIPOS DE CÉLULA PRÉ-DEFINIDOS (CELL STYLE)

Cities of the World	
Helsinki	>
Southern Finland, Finland	
Paris	>
Ile-de-France, France	
Berlin	>
BE, Germany	
Munich	>
Bavaria, Germany	
Felde	>
Schleswig-Holstein, Germany	
Fo Tan	>
New Territories, Hong Kong	
Budapest	>
Budapest, Hungary	
Dublin	>
DUB, Ireland	
Venice	>
Veneto, Italy	
Milan	>

### Subtitle

UITableViewCellStyle.subtitle

Cities of the World	
Sydney	>
Melbourne	>
Canberra	>
Anif	>
Vienna	>
Salzburg	>
Spontin	>
Ans	>
Bruxelles	>

### Basic

.basic

Cities of the World	
Sydney	New South Wales, Austr...
Melbourne	Victoria, Australia
Canberra	Australian Capital Terri...
Anif	Salzburg, Austria
Vienna	Vienna, Austria
Salzburg	SBG, Austria
Spontin	Namur, Belgium
Ans	Liege, Belgium
Bruxelles	Capital Region of Brus...

### Right Detail

.value1

Cities of the World	
Sydney	New South Wales, Austr...
Melbourne	Victoria, Australia
Canberra	Australian Capital Territo...
Anif	Salzburg, Austria
Vienna	Vienna, Austria
Salzburg	SBG, Austria
Spontin	Namur, Belgium
Ans	Liege, Belgium
Bruxelles	Capital Region of Bruss...

### Left Detail

.value2

A classe **UITableViewController** oferece um meio fácil para criar MVC de UITableView

O uso mais comum é quando desejamos que a UITableView ocupe toda a tela (`self.view`). Na verdade, quando usamos o **UITableViewController**, `self.view` é um **UITableViewCell**.

Você pode adicionar um no seu storyboard simplesmente arrastado da paleta.

No Selection

Navigation Controller - A controller that manages navigation through a hierarchy of views.

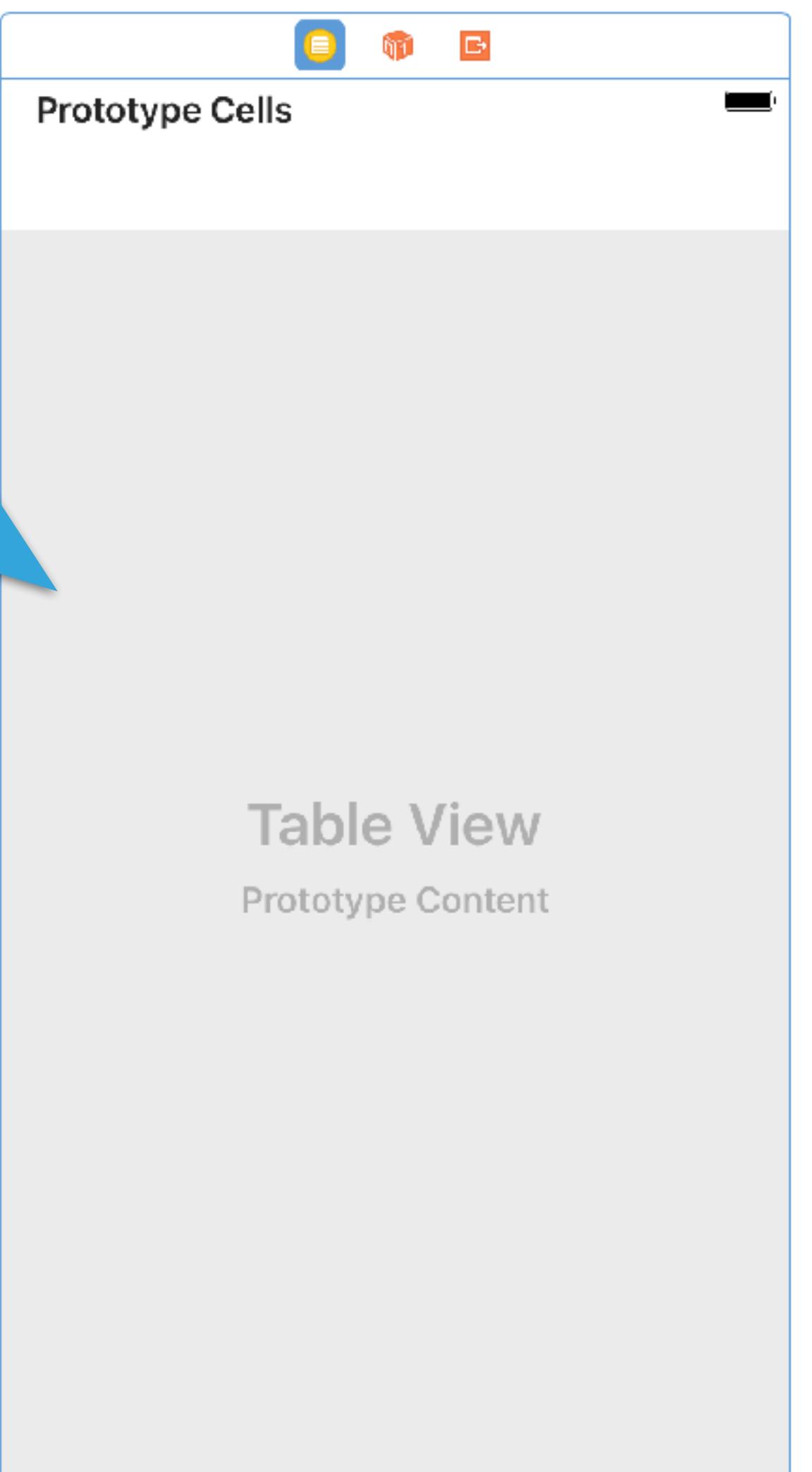
Table View Controller - A controller that manages a table view.

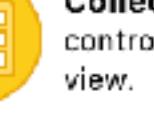
Collection View Controller - A controller that manages a collection view.

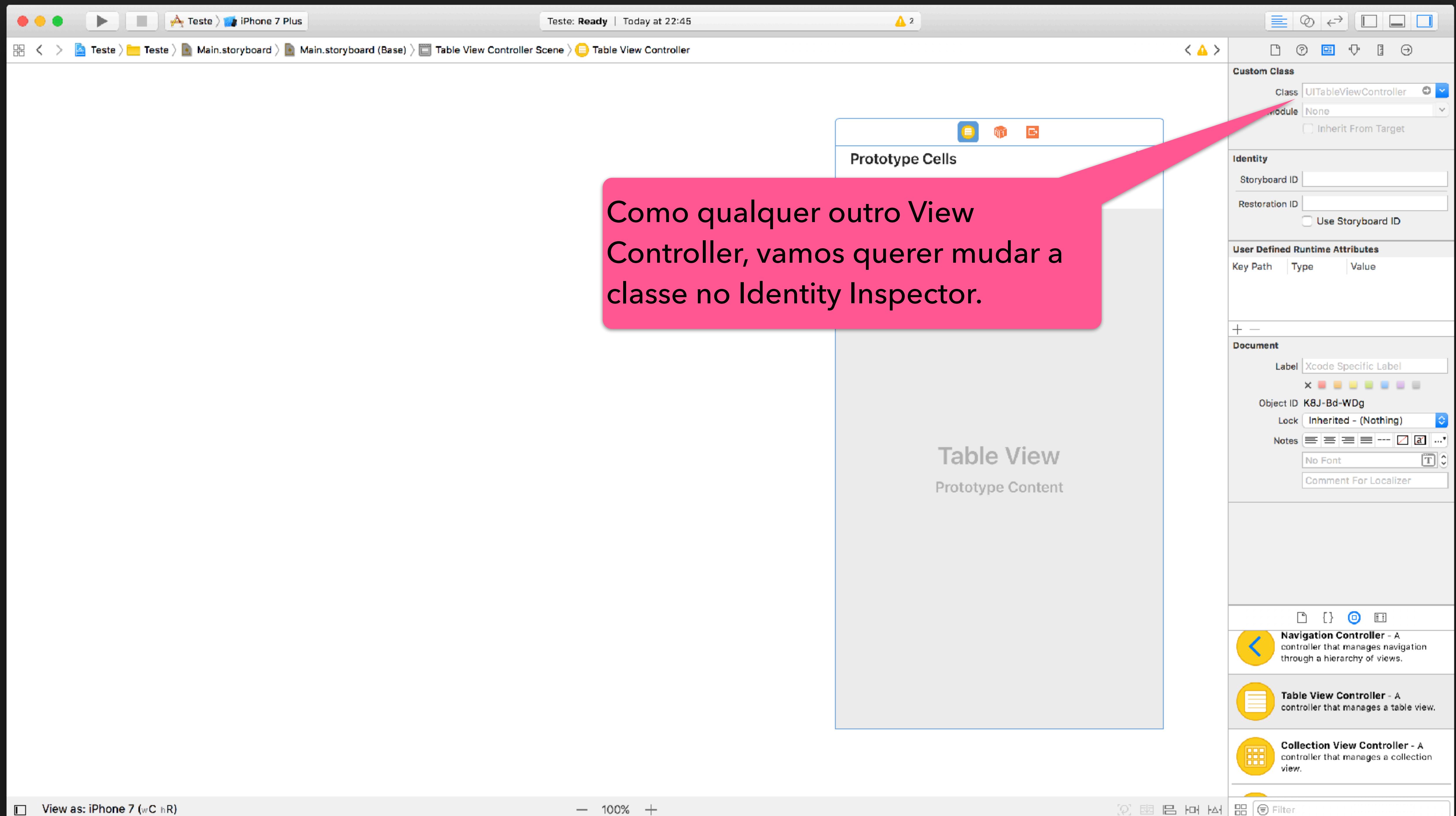
View Controller

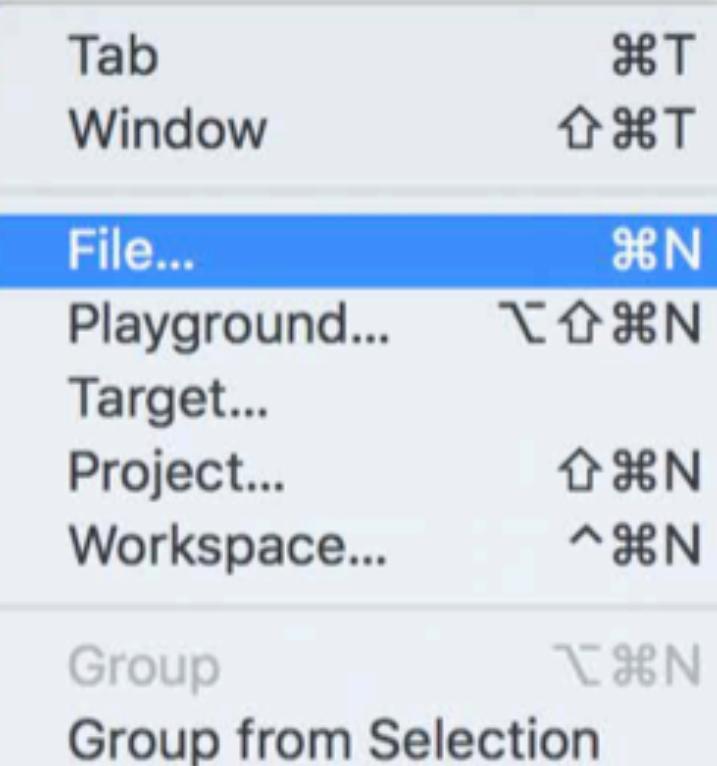
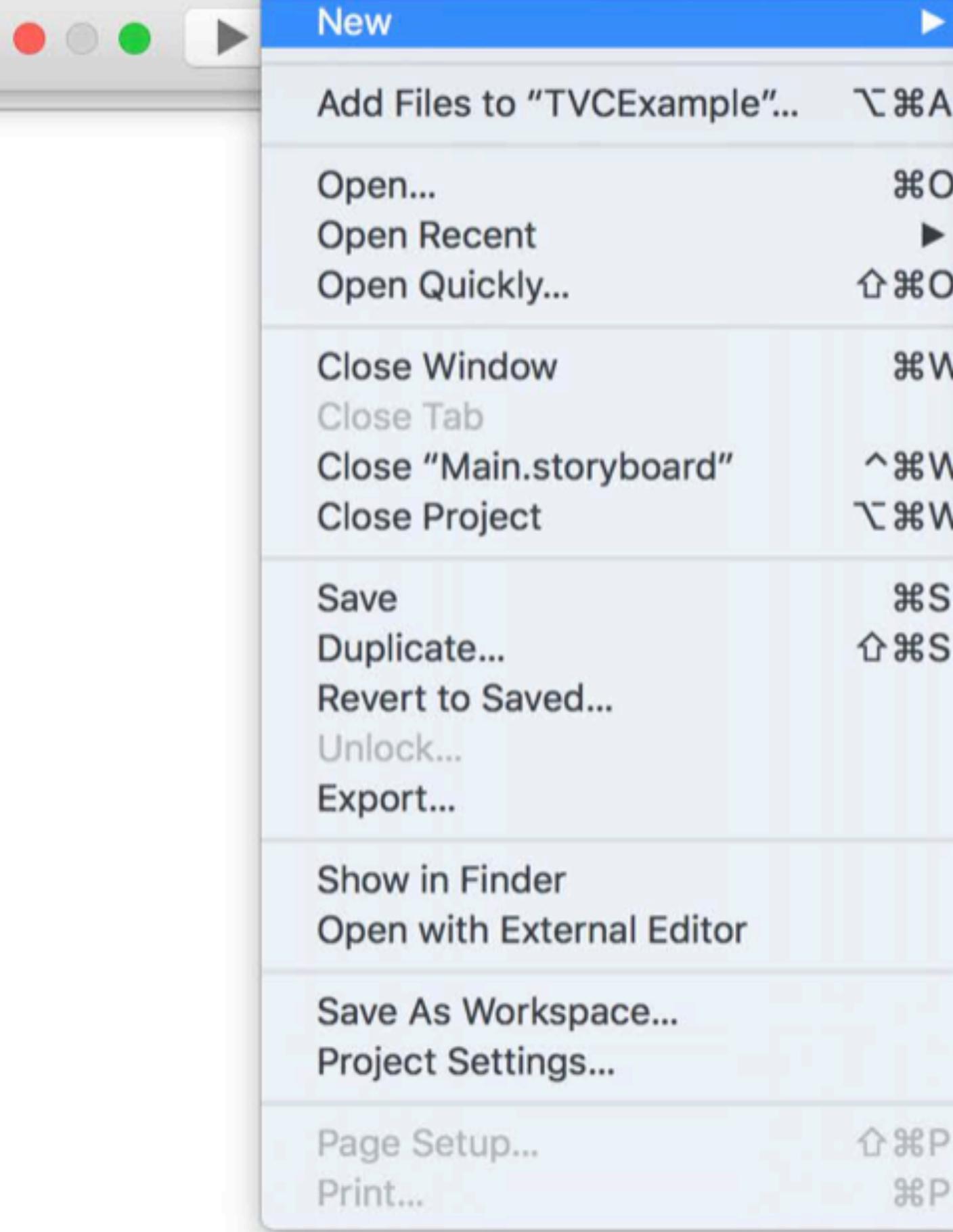
Simulated Size Fixed

O controller é: (uma subclasse de) UITableViewController  
A **view**: uma UITableView

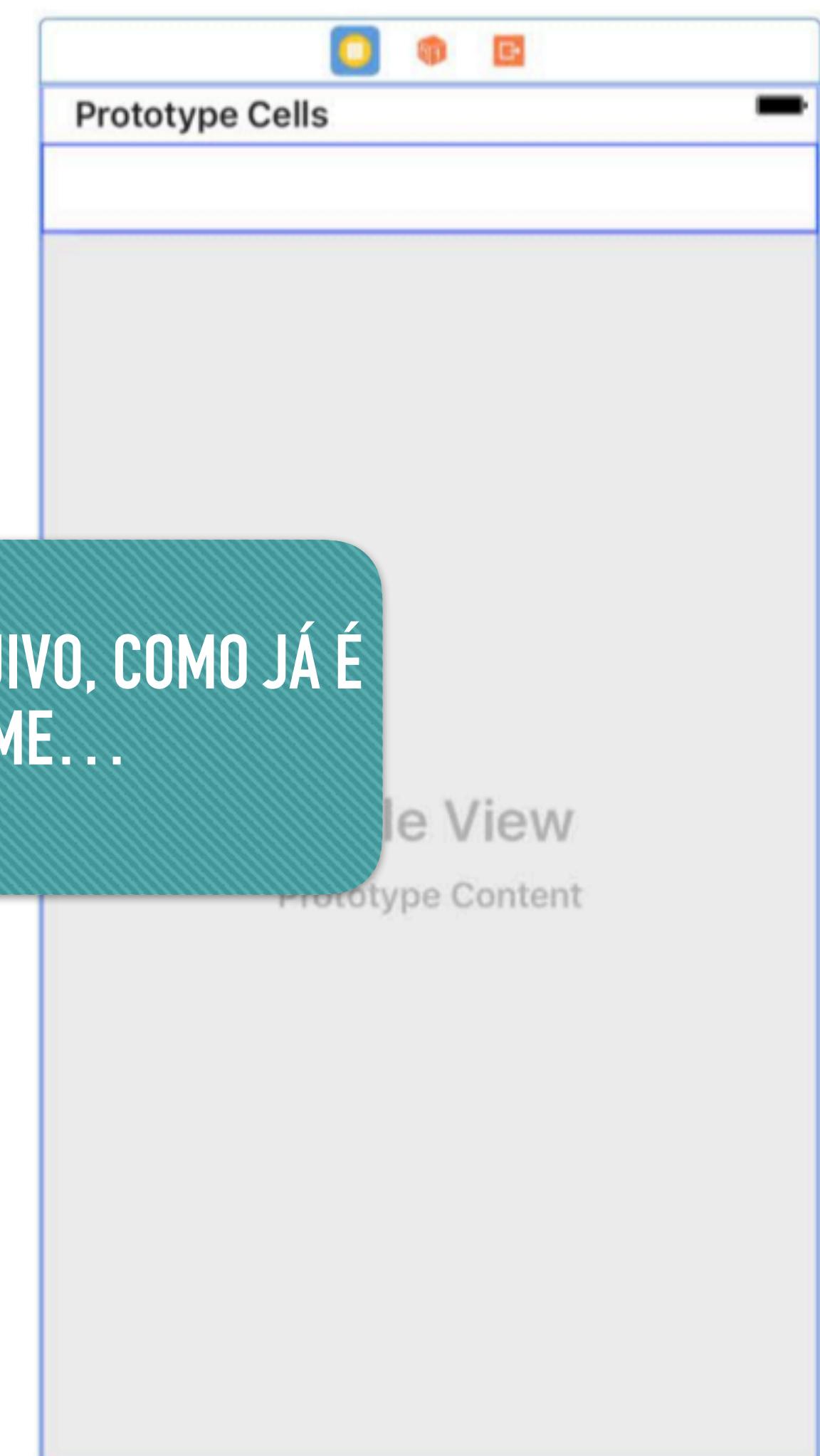


-  **Navigation Controller** - A controller that manages navigation through a hierarchy of views.
-  **Table View Controller** - A controller that manages a table view.
-  **Collection View Controller** - A controller that manages a collection view.





CRIE UM NOVO ARQUIVO, COMO JÁ É  
DE COSTUME...



Custom Class		
Class	UITableViewController	Module
Identity		
Storyboard ID		
Restoration ID		<input type="checkbox"/> Use Story
User Defined Runtime Attributes		
Key Path	Type	Value
Document		
Label	Xcode Spec	
Object ID	ab2-9w-rly	
Lock	Inherited -	()
		□ { } ○
	Navigation Controller	controller that manages through a hierarchy
	Table View Controller	controller that manages table view.
	Collection View Controller	controller that manages collection view.
	Tab Bar Controller	

Choose a template for your new file:

iOS

watchOS

tvOS

macOS

Filter

Source



Cocoa Touch  
Class



UI Test Case  
Class



Unit Test Case  
Class



Playground



Swift File



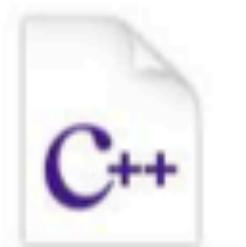
Objective-C File



Header File



C File



C++ File



Metal File

User Interface



Storyboard



View



Empty



Launch Screen

Cancel

Previous

Next

Choose options for your new file:

Class: `TableViewController`

Subclass of: `UITableViewController`

`UIView`

`UIViewController`

Language: `UITableViewController`

`UITableViewCell`

`UICollectionViewController`

Certifique-se de escolher a  
superclasse:  
**UITableViewController.**

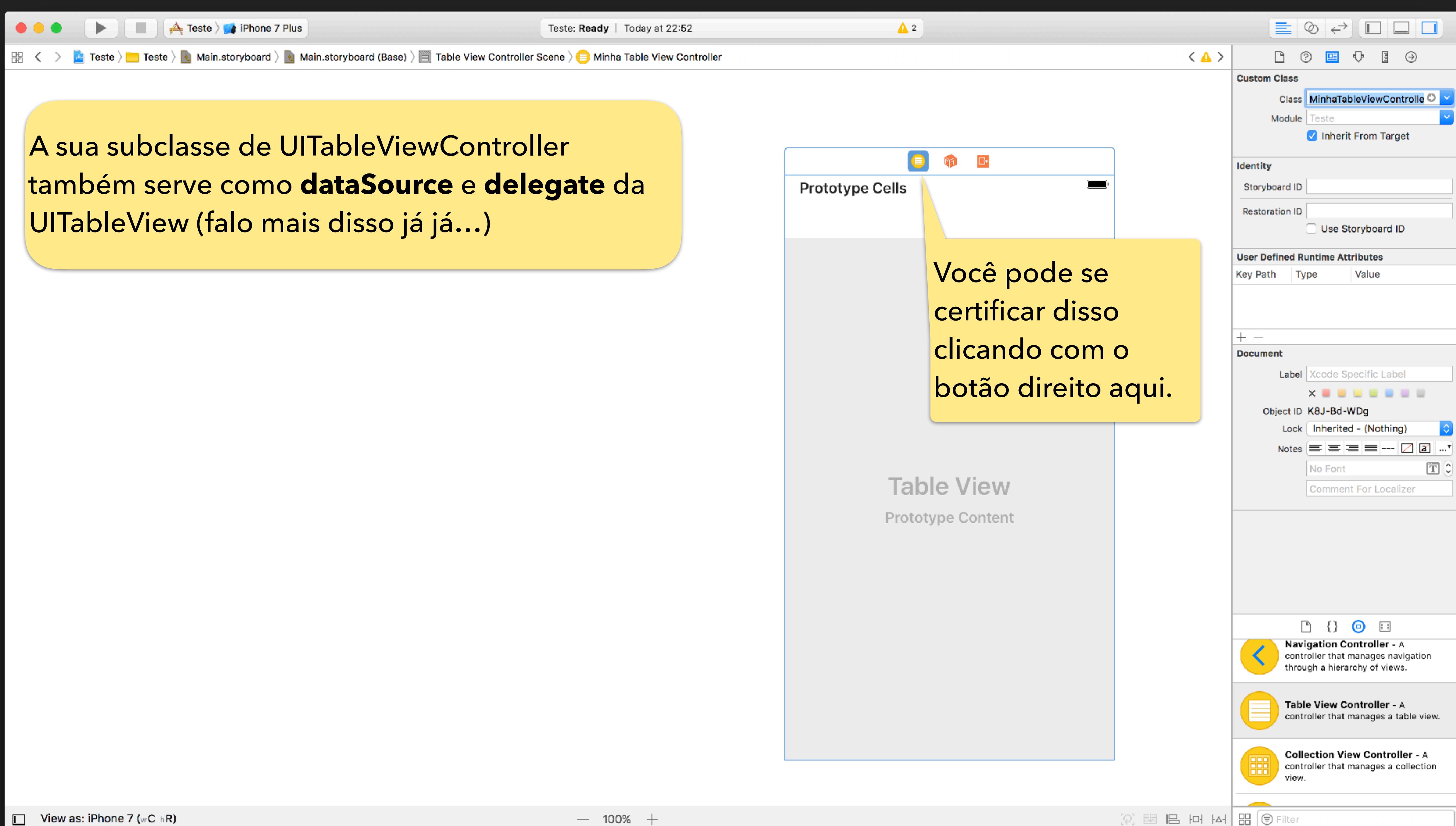
Cancel

Previous

Next

Para então indicar a classe correta aqui no Identity Inspector do storyboard.

The screenshot shows the Xcode interface with the storyboard editor open. The storyboard contains a single scene labeled "Table View Controller Scene". The table view controller's class is set to "MinhaTableViewController" in the Identity Inspector. A red callout points to the "Class" field in the Identity Inspector, highlighting the correct configuration. The storyboard editor shows the table view controller with its prototype content.



Propriedades **dataSource** e **delegate**.

Se você decidir usar uma UITableView sem um UITableViewController, você terá que ligar estas propriedades manualmente.

The screenshot shows the Xcode interface with the storyboard editor open. A UITableViewController is selected, and its connections are being viewed in the Connections Inspector. Two outlets, 'dataSource' and 'delegate', are connected to the UITableViewController's Table View. A green arrow points from the text 'Propriedades dataSource e delegate.' to these outlets. A blue arrow points from the text 'Se você decidir usar uma UITableView sem um UITableViewController, você terá que ligar estas propriedades manualmente.' to the same outlets. The storyboard navigation stack shows 'Teste' at the top, followed by 'Main.storyboard' and 'Main.storyboard (Base)'. The current scene is 'Table View Controller Scene' with 'Minha Table View Controller' selected. The status bar indicates 'Teste: Ready | Today at 22:59' and has a warning icon with '2'.

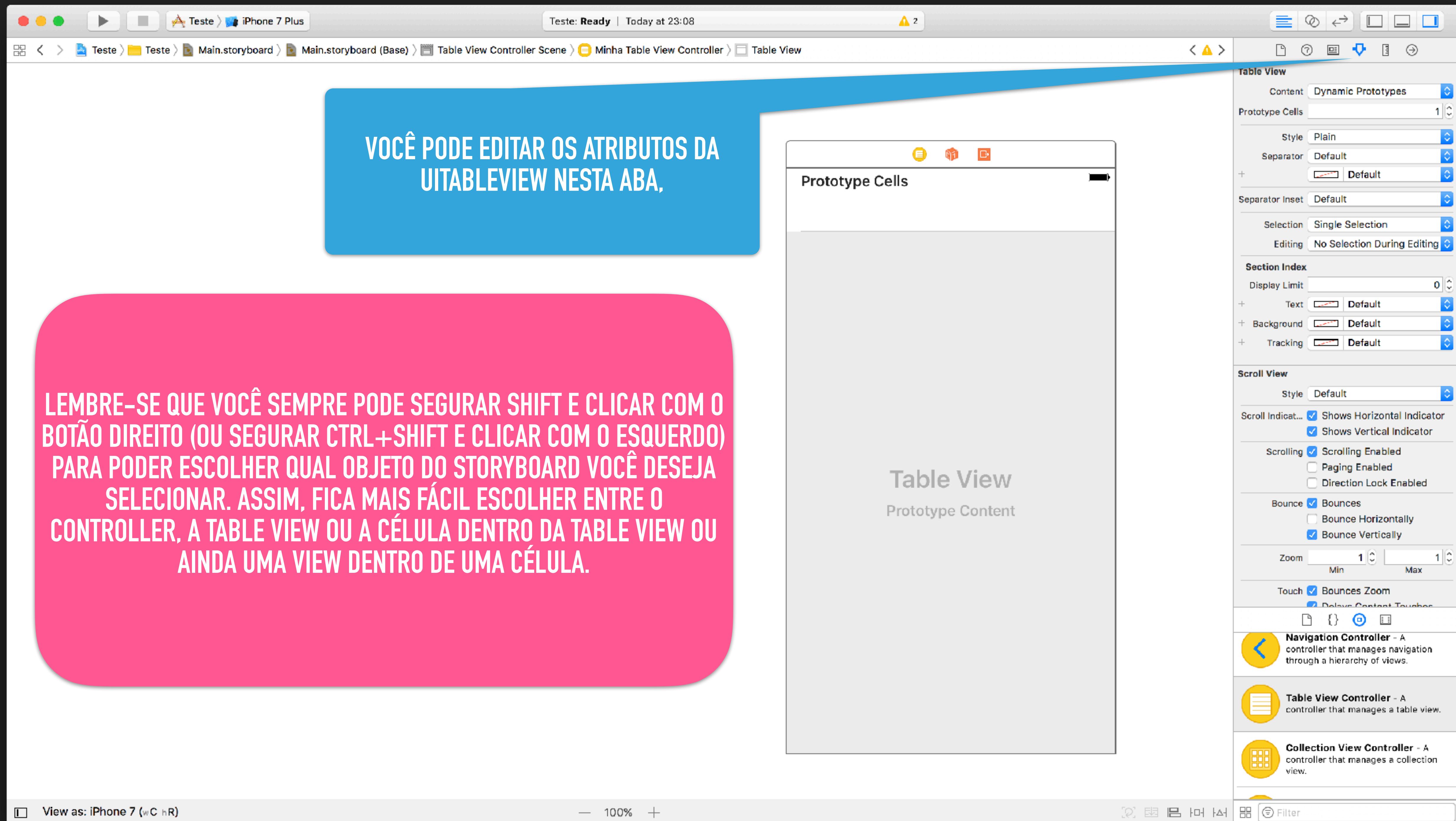


Table View

Content Dynamic Prototypes

Prototype Cells 1

Style Plain

Separator Default

+ Default

Separator Inset Default

Selection Single Selection

Editing No Selection During Editing

Section Index

Display Limit 0

+ Text Default

+ Background Default

+ Tracking Default

ScrollView

Style Default

Scroll Indicat...  Shows Horizontal Indicator  
 Shows Vertical Indicator

Scrolling  Scrolling Enabled  
 Paging Enabled  
 Direction Lock Enabled

Bounce  Bounces  
 Bounce Horizontally  
 Bounce Vertically

Zoom 1 Min Max

Touch  Bounces Zoom  
 Delays Content Touches

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

Filter

View as: iPhone 7 (wC hR)

— 100% +

Prototype Cells

UM ATRIBUTO IMPORTANTE É O STYLE (GROUPED OU PLAIN)

Table View

Prototype Content

OUTRO ATRIBUTO IMPORTANTE É O TIPO DE CONTEÚDO: DYNAMIC OU STATIC

Table View  
Prototype Content

Table View

Content Dynamic Prototypes

Prototype Cells 1

Style Plain

Separator Default

+ Default

Separator Inset Default

Selection Single Selection

Editing No Selection During Editing

Section Index

Display Limit 0

+ Text Default

+ Background Default

+ Tracking Default

Scroll View

Style Default

Scroll Indicat...  Shows Horizontal Indicator  
 Shows Vertical Indicator

Scrolling  Scrolling Enabled  
 Paging Enabled  
 Direction Lock Enabled

Bounce  Bounces  
 Bounce Horizontally  
 Bounce Vertically

Zoom 1 Min Max

Touch  Bounces Zoom  
 Delays Content Touches

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

View as: iPhone 7 (wC hR) — 100% + ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ Filter



TVCEExample > iPhone 7

TVCEExample: Ready

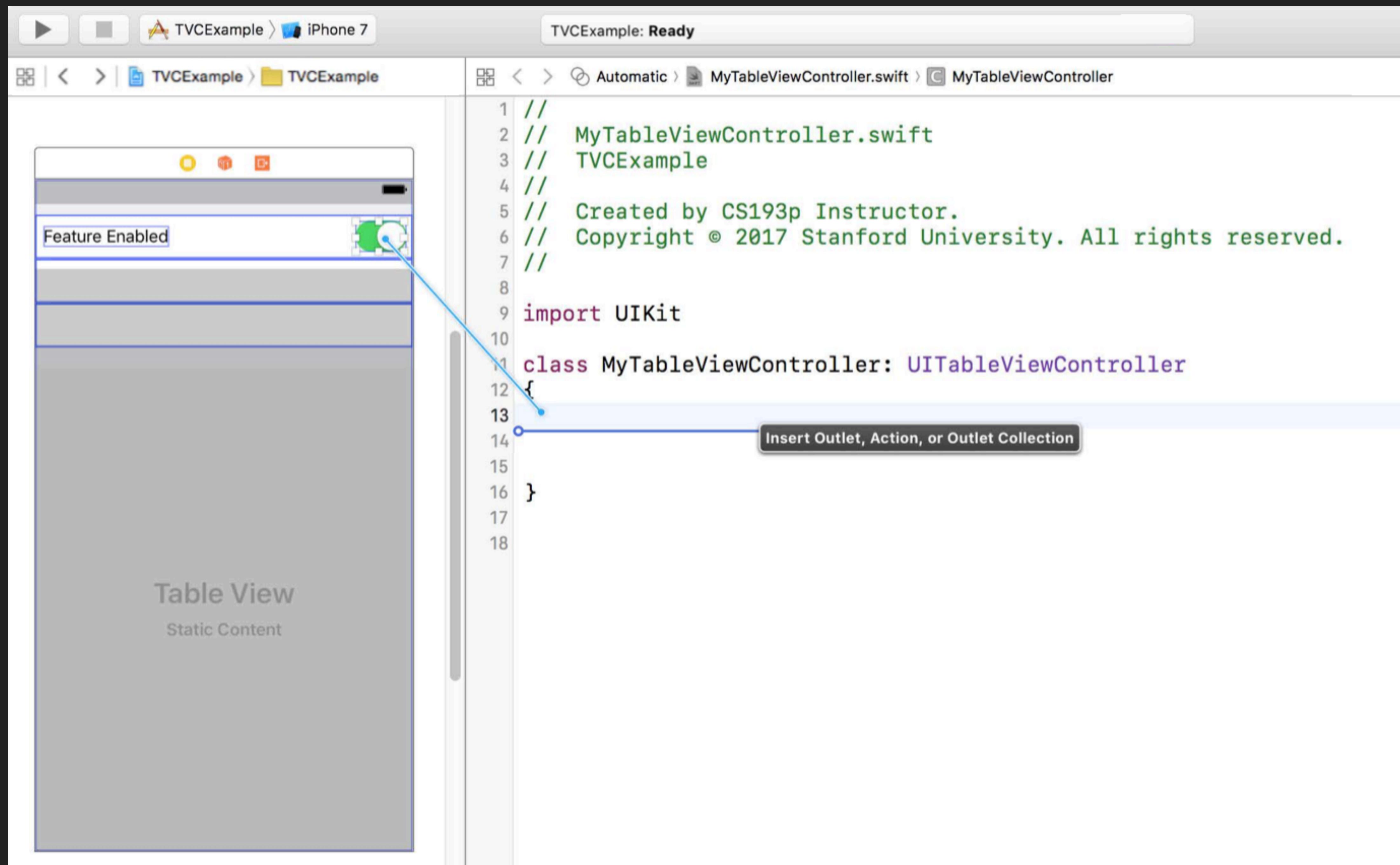
TVCEExample > TVCEExample

```
//  
// MyTableViewController.swift  
// TVCEExample  
//  
// Created by CS193p Instructor.  
// Copyright © 2017 Stanford University. All rights reserved.  
  
import UIKit  
  
class MyTableViewController: UITableViewController  
{  
    ○ 13  
    ○ 14 Insert Outlet, Action, or Outlet Collection  
    ○ 15  
    ○ 16 }  
    ○ 17  
    ○ 18
```

Feature Enabled

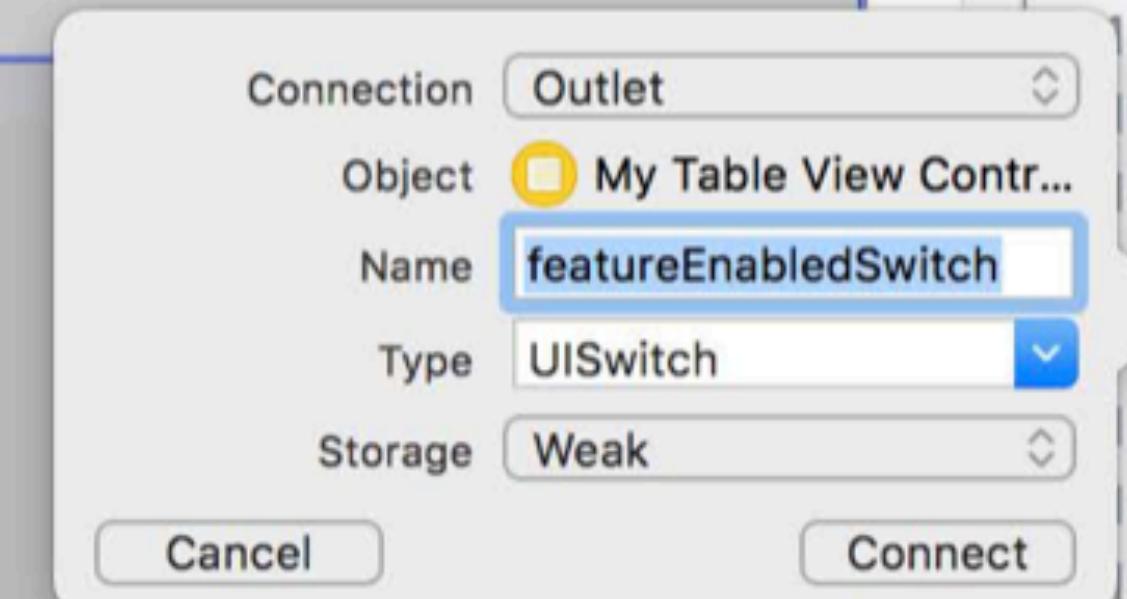
Table View

Static Content



TVCExample &gt; TVCExample

Automatic &gt; MyTableViewController.swift &gt; MyTableViewController



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

TVCEExample > iPhone 7

TVCEExample: Ready

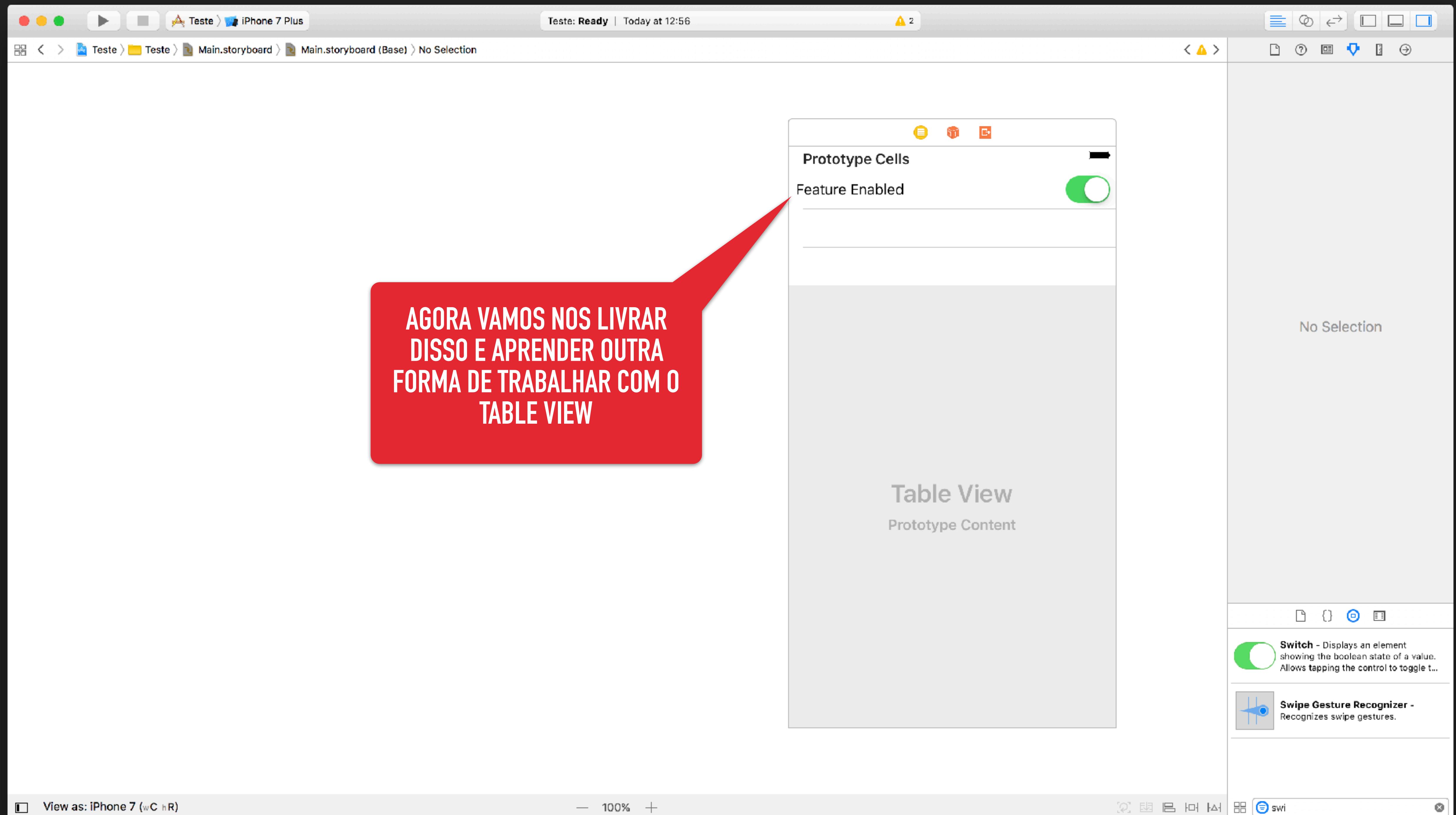
TVCEExample > TVCEExample > MyTableViewController.swift > MyTableViewController

The screenshot shows the Xcode interface with two main panes. The left pane displays a storyboard scene titled 'Table View' with 'Static Content'. It contains a single table view cell with the title 'Feature Enabled' and a green switch icon. The right pane shows the 'MyTableViewController.swift' file under the 'TVCEExample' project. The code is as follows:

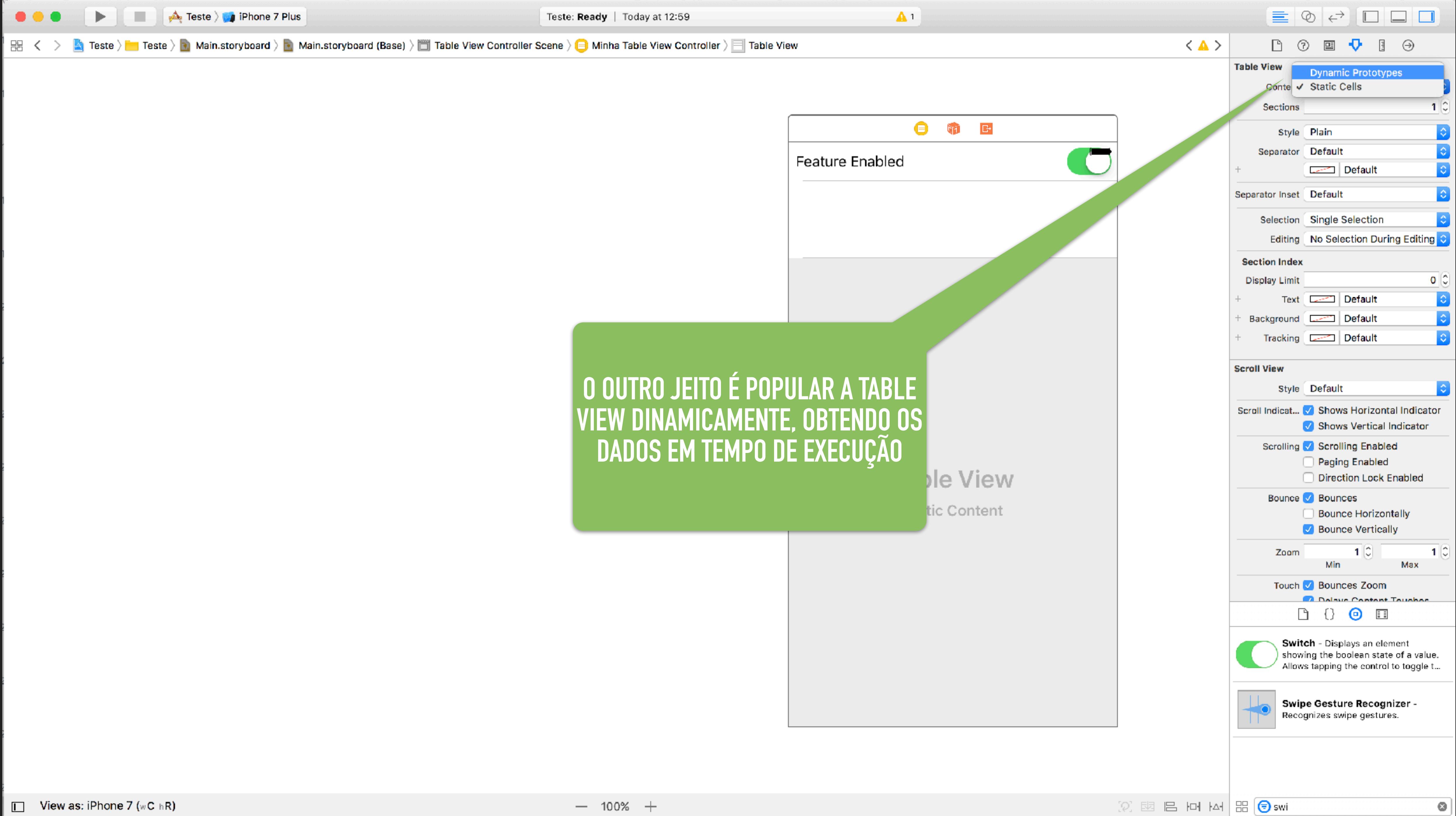
```
// MyTableViewController.swift
// TVCEExample
//
// Created by CS193p Instructor.
// Copyright © 2017 Stanford University. All rights reserved.

import UIKit

class MyTableViewController: UITableViewController {
    @IBOutlet weak var featureEnabledSwitch: UISwitch!
}
```



**AGORA VAMOS NOS LIVRAR  
DISSO E APRENDER OUTRA  
FORMA DE TRABALHAR COM O  
TABLE VIEW**



Teste > iPhone 7 Plus

Teste: Ready | Today at 13:01

Table View

Content: Static Cells

Sections: 1

Style: Plain

Separator: Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Editing

Section Index

Display Limit: 0

Text: Default

Background: Default

Tracking: Default

Scroll View

Style: Default

Scroll Indicat...:

- Shows Horizontal Indicator
- Shows Vertical Indicator

Scrolling:

- Scrolling Enabled
- Paging Enabled
- Direction Lock Enabled

Bounce:

- Bounces
- Bounce Horizontally
- Bounce Vertically

Zoom:

Touch:

- Bounces Zoom
- Delays Content Touches

Switch - Displays an element showing the boolean state of a value. Allows tapping the control to toggle t...

Swipe Gesture Recognizer - Recognizes swipe gestures.

Feature Enabled

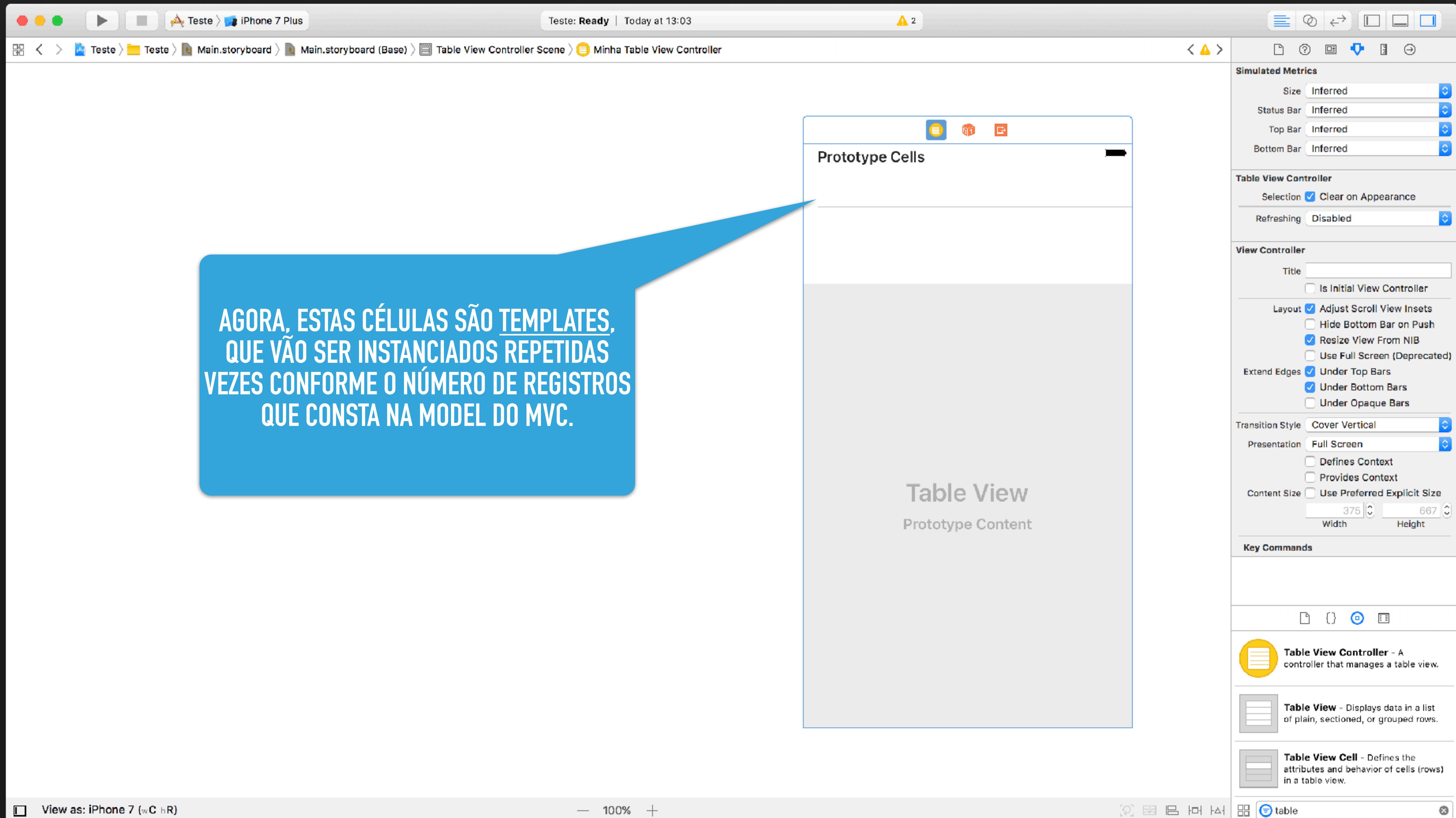
E ISSO É FEITO, NA MAIORIA DAS VEZES, USANDO O ESTILO PLAIN (PLAIN STYLE)

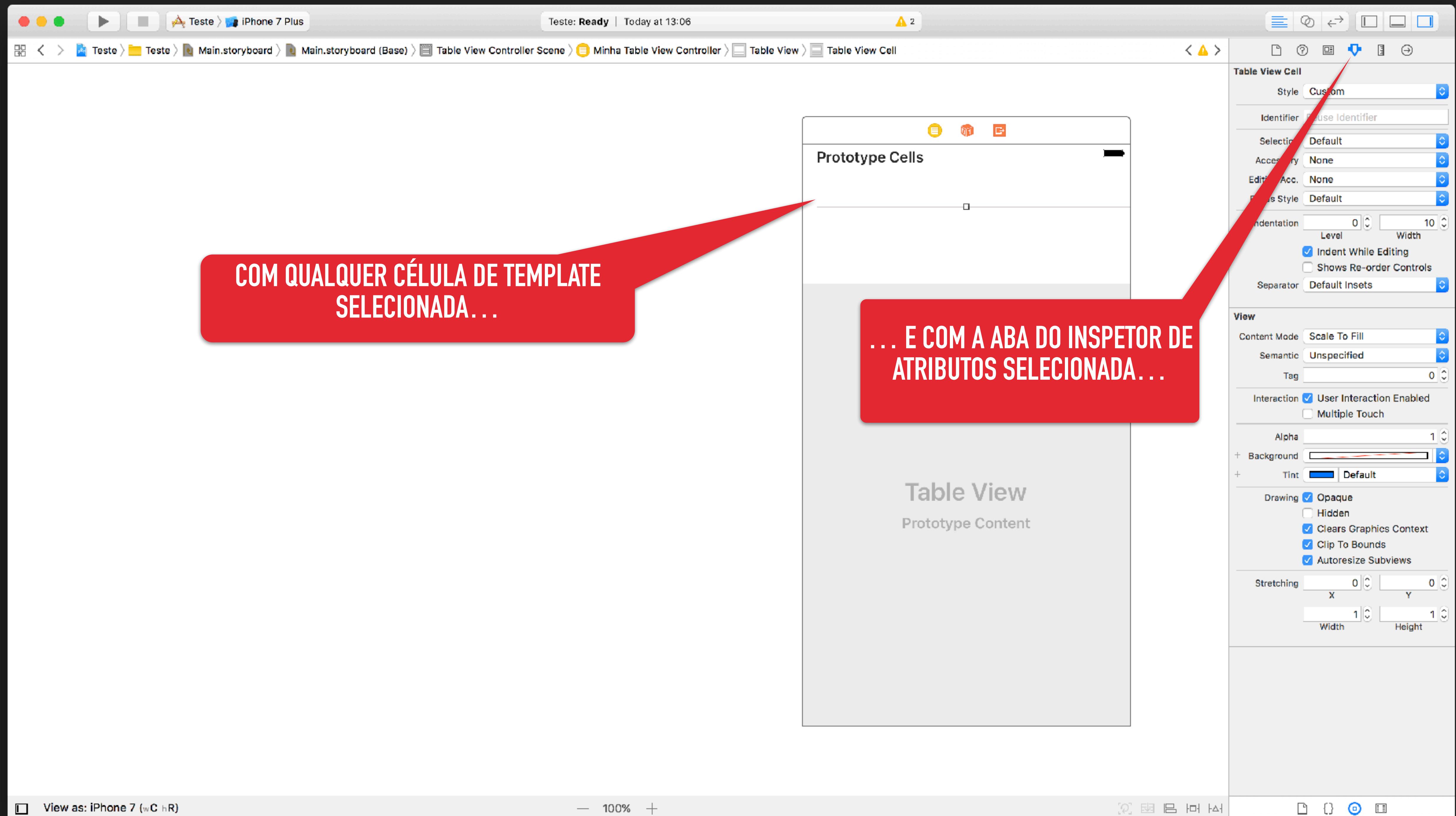
Table View

Static Content

View as: iPhone 7 (wC hR) 100% +

swi





**Table View Cell**

Style ✓ Custom

Identifier Basic

Selection Right Detail

Accessory Left Detail

Accessory Subtitle

Editing Accessory None

Focus Style Default

Indentation 0 Level 10

✓ Indent While Editing

Shows Re-order Controls

Separator Default Insets

**View**

Content Mode Scale To Fill

Semantic Unspecified

Tag 0

Interaction ✓ User Interaction Enabled

Multiple Touch

Alpha 1

+ Background

Tint Default

Drawing Opaque

Hidden

Clears Graphics Context

Clip To Bounds

Autoresize Subviews

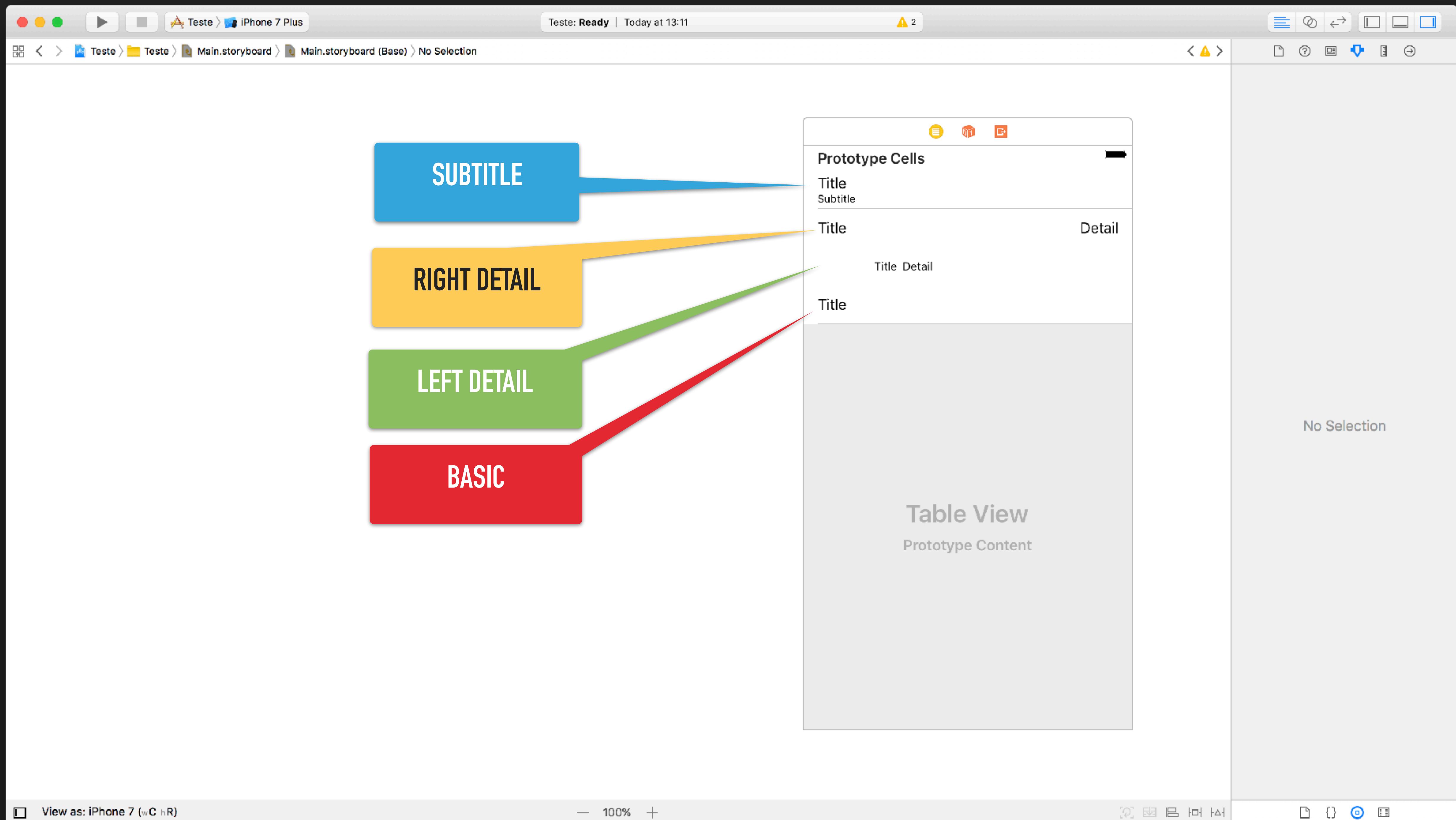
Stretching 0 X 0 Y

Width 1 Height 1

... VOCÊ PODE ESCOLHER AQUI O ESTILO DA  
CÉLULA TEMPLATE (CELL STYLE)

**Table View**

Prototype Content



**Table View Cell**

Style Subtitle

Image Image

Identifier Reuse Identifier

Selection Default

Accessory  None

Disclosure Indicator

Editing Accessory  Detail Disclosure

Focus Style Checkmark

Detail

Indentation Level Width

 Indent While Editing Shows Re-order Controls

Separator Default Insets

**View**

Content Mode Scale To Fill

Semantic Unspecified

Tag 0

Interaction  User Interaction Enabled Multiple Touch

Alpha 1

+ Background

+ Tint

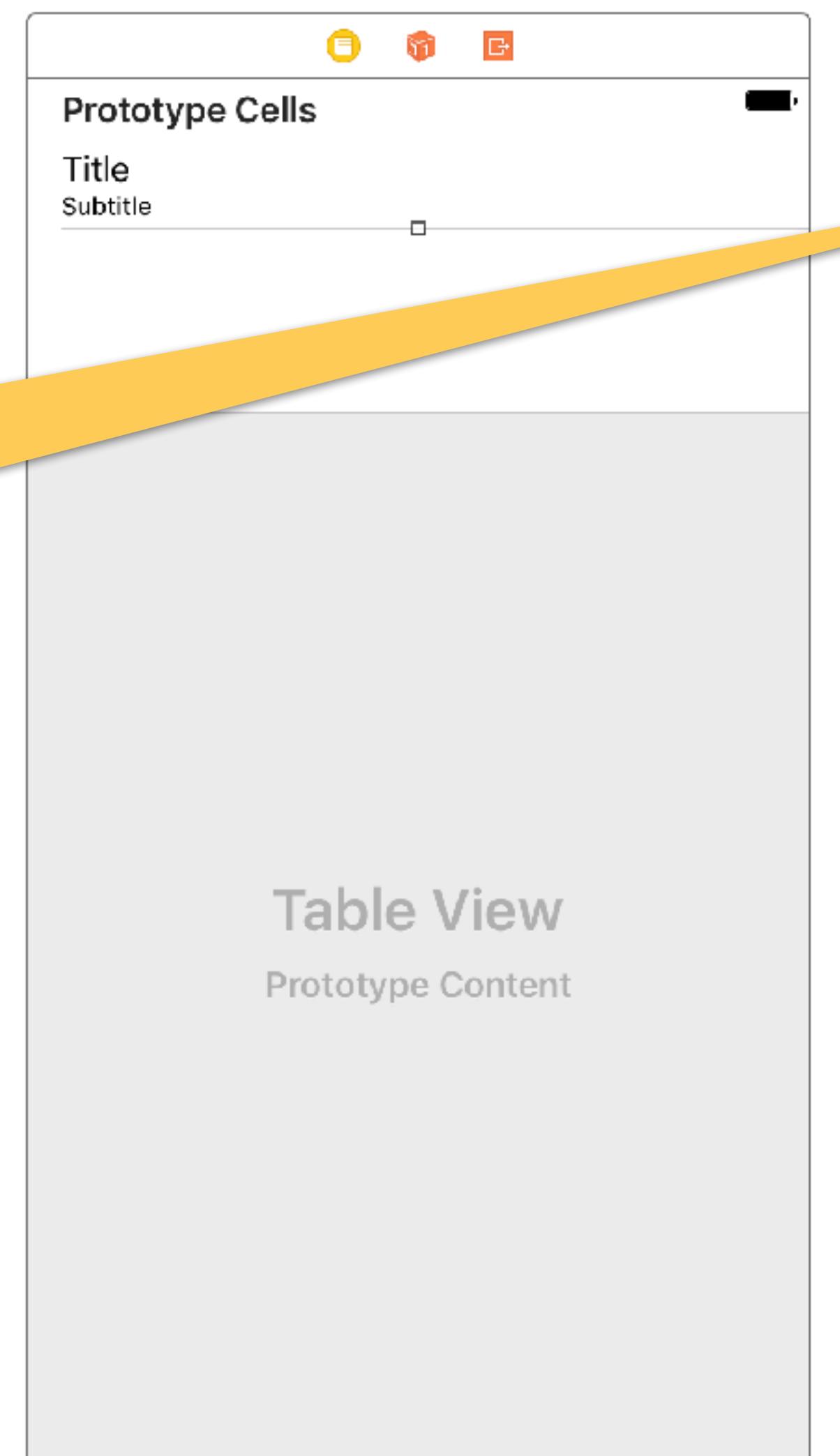
Default

Drawing  Opaque Hidden Clears Graphics Context Clip To Bounds AutoresizesSubviews

Stretching X 0 Y 0

Width 1 Height 1

TAMBÉM É POSSÍVEL  
COLOCAR UM ACESSÓRIO NA  
PARTE DIREITA DA CÉLULA.



**Table View Cell**

Style Subtitle

Image Image

Identifier Reuse Identifier

Selection Default

Accessory  None

Disclosure Indicator

Editing Accessory  Detail Disclosure

Focus Style Checkmark

Indentation Detail

Indent While Editing Shows Re-order Controls 

Separator Default Insets

**View**

Content Mode Scale To Fill

Semantic Unspecified

Tag 0

Interaction  User Interaction EnabledMultiple Touch 

Alpha 1

+ Background

+ Tint Default

Drawing  Opaque Hidden Clears Graphics Context Clip To Bounds AutoresizesSubviews

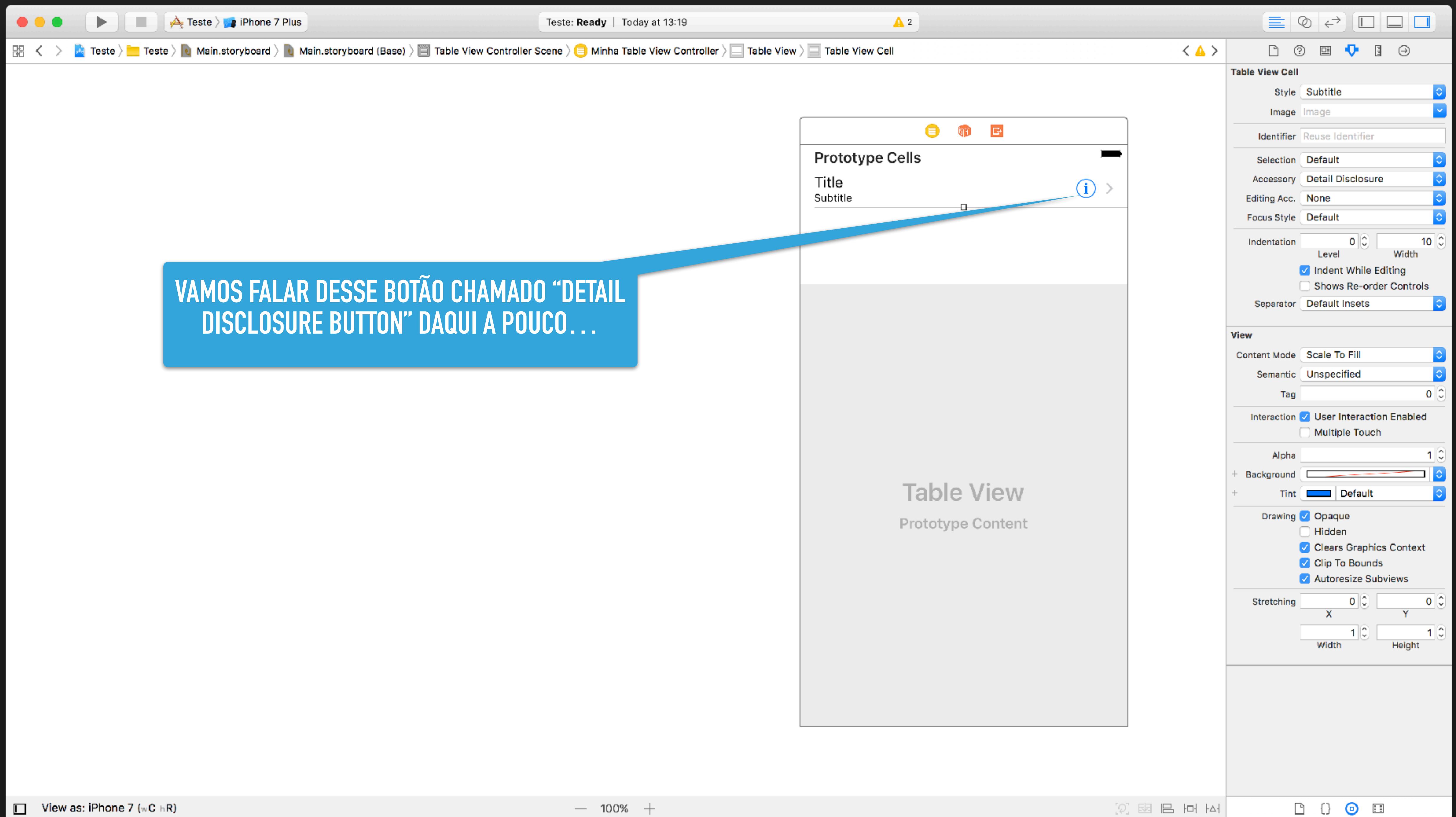
Stretching X 0 Y 0

Width 1 Height 1

TAMBÉM É POSSÍVEL  
COLOCAR UM ACESSÓRIO NA  
PARTE DIREITA DA CÉLULA.

Table View  
Prototype Content

E ESTE AQUI É DE UM TIPO  
ESPECIAL...



**Table View Cell**

Style ✓ Custom

Identifier Basic

Selection Right Detail

Accessory Left Detail

Accessory Subtitle

Editing Accessory None

Focus Style Default

Indentation 0 Level 10 Width

 Indent While Editing Shows Re-order Controls

Separator Default Insets

**View**

Content Mode Scale To Fill

Semantic Unspecified

Tag 0

 User Interaction Enabled Multiple Touch

Alpha 1

+ Background

+ Tint Default

Drawing  Opaque Hidden Clears Graphics Context Clip To Bounds Autoresizes Subviews

Stretching 0 X 0 Y

Width 1 Height 1

UM DOS TIPOS DE CÉLULA QUE VOCÊ PODE  
ESCOLHER É O CUSTOM.

**Table View**

Prototype Content

Teste: Ready | Today at 13:09

Teste > Teste > Main.storyboard > Main.storyboard (Base) > Table View Controller Scene > Minha Table View Controller > Table View > Table View Cell

Table View Cell

Style: Custom

Identifier: Basic

Selection: Right Detail

Accessory: Left Detail

Accessories: Subtitle

Editing Access: None

Focus Style: Default

Indentation: 0 Level: 10 Width:

Indent While Editing

Shows Re-order Controls

Separator: Default Insets

View

Content Mode: Scale To Fill

Semantic: Unspecified

Tag: 0

Interaction:  User Interaction Enabled

Multiple Touch

Alpha: 1

+ Background: Red dashed line

+ Tint: Default

Drawing:  Opaque

Hidden

Clears Graphics Context

Clip To Bounds

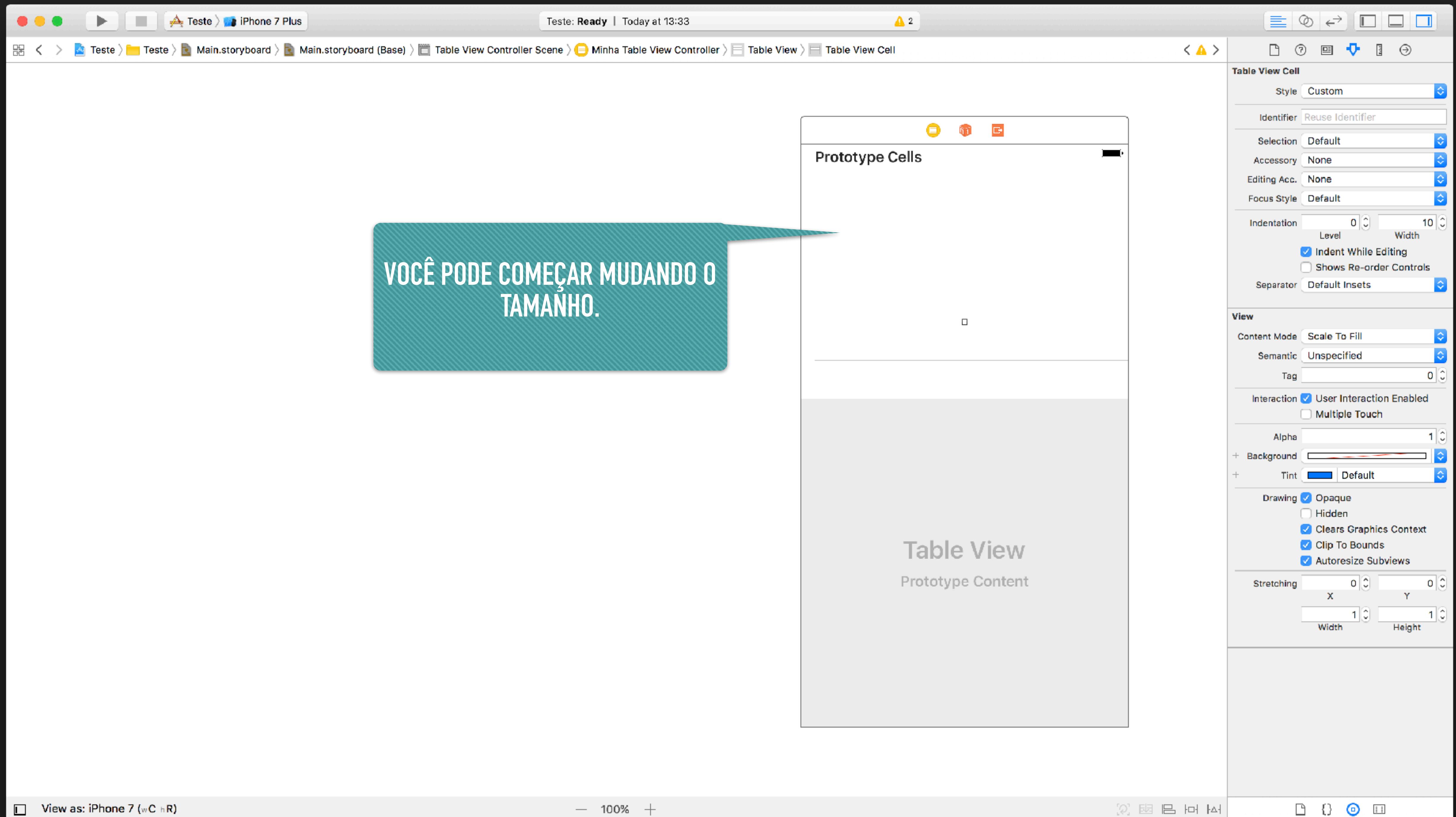
Autoresizes Subviews

Stretching: 0 X: 1 Y: 1

Width: 1 Height: 1

DA MESMA FORMA QUE NA TABELA DE DADOS ESTÁTICOS, AS CÉLULAS DO TIPO CUSTOM PRECISAM TER A INTERFACE CONSTRUÍDA INTERNAMENTE.

A INTERFACE DESTAS CÉLULAS VAI SER REPLICADA PARA CADA REGISTRO EXIBIDO. ISTO OCORRE PORQUE NA TABELA DINÂMICA ESTAS CÉLULAS SÃO TEMPLATES.



Teste Ready | Today at 13:39

2

Teste > Teste > Main.storyboard > Main.storyboard (Base) > Table View Controller Scene > Minha Table View Controller > Table View > Table View Cell

Table View Cell

Style: Custom

Identifier: Reuse Identifier

Selection: Default

Accessory: None

Editing Acc.: None

Focus Style: Default

Indentation: 0 Level 10 Width

Indent While Editing

Shows Re-order Controls

Separator: Default Insets

View

Content Mode: Scale To Fill

Semantic: Unspecified

Tag: 0

Interaction:  User Interaction Enabled

Multiple Touch

Alpha: 1

+ Background: Red

+ Tint: Default

Drawing:  Opaque

Hidden

Clears Graphics Context

Clip To Bounds

Autoresizes Subviews

Stretching: X 0 Y 0

Width 1 Height 1

Label Label - A variably sized amount of static text.

DEPOIS VOCÊ VAI COLOCANDO OS COMPONENTES PARA CRIAR SUA INTERFACE.

DICA DO ❤️: É IMPORTANTE COLOCAR CORRETAMENTE AS CONSTRAINTS DO AUTOLAYOUT SE VOCÊ QUISER QUE AS CÉLULAS TENHAM AJUSTE AUTOMÁTICO DE ALTURA PARA ACOMODAR O CONTEÚDO QUE SERÁ EXIBIDO.

View as: iPhone 7 (wC hR)

— 100% +

VOCÊ NÃO PODE LIGAR OS OUTLETS DESTA CÉLULA TEMPLATE DIRETAMENTE À SUA SUBCLASSE DE UITABLEVIEWCONTROLLER.

POR QUE NÃO? POR QUE SIMPLEMENTE PODERIA EXISTIR 1000 CÓPIAS DESTES OUTLETS!

ENTÃO, AO INVÉS DISSO, VAMOS LIGAR OS OUTLETS À UMA SUBCLASSE DE UIVIEW QUE SABERÁ FAZER TUDO FUNCIONAR ADEQUADAMENTE.

ESTA SUBCLASSE DE UIVIEW É DE UM TIPO ESPECIAL. A CLASSE PAI DA SUA CLASSE DEVE SER A UITABLEVIEWCELL, QUE É A CLASSE QUE DESCREVE UMA CÉLULA DA TABELA.

ASSIM COMO ACONTECE COM O CONTROLLER, VOCÊ VAI PRECISAR CRIAR UMA SUBCLASSE DE UITABLEVIEWCELL PARA PODER TER OS OUTLETS CONFIGURADOS E FUNCIONANDO.

The screenshot shows the Xcode interface with a storyboard scene selected. The navigation bar at the top displays the path: Teste > Main.storyboard > Main.storyboard (Base) > Table View Controller Scene > Minha Table View Controller > Table View > Table View Cell. The main canvas shows a prototype cell with a UIImageView labeled 'Foto' and a Label below it with placeholder text. The right side of the screen features the Attributes Inspector for the 'Table View Cell' object, which includes sections for Style (Custom), Identifier (Reuse Identifier), Selection (Default), Accessory (None), Editing Acc. (None), Focus Style (Default), View (Content Mode: Scale To Fill, Semantic: Unspecified, Tag: 0), Interaction (User Interaction Enabled checked, Multiple Touch unchecked), Drawing (Opaque checked, Hidden unchecked, Clears Graphics Context checked, Clip To Bounds checked, Autoresize Subviews checked), Stretching (X: 0, Y: 0, Width: 1, Height: 1), and Label (Label - A variably sized amount of static text).

Teste: Ready | Today at 20:38

Custom Class  
Class: UITableViewCell  
Module: None  
Inherit From Target

Identity  
Restoration ID:

User-Defined Runtime Attributes  
Key: Path | Type | Value

Document  
Label: Xcode Specific Label  
Object ID: 5x0-5x-U1W  
Lock: Inherited - (Nothing)  
Notes: No Font  
Comment For Localizer

Accessibility  
Accessibility: Enabled  
Label: Label  
Hint: Hint  
Identifier: Identifier  
Traits: Button, Image, Static Text, Search Field, Plays Sound

Table View  
Prototype Content

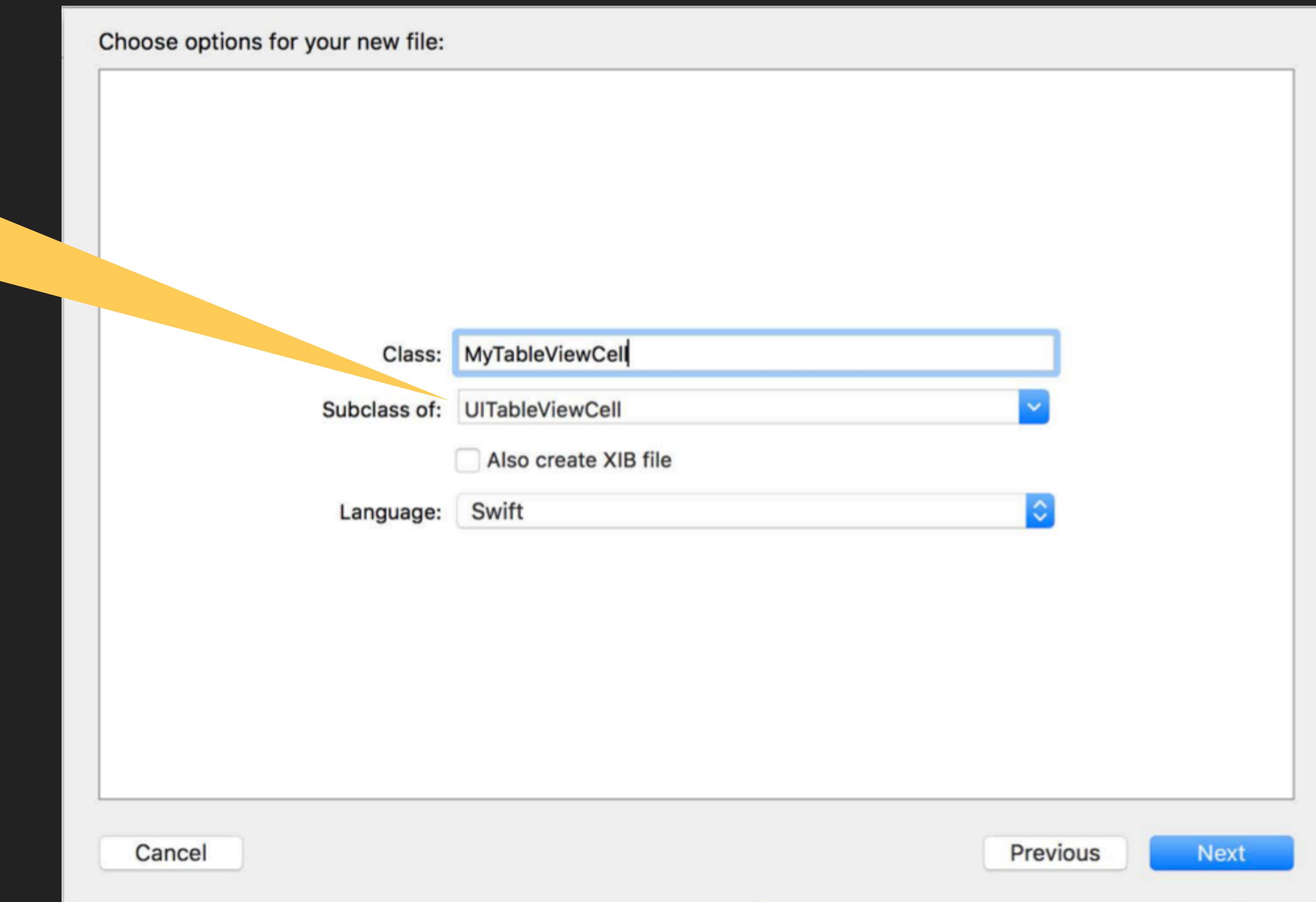
Label: Label - A variably sized amount of static text.

View as: iPhone 7 (wC hR) 100% +

ENTÃO, DA MESMA FORMA QUE VOCÊ FAZ COM O CONTROLLER, SELECIONE A CÉLULA ...

E ENTÃO USE O INSPETOR DE IDENTIDADE PARA DEFINIR QUAL É A SUBCLASSE.

**PARA ISSO, VOCÊ PRECISA CRIAR A SUA  
SUBCLASSE, QUE HERA DE UITABLEVIEWCELL.**



Teste: Ready | Today at 20:43

2

Teste < > Main.storyboard > Main.storyboard (Base) > Minha Table View Controller Scene > Minha Table View Controller > Table View > My TableView Cell

Custom Class  
Class MyTableViewCell  
Module Teste  
✓ Inherit From Target

Identity  
Restoration ID

User Defined Runtime Attributes  
Key Path Type Value

Document  
Label Xcode Specific Label  
Object ID 5x0-5x-U1W  
Lock Inherited - (Nothing)  
Notes  
No Font  
Comment For Localizer

Accessibility  
Accessibility  Enabled  
Label Label  
Hint Hint  
Identifier Identifier  
Traits  Button  Link  
 Image  Selected  
 Static Text  
 Search Field  
 Plays Sound

Label Label - A variably sized amount of static text.

E DEPOIS, É SÓ APONTAR PARA A SUBCLASSE CORRETA, COMO VOCÊ JÁ ESTÁ ACOSTUMADO.

Prototype Cells

UIImageView Foto

Título

Mussum Ipsum, cacilds vidis litro abertis. In elementis mé pra quem é amistosis quis leo. Quem num gosta di mé, boa gentis num é. Detraxit consequat et quo num tendi nada....

Table View

Prototype Content

View as: iPhone 7 (wC hR)

— 100% +

label

Teste Ready | Today at 20:43

Custom Class  
Class MyTableViewCell  
Module Teste  
✓ Inherit From Target

Identity  
Restoration ID

User Defined Runtime Attributes  
Key Path Type Value

Document  
Label Xcode Specific Label  
Object ID 5x0-5x-U1W  
Lock Inherited - (Nothing)  
Notes  
No Font  
Comment For Localizer

Accessibility  
Accessibility  Enabled  
Label Label  
Hint Hint  
Identifier Identifier

Traits  Button  Link  
 Image  Selected  
 Static Text  
 Search Field  
 Plays Sound

Table View  
Prototype Content

Custom Class  
Class MyTableViewCell  
Module Teste  
✓ Inherit From Target

Identity  
Restoration ID

User Defined Runtime Attributes  
Key Path Type Value

Document  
Label Xcode Specific Label  
Object ID 5x0-5x-U1W  
Lock Inherited - (Nothing)  
Notes  
No Font  
Comment For Localizer

Accessibility  
Accessibility  Enabled  
Label Label  
Hint Hint  
Identifier Identifier

Traits  Button  Link  
 Image  Selected  
 Static Text  
 Search Field  
 Plays Sound

Label Label - A variably sized amount of static text.

View as: iPhone 7 (wC hR) 100% +

AGORA VOCÊ PODE LIGAR SEUS OUTLETS E ACTIONS!

DICA DO ❤️: ELEMENTOS DE UMA CÉLULA ESTÁTICA SÃO LIGADOS DIRETAMENTE NO CONTROLLER. ELEMENTOS DE UMA CÉLULA DINÂMICA, NO ENTANTO, SÃO LIGADOS À SUBCLASSE DE UITABLEVIEWCELL QUE OS CONTÉM.

The screenshot shows the Xcode interface with two main panes. The left pane displays a storyboard titled 'Teste' for an 'iPhone 7 Plus'. It contains a 'Table View' with a single 'Prototype Content' section. Inside the table view, there is a 'My TableView Cell' with a 'Content View'. The content view contains a 'UIImageView' labeled 'Foto' and a 'UILabel' labeled 'Título'. The 'Foto' label has a tooltip 'UImageview'. Below the table view, the status bar indicates 'View as: iPhone 7 (wC hR) 100%'.

The right pane shows the 'MyTableViewCell.swift' file. The code is as follows:

```
1 //  
2 // MyTableViewCell.swift  
3 // Teste  
4 //  
5 // Created by Pedro Henrique on 09/08/17.  
6 // Copyright © 2017 IESB. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class MyTableViewCell: UITableViewCell {  
12  
13  
14 }  
15  
16
```

A large red callout bubble is overlaid on the right side of the screen, containing the text:

AGORA SIM ESTAMOS PRONTOS PARA FAZER AS LIGAÇÕES DE OUTLETS E ACTIONS!

The screenshot shows the Xcode interface with two main panes. The left pane displays a storyboard for an iPhone 7 Plus device. It features a 'Table View' containing a single 'Prototype Content' section. Within this section, there is a 'Photo Image View' and a 'Text View' labeled 'Foto'. The right pane shows the corresponding Swift code for a file named 'MyTableViewCell.swift'. The code defines a class 'MyTableViewCell' that inherits from 'UITableViewCell'. It includes an outlet 'photoImageView' of type 'UIImageView', a private variable 'infoShownByThisCell' of type 'Type' with a didSet observer, and a private function 'updateUI()' that contains a comment about updating the cell with information.

```
// MyTableViewCell.swift
// Teste
// Created by Pedro Henrique on 09/08/17.
// Copyright © 2017 IESB. All rights reserved.

import UIKit

class MyTableViewCell: UITableViewCell {

    @IBOutlet weak var photoImageView: UIImageView!

    var infoShownByThisCell: Type {
        didSet {
            updateUI()
        }
    }

    private func updateUI() {
        //Atualizar a célula com as informações!
    }
}
```

OBSERVAR QUE ESTE OUTLET NÃO ESTÁ EM UM CONTROLLER! ELE ESTÁ NA CLASSE CORRESPONDENTE AO TEMPLATE DE CÉLULA, DE SORTE QUE CADA LINHA DA TABELA VAI TER A SUA PRÓPRIA PHOTOIMAGEVIEW.

A CÉLULA PRECISA FORNECER ALGUMA API PÚBLICA PARA QUE ELA POSSA OBTER A INFORMAÇÃO A SER EXIBIDA NOS OUTLETS.

VAMOS DESCOBRIR DAQUI A POUCO COMO PREENCHER ESTA VARIÁVEL.

## OS PROTOCOLOS DA UITABLEVIEW... COMO LIGAR AS COISAS?

- ▶ As conexões ao código são feitas usando os protocolos da UITableView chamados **dataSource** e **delegate**;
- ▶ O **delegate** é usado para controlar como a tabela vai ser exibida (o look and feel da tabela);
- ▶ Já o **dataSource** é responsável por prover os dados que são exibidos nas linhas da tabela - dentro das células;
- ▶ O **UITableViewController** vai configurar automaticamente as ligações de **delegate** e **dataSource**;
- ▶ Sua subclasse de **UITableViewController** também tem uma propriedade que aponta para a UITableView...

**var tableView: UITableView**

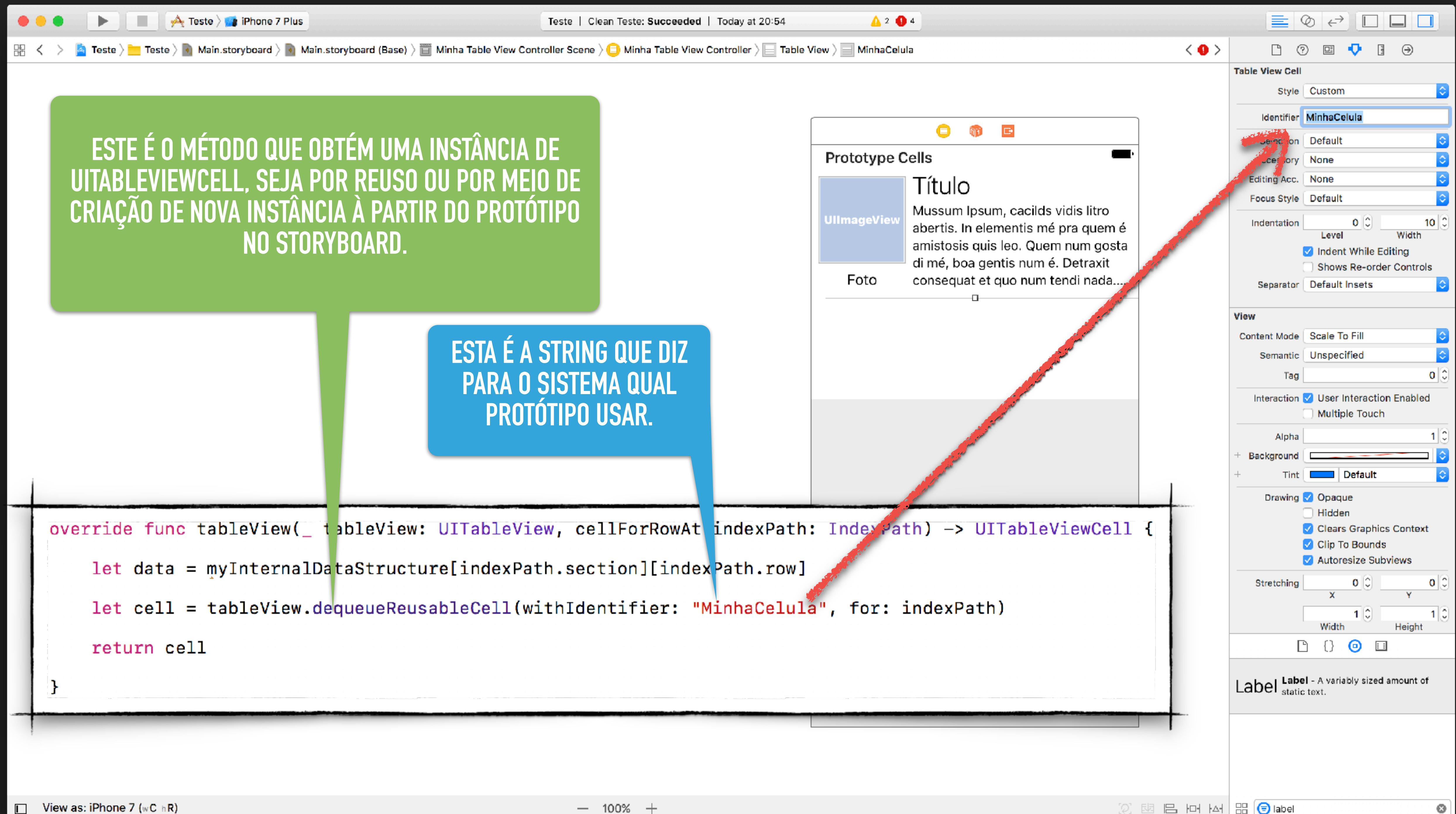
## QUANDO VOU PRECISAR IMPLEMENTAR O DATASOURCE???

- ▶ Quando os dados a exibir na tabela forem dinâmicos (as células não são estáticas);
- ▶ Existem três métodos muito importantes neste protocolo que fazem perguntas fundamentais para o funcionamento da tabela:  
**Quantas seções existem na tabela?**  
**Quantas linhas existem em cada seção?**  
**Qual é a célula que eu tenho que desenhar para uma dada posição (seção e linha)?**
- ▶ Vamos falar da última pergunta primeiro, já que as duas primeiras são bem triviais...

## 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);
- ▶ Não se preocupe! Se você tiver 10.000 registros, apenas os 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 data = myInternalDataStructure[indexPath.section][indexPath.row]  
    let cell = //criar uma instância de UITableViewcell e preencher com 'data'  
    return cell  
}
```



## UITABLEVIEWDELEGATE

- ▶ Até agora só falamos do dataSource, mas o UITableView tem um outro protocolo importante: o **UITableViewDelegate**;
- ▶ O delegate controle como a tabela é exibida. Não os dados que ela exibe;
- ▶ É normal que o dataSource e o delegate sejam o mesmo objeto. É usual que o ViewController do MVC em questão assuma esses papéis;
- ▶ O delegate também de permite observar o que a tabela anda fazendo. Especialmente, o delegate te permite reagir quando o usuário "clica" numa determinada linha da tabela. Você pode simplesmente fazer uma **segue**, ou você pode implementar o código que quiser para fazer magia.

## “ACTIONS” COM O UITABLEVIEW

- ▶ Ao implementar o protocolo delegate, você será notificado quando uma linha for selecionada pelo usuário. Isso funciona, mais ou menos, como o clique de um botão, sendo necessário apenas quando você não for fazer uma segue diretamente;
- ▶ O delegate ainda é capaz de dizer quando o **Detail Disclosure Button** for clicado. Novamente, você só precisa deste método do delegasse quando não for fazer apenas uma segue diretamente.

## MUITO... MUITO MAIS MÉTODOS NO DELEGASSE

- ▶ O delegate é um cara X9!
- ▶ Ele oferece métodos **will/did** para quase tudo que acontece no Table View;
- ▶ Ele te pergunta por UIViews para desenhar o cabeçalho e o rodapé da tabela;
- ▶ Ele te dá mecanismos para lidar com a edição de registros exibidos na tabela;
- ▶ Ele oferece ainda meios para lidar com a área de transferência (copiar e colar registros);
- ▶ Então... não deixe de conferir a documentação para saber tudo que é possível fazer.

## E SE A MINHA MODEL MUDAR?

- ▶ O método **reloadData()** funciona como o martelo do Thor! Ele apaga tudo que está na tabela e carrega tudo novamente, chamando todos os métodos envolvidos do **dataSource**;
- ▶ Como é de se imaginar, este método é relativamente pesado, mas se a sua estrutura de dados mudar completamente, é dele que você vai precisar;
- ▶ Mas, por outro lado, se apenas um pedaço do dado mudar, existe uma função mais leve para atualizar só o que interessa:  
**reloadRows(at indexPaths: [IndexPath], with animation: UITableViewRowAnimation)**

## CONTROLANDO A ALTURA DAS CÉLULAS

- ▶ A altura das células pode ser fixa (storyboard, via propriedade **rowHeight**);
- ▶ Ou, ela pode ser determinada usando o autolayout (setar a propriedade **rowHeight = UITableViewAutomaticDimension**);
- ▶ Caso você vá para o lado automático da força, você pode dar uma ajudinha para a tableView através da propriedade **estimatedRowHeight**. Quando esta propriedade está preenchida, a tabela só executa o autolayout das linhas que estão visíveis e aplica o valor estimado nas demais;

## DOMINANDO A TABLEVIEW

- ▶ Existem várias dezenas de métodos e outras coisas interessantes sobre a tableView;
- ▶ Coisas para lidar com cabeçalhos, rodapés, controle refinado de aparência (estilo, cor e altura do separador, por exemplo), controle do scroll;
- ▶ Ainda temos meios para gerenciar as seções, lidar com seleção única e múltipla de registros, mover, inserir e excluir linhas;
- ▶ E, mais uma vez, não deixe de conferir da documentação para saber tudo mais que existe de incrível sobre este que é o componente mais usado do iOS, desde que surgiu.



# K-BÔ!!!

E hoje tem exercício para casa!  
Descrição e prazo no Blackboard.