

TableView

X-Code com Swift
Prof. Agesandro Scarpioni
agesandro@fiap.com.br

Table View

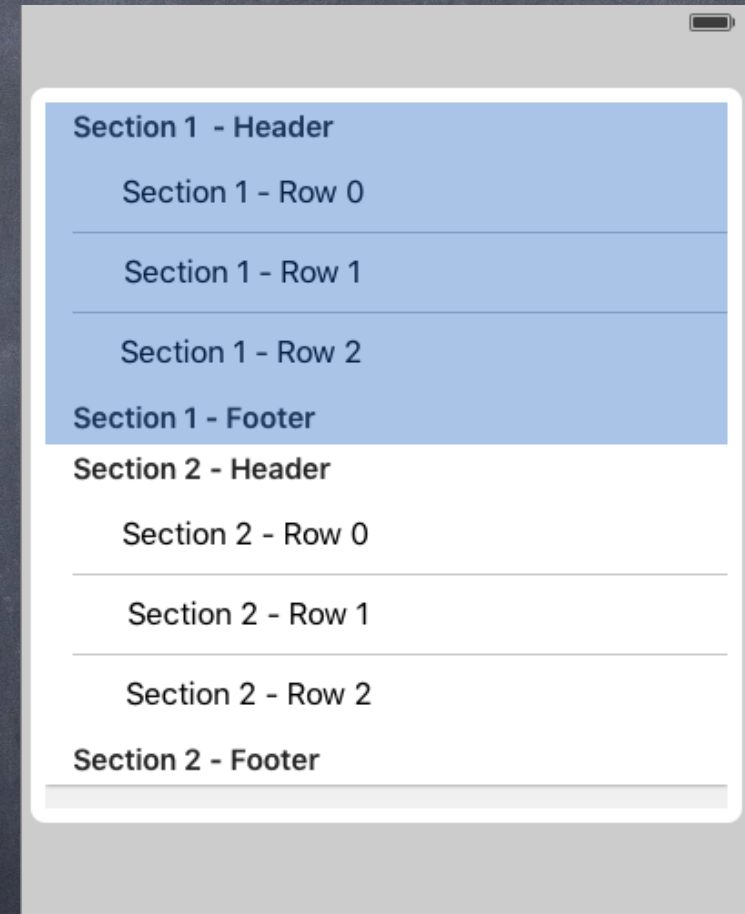
- Veja como é fácil utilizar o TableView um objeto muito utilizado para exibir e manipular listas.

Table View

- As tabelas exibem listas de informação, são representadas por UITableView e controladas por subclasses de UITableViewController.
- As tabelas podem ter um número ilimitado de linhas, no entanto só podem ter uma coluna.
- A Table View adota os protocolos UITableViewDataSource (possui assinaturas para popular a tabela) e UITableViewDelegate (possui assinaturas para detectar a interação com as células e controles visuais), um UITableViewController já possui esses protocolos pré adotados, faremos uso de UITableViewController no próximo conjunto de slides.
- Cada item de uma tabela é uma instância de UITableViewCell, elas herdam de UIView e podem ter qualquer tipo de componente, sendo assim, toda célula pode ser configurada conforme a necessidade do desenvolvedor, por exemplo: é possível colocar um Button ou um UIImageView dentro de uma célula.

Table View

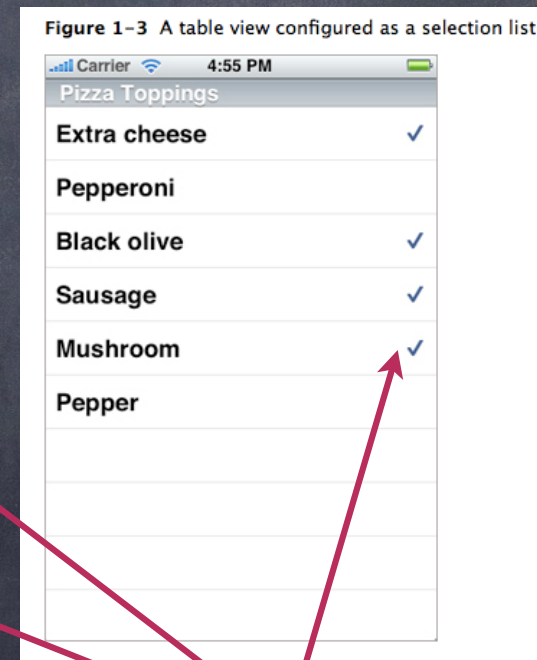
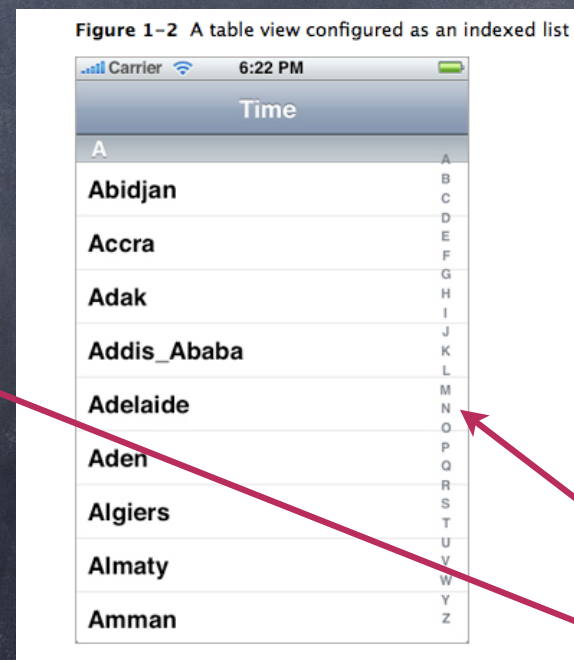
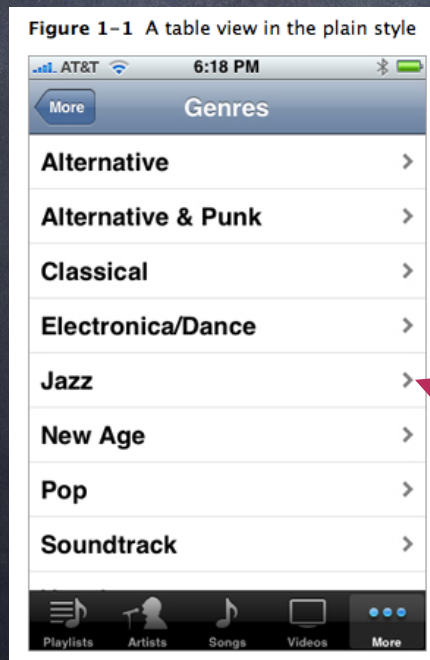
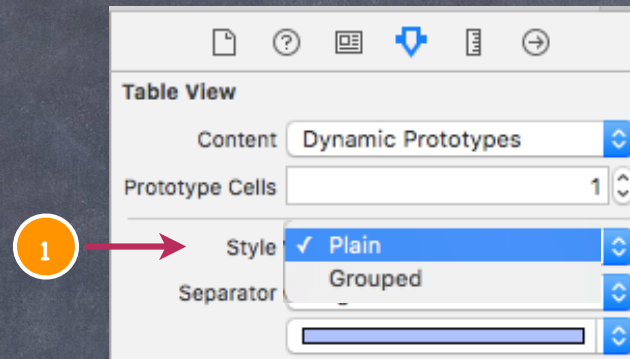
- As tabelas possuem os seguintes elementos: Section, header, footer e row.
- As seções são grupos de linhas de uma tabela.
- O header é o cabeçalho de cada seção, por exemplo: imagine que as seções contenham linhas com nomes de jogos separadas por tipo de consoles, no header será exibido o nome do console e em row será exibido os nomes dos jogos.
- Row é formado por objetos do tipo UITableViewCell, ou seja, células.
- O footer é o rodapé de cada seção, no exemplo acima é possível exibir o total de jogos desse tipo de console



Section 1 - Header
Section 1 - Row 0
Section 1 - Row 1
Section 1 - Row 2
Section 1 - Footer
Section 2 - Header
Section 2 - Row 0
Section 2 - Row 1
Section 2 - Row 2
Section 2 - Footer

Table View (Tabelas) – Estilos

- As tabelas podem ter 2 estilos Plain (simples) (1) ou Grouped (agrupada)



OBS: Acima alguns modelos de tabelas simples com uso de accessory

Table View (Tabelas) – Estilos

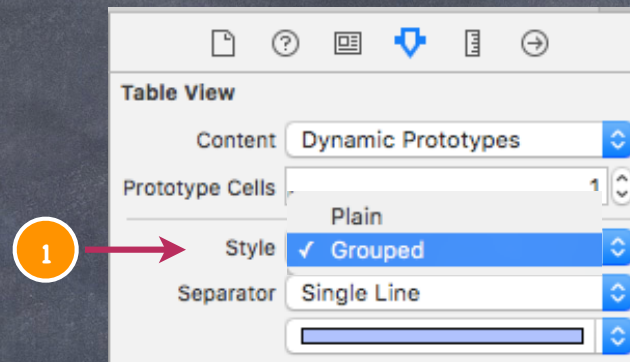
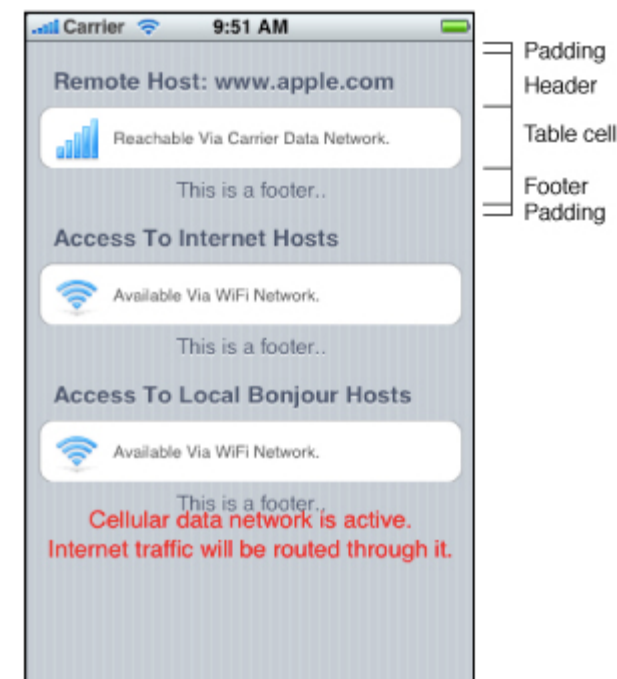


Figure 1-4 A table view in the grouped style



Figure 1-5 Header and footer of a section



OBS: Acima alguns modelos de tabelas agrupadas

UITableViewDataSource

- O protocolo UITableViewDataSource fornece métodos para popular a tabela, os mais importantes são:

```
16 //retorna número de seções da tabela - método obrigatório
17 func numberOfSectionsInTableView(tableView: UITableView) -> Int {
18     code
19 }
20
21 //retorna o número de linhas por seção - método obrigatório
22 func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
23     code
24 }
25
26 //retorna uma célula da tabela para o índice informado - método obrigatório
27 func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
28     code
29 }
30
31 //Título do cabeçalho de cada seção
32 func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
33     code
34 }
35
36 //Título do rodapé de cada seção
37 func tableView(tableView: UITableView, titleForFooterInSection section: Int) -> String? {
38     code
39 }
40
41 //Edição da tabela pelo usuário, delete ou insert de linhas na tabela
42 func tableView(tableView: UITableView, commitEditingStyle editingStyle: UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath) {
43     code
44 }
```

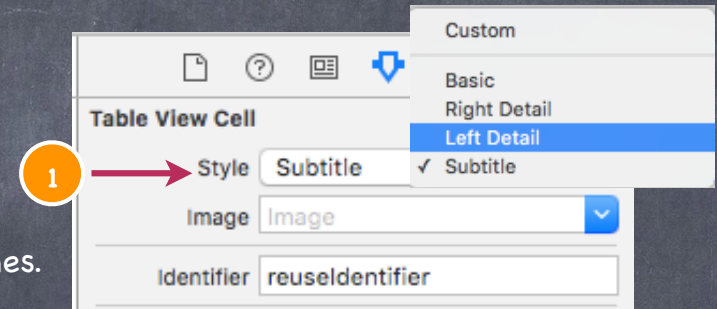

UITableViewDelegate

- O protocolo UITableViewDelegate fornece métodos para controles visuais e interação com a tabela, como por exemplo saber se ocorreu a seleção de uma linha ou alterar a altura de um cabeçalho.

```
55 //seleção de um item na tabela
56 func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
57     code
58 }
59 //seleção do botão de acessório
60 func tableView(tableView: UITableView, accessoryButtonTappedForRowWithIndexPath indexPath: NSIndexPath) {
61     code
62 }
63 //alteração de altura (height) de linha (row), cabeçalho (header) e rodapé (footer).
64 func tableView(tableView: UITableView, heightForRowAtIndexPath indexPath: NSIndexPath) -> CGFloat {
65     code
66 }
67 func tableView(tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
68     code
69 }
70 func tableView(tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {
71     code
72 }
73 //customização da view de cabeçalho e de rodapé
74 func tableView(tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
75     code
76 }
77 func tableView(tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {
78     code
79 }
80 }
```


Table View Cell (Células) – Estilos

- Possuem 4 estilos de apresentação do conteúdo (1).



- Basic (1-6) – Texto principal alinhado à esquerda sem texto de detalhes.
- Subtitle (1-7) – Texto principal em fonte maior alinhado acima do texto de detalhe com fonte menor.
- Right Detail (1-8) – Texto principal alinhado à esquerda com detalhes alinhado à direita.
- Left Detail (1-9) – Texto principal alinhado à esquerda com detalhes alinhado à esquerda.

Figure 1-6 Default table row style

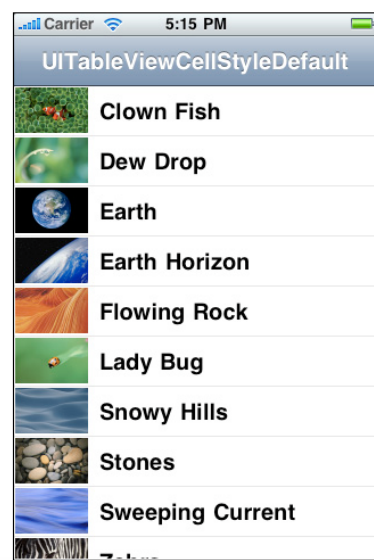


Figure 1-7 Table row style with a subtitle under the title

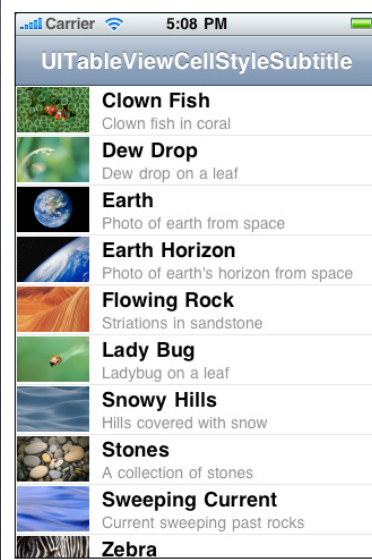


Figure 1-8 Table row style with a right-aligned subtitle

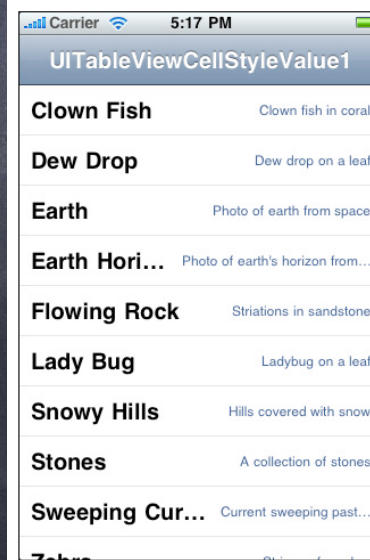
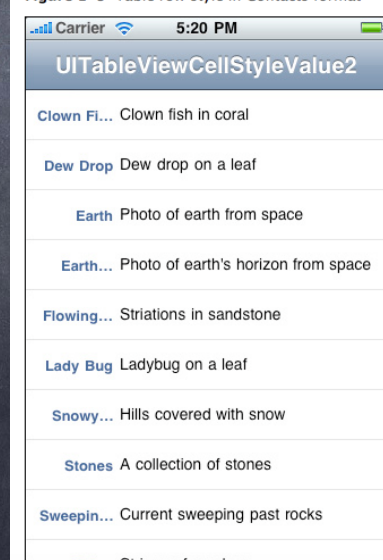
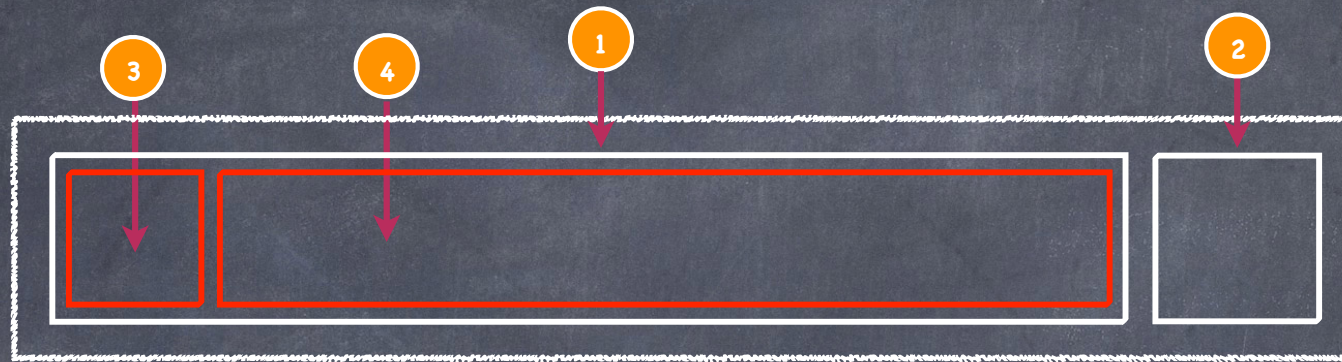


Figure 1-9 Table row style in Contacts format



OBS: É necessário chamar a propriedade `.detailTextLabel` para alterar o texto de detalhes, veja a página 11

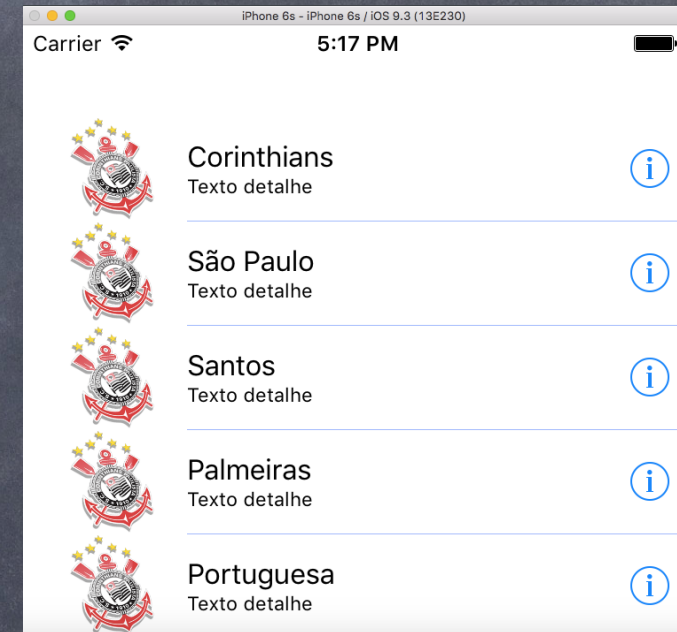
Table View Cell



- A célula (UITableViewCell) possui os seguintes elementos:
 - Cell content (1).
 - Accessory view (2).
 - Image (3).
 - Text (4).

Table View Cell

- As principais propriedades da célula são:
 - `backgroundColor` para alterar a cor de fundo da célula.
 - `imageView` para alterar a imagem da célula.
 - `textLabel` para alterar o conteúdo principal da célula.
 - `accessoryType` para definir qual é o tipo de acessório da célula.



```
31     celula.textLabel?.text = "\(time[indexPath.row])"
32     celula.imageView?.image = UIImage(named: "corinthians.png")
33     celula.detailTextLabel?.text = "Texto detalhe"
34     celula.accessoryType = .DetailButton
```

OBS: Observe que os nomes dos times tem como origem um Array, já a imagem é a mesma, pois vem de uma única fonte.

Table View Cell

- A célula (UITableViewCell) possui os seguintes acessórios (UITableViewCellAccessoryType) : None, DisclosureIndicator, DetailDisclosureButton, Checkmark e DetailButton.
- None sem acessório.
- O DisclosureIndicator (1), indicador padrão de navegação sem ação.
- DetailDisclosureButton (2), botão de informação com indicador de navegação com ação.
- Checkmark (3) indicador com estilo checked sem ação.
- DetailButton (4), botão de informação com ação (disponível a partir do iOS 7).

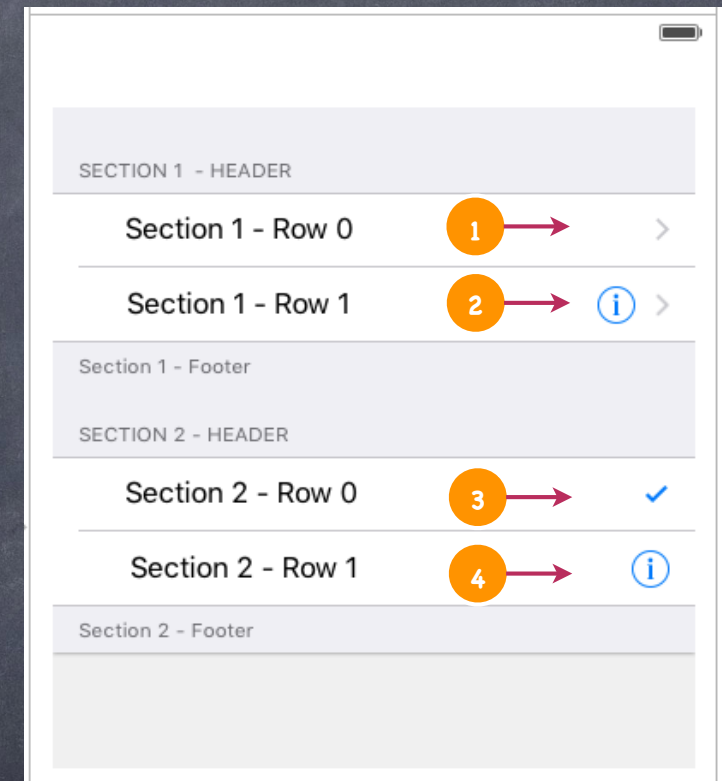


Table View

- Vamos criar um projeto novo do tipo iOS application (Single View Application), clique em Next.

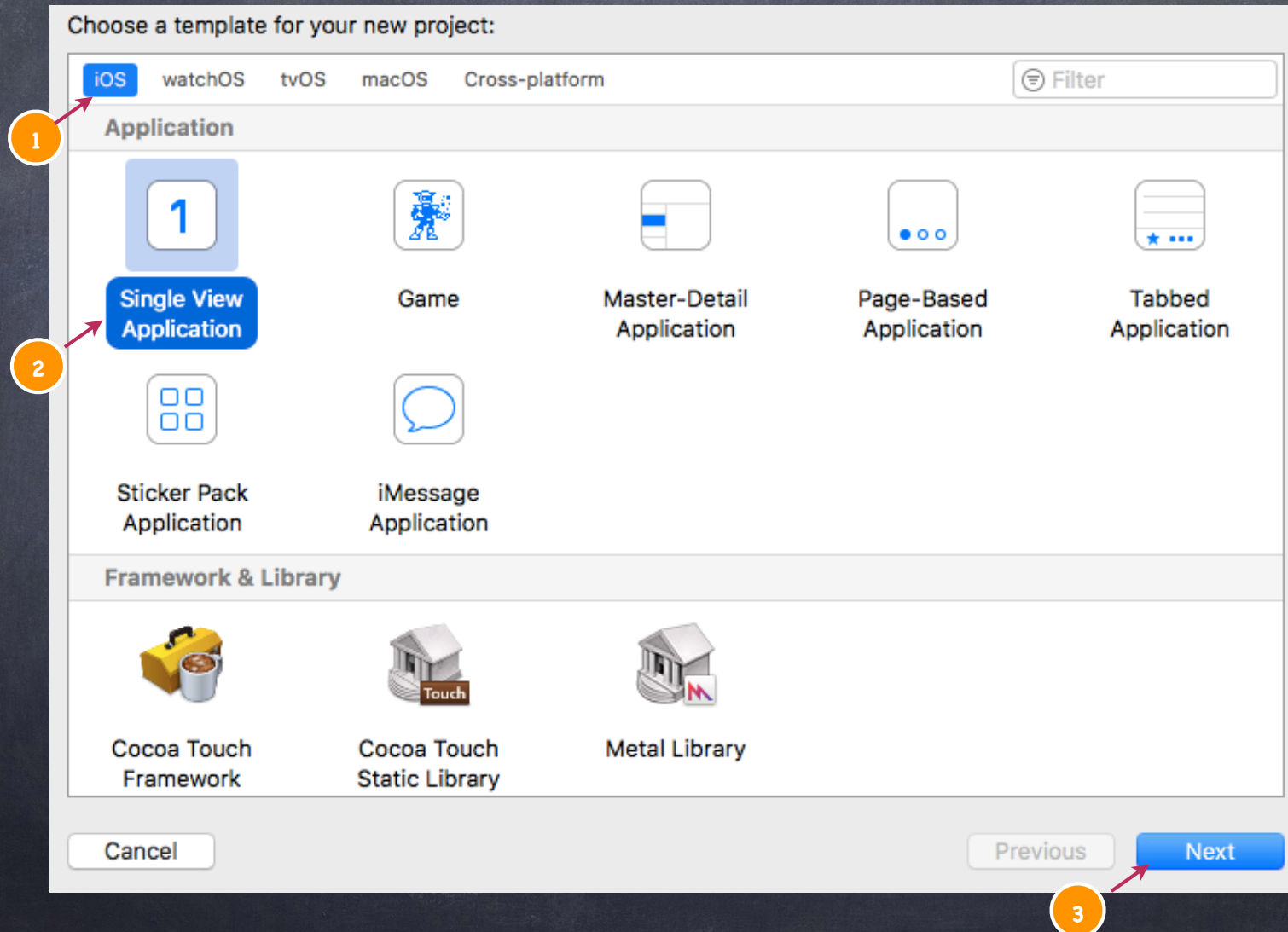


Table View

- Nomeie o projeto como: "Exemplo1_TableView" (4), escolha em language: Swift (5) e em device: iPhone (6).

Choose options for your new project:

Product Name:

Team:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Devices:

☐ Use Core Data
☐ Include Unit Tests
☐ Include UI Tests

Table View

- Arraste para sua View Controller um objeto Table View(2), depois clique nos botões 3 e 4 para deixarmos abertas simultaneamente as telas de interface e arquivo.swift.

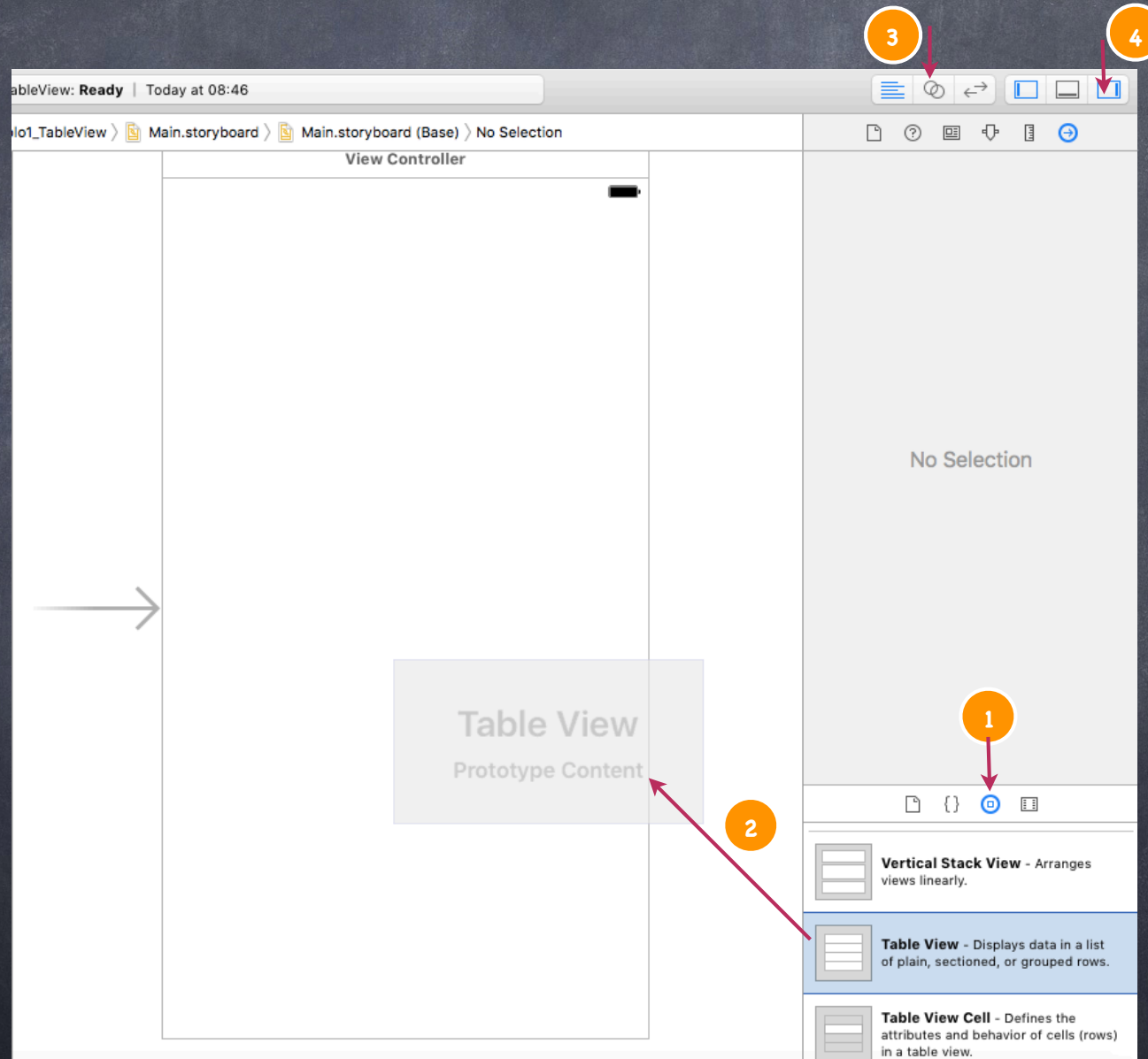


Table View

- Vamos criar um Outlet da TableView clicando com o botão direito sobre o objeto, escolha New Referencing Outlet(1) e arraste até a área indicada, nomeie seu Outlet como minhaTableView(2).

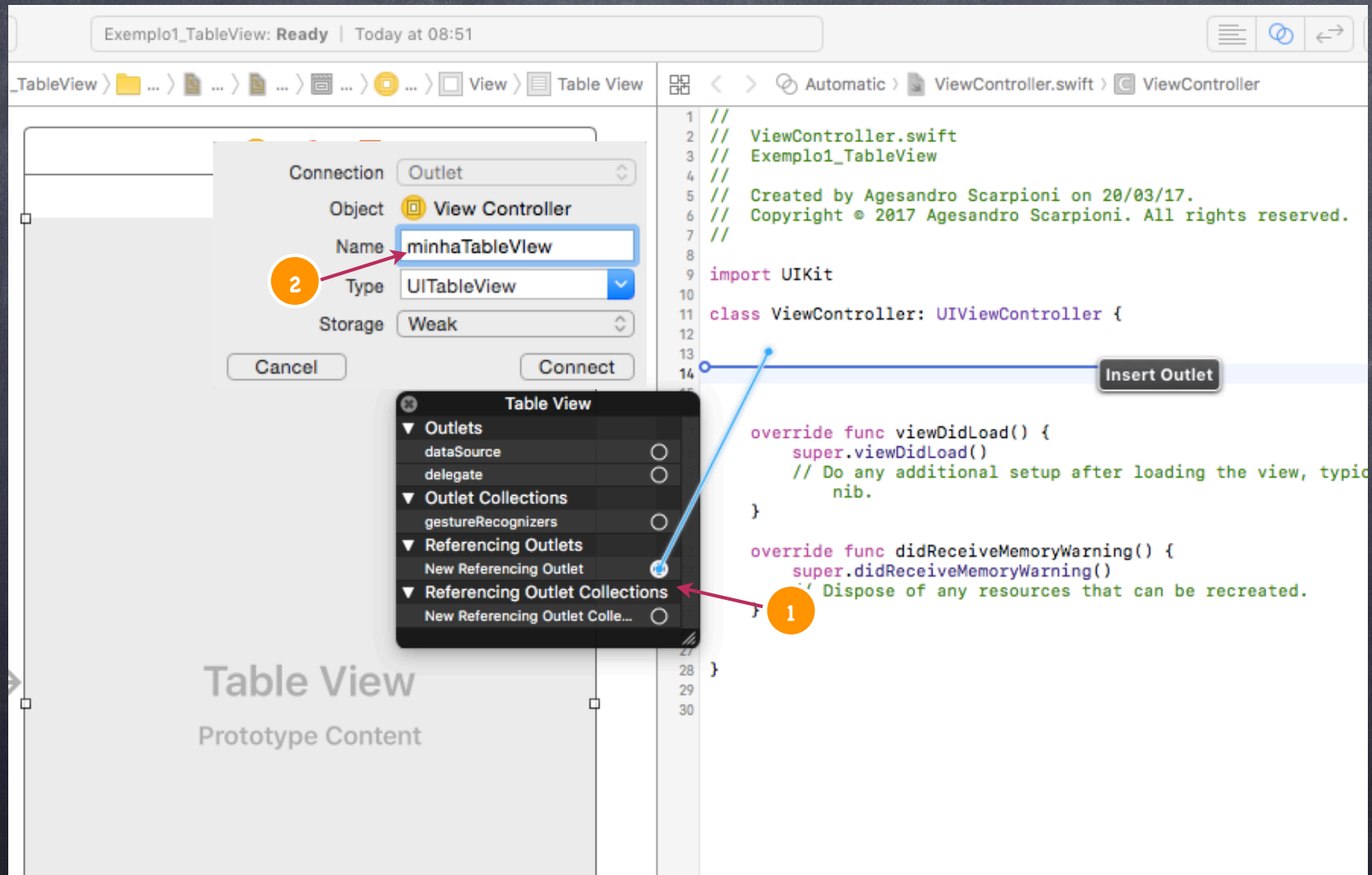


Table View

- No arquivo ViewController.swift, vamos informar os dois protocolos da TableView que serão implementados, UITableViewDataSource e UITableViewDelegate, o primeiro possui assinaturas que cuidam dos dados que serão apresentados, o segundo será responsável por assinaturas de controle visual e de interação com as células, como por exemplo métodos que checam quando ocorre um "tap" em uma linha na TableView.

```
1 //
2 // ViewController.swift
3 // Exemplo1_TableView
4 //
5 // Created by Agesandro Scarpioni on 20/03/17.
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
12
13
14     @IBOutlet weak var minhaTableView: UITableView!
15
16
17
18     override func viewDidLoad() {
19         super.viewDidLoad()
20         // Do any additional setup after loading the view, typically from a nib.
21     }
22
23     override func didReceiveMemoryWarning() {
24         super.didReceiveMemoryWarning()
25         // Dispose of any resources that can be recreated.
26     }
27
28
29 }
30
31
```

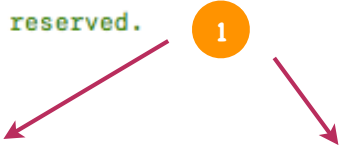


Table View

- Com o **Table View selecionado** (1) vá no connections inspector (2), ligue o protocolo datasource(2) do tableView ao View controller(4), desta forma não precisamos fazer a ligação via linha de código.

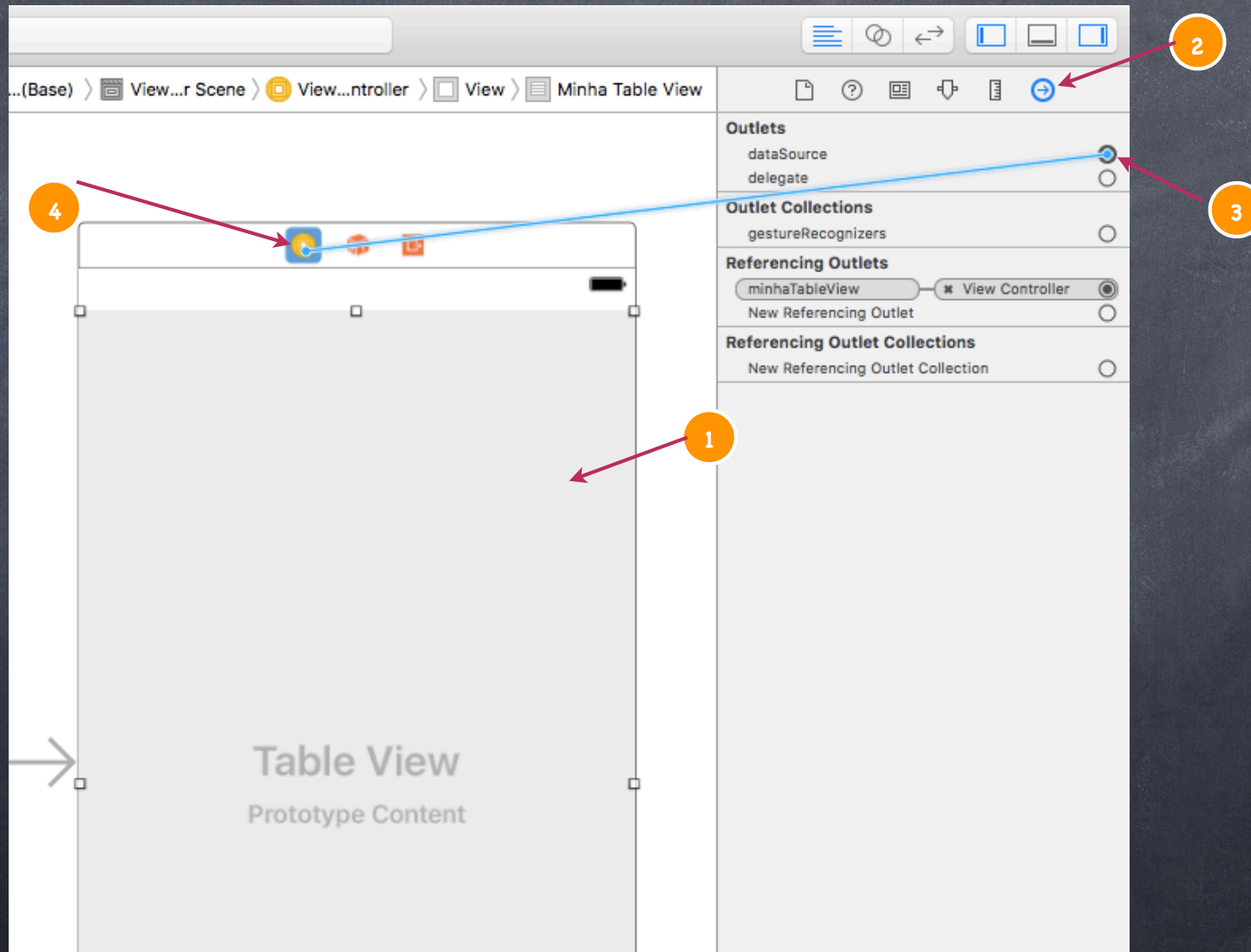
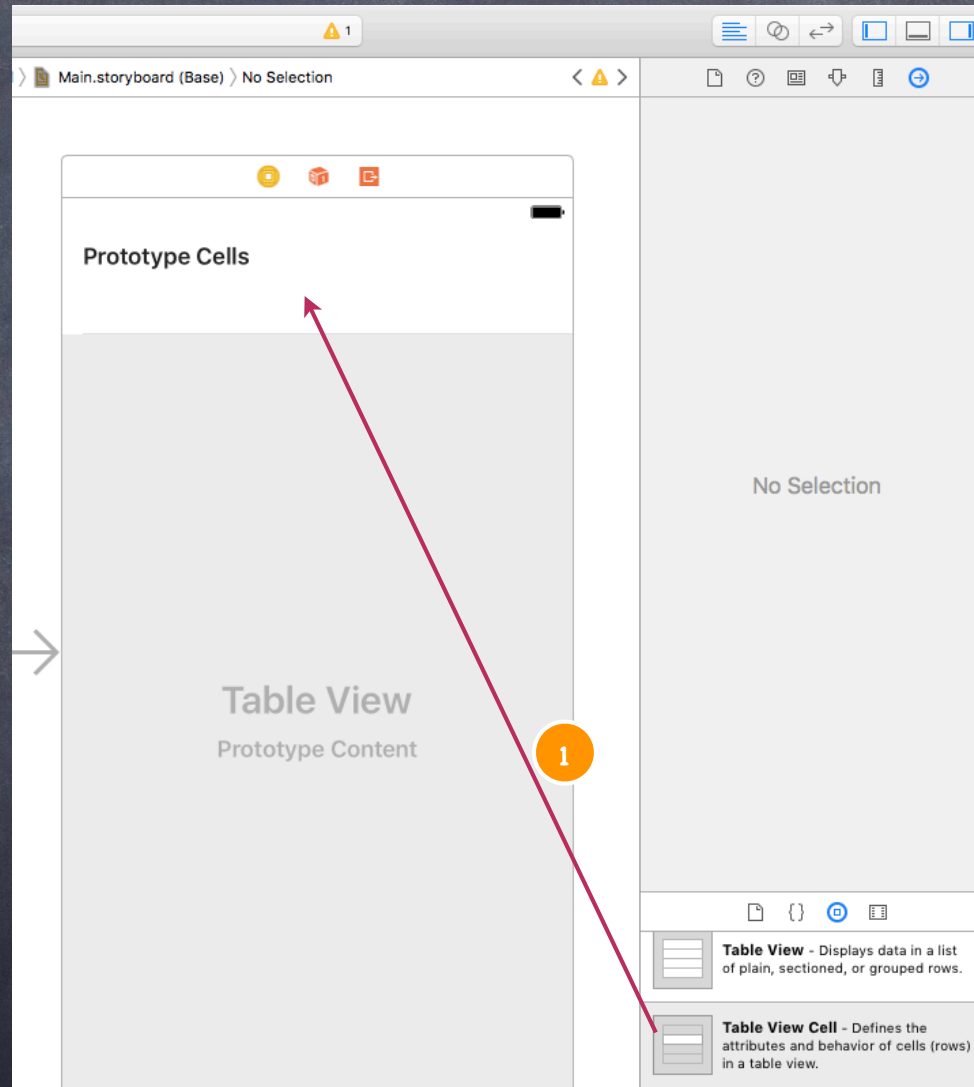


Table View

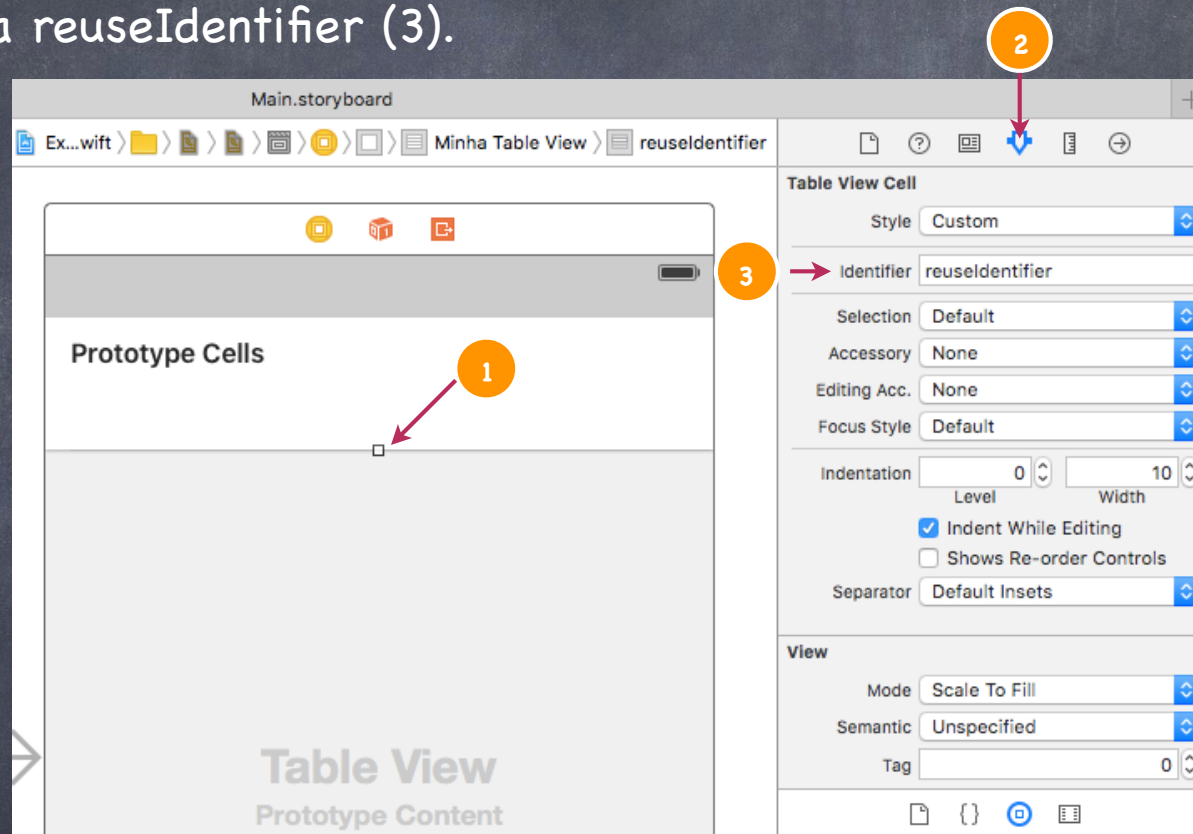
- Adicione um Table View Cell em seu projeto.



OBS: O Table View Cell deve ser colocado no topo, porém, dentro do Table View.

Table View

- Selecione a Table View Cell (1) e em attributes inspector (2) altere o atributo Identifier para reuseIdentifier (3).



- Para melhorar a performance do app, as células da tabela são reutilizadas e por isso precisam de um identificador de reuso. O identificador deve ser usado para recuperar células no momento da configuração (ver código na página 12) e deve ter o mesmo nome que digitamos aqui no tem (3).

Table View

- Implemente dois protocolos de TableView, implemente inicialmente apenas os métodos do protocolo UITableViewDataSource(4) no arquivo .swift, lembre-se, esse primeiro protocolo(1) vai cuidar dos dados que serão apresentados. Veja os comentários de cada método (2,3,4).

```
1 //
2 // ViewController.swift
3 // Exemplo1_TableView
4 //
5 // Created by Agesandro Scarpioni on 20/03/17.
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
12
13
14     @IBOutlet weak var minhaTableView: UITableView!
15
16     //retorna quantas seções terá nossa lista - Método obrigatório
17     func numberOfSections(in tableView: UITableView) -> Int { ← 2
18         code
19     }
20
21     //retorna o número de linhas da lista - Método obrigatório
22     func tableView(_ tableView: UITableView, numberOfRowsInSectionSection section: Int) -> Int { ← 3
23         code
24     }
25
26     //retorna uma célula da tabela para o índice informado - Método obrigatório
27     func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell { ← 4
28         code
29     }
30
31     override func viewDidLoad() {
32         super.viewDidLoad()
33         // Do any additional setup after loading the view, typically from a nib.
34     }
35
36     override func didReceiveMemoryWarning() {
37         super.didReceiveMemoryWarning()
38         // Dispose of any resources that can be recreated.
39     }
40
41 }
```


Table View

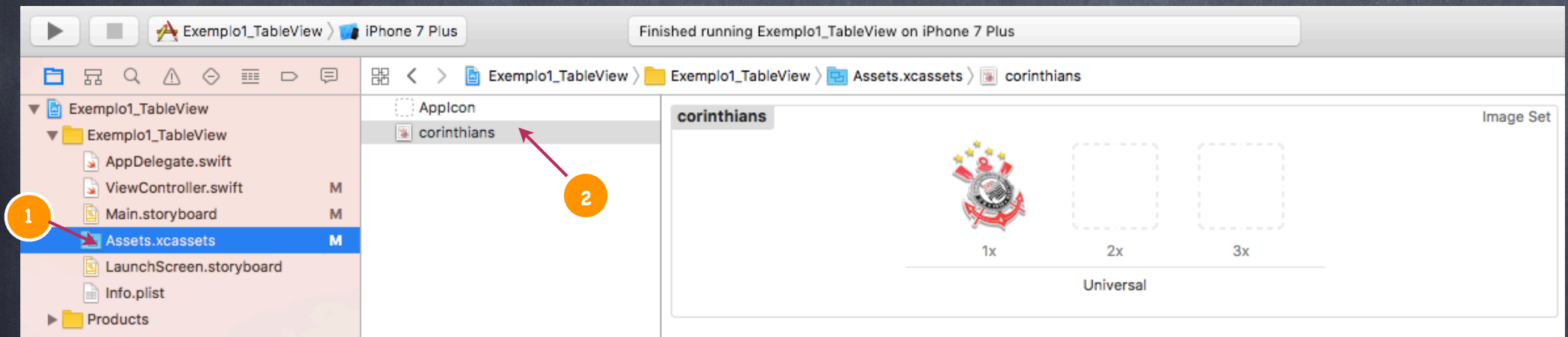
- Faça a implementação dos 3 métodos do protocolo UITableViewDataSource

```
8
9 import UIKit
10
11 class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
12
13     @IBOutlet weak var minhaTableView: UITableView!
14
15     //retorna quantas seções terá nossa lista - Método obrigatório
16     func numberOfSections(in tableView: UITableView) -> Int {
17         return 1
18     }
19
20     //retorna o número de linhas da lista - Método obrigatório
21     func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
22         return 5
23     }
24
25     //retorna uma célula da tabela para o índice informado - Método obrigatório
26     func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
27         let celula = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)
28
29         celula.textLabel?.text = "Item número \(indexPath.row)"
30         celula.imageView?.image = UIImage(named: "corinthians.png")
31         return celula
32     }
33
34 }
```

OBS: Esse é o identificador para reuso da célula, deve ser igual ao digitado no atributo Identifier na pág 20

Table View

- Abra Assets.xcassets (1) e arraste a imagem fornecida para a área indicada (2).



OBS: Existe outra forma de inserir a imagem no projeto veja o slide TableView_ObjC página 22

Table View

- Command + R, execute e veja o resultado.

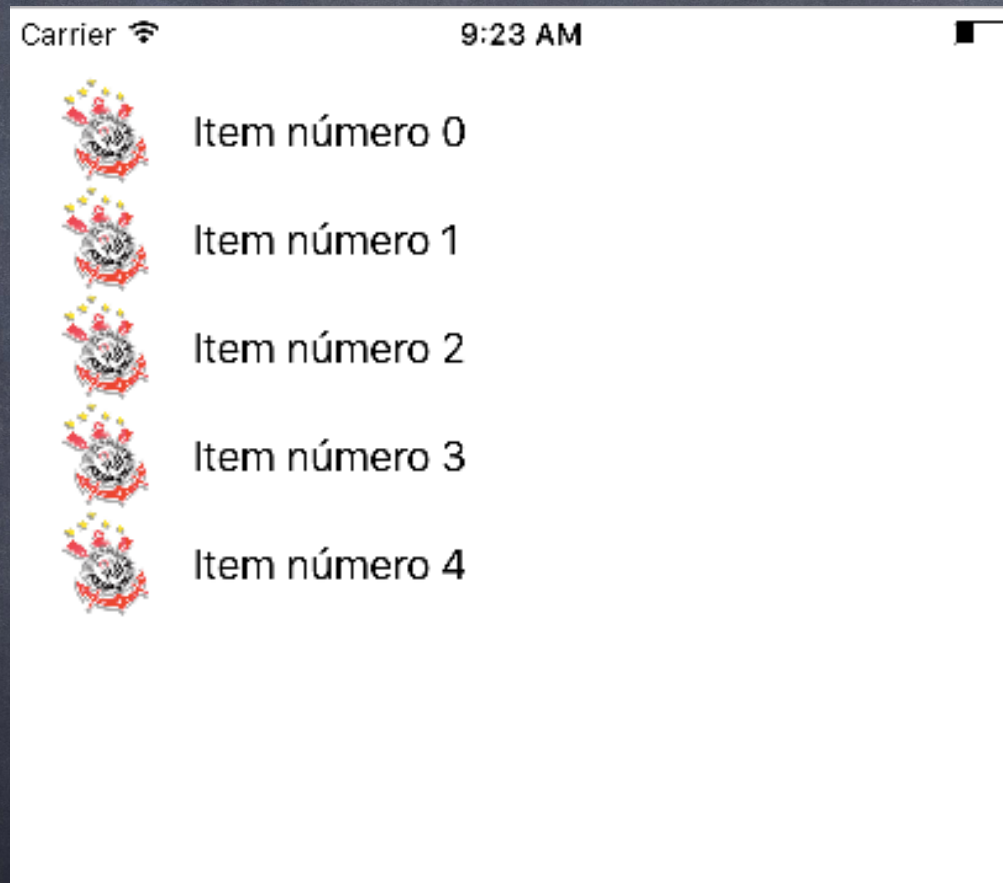


Table View

- Vamos criar um array mutável e descarregá-lo na TableView.

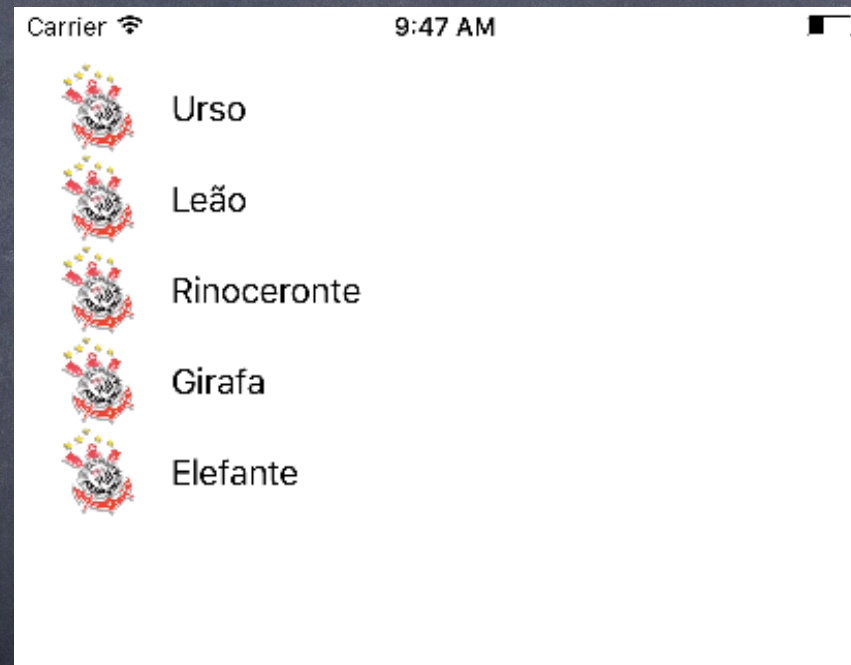
```
1 //  
2 // ViewController.swift  
3 // Exemplo1_TableView  
4 //  
5 // Created by Agesandro Scarpioni on 20/03/17.  
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {  
12  
13     var animal = ["Urso", "Leão", "Rinoceronte", "Girafa", "Elefante"]  
14  
15     @IBOutlet weak var minhaTableView: UITableView!  
16
```

- Comente a linha antiga (1) e crie uma nova para exibir o conteúdo do array (2).

```
26  
27 //retorna uma célula da tabela para o índice informado - Método obrigatório  
28 func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
29     let celula = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)  
30  
31     //celula.textLabel?.text = "Item número \(indexPath.row)" ← 1  
32     celula.textLabel?.text = "\(animal[indexPath.row])"  
33     2 → celula.imageView?.image = UIImage(named: "corinthians.png")  
34     return celula  
35 }
```

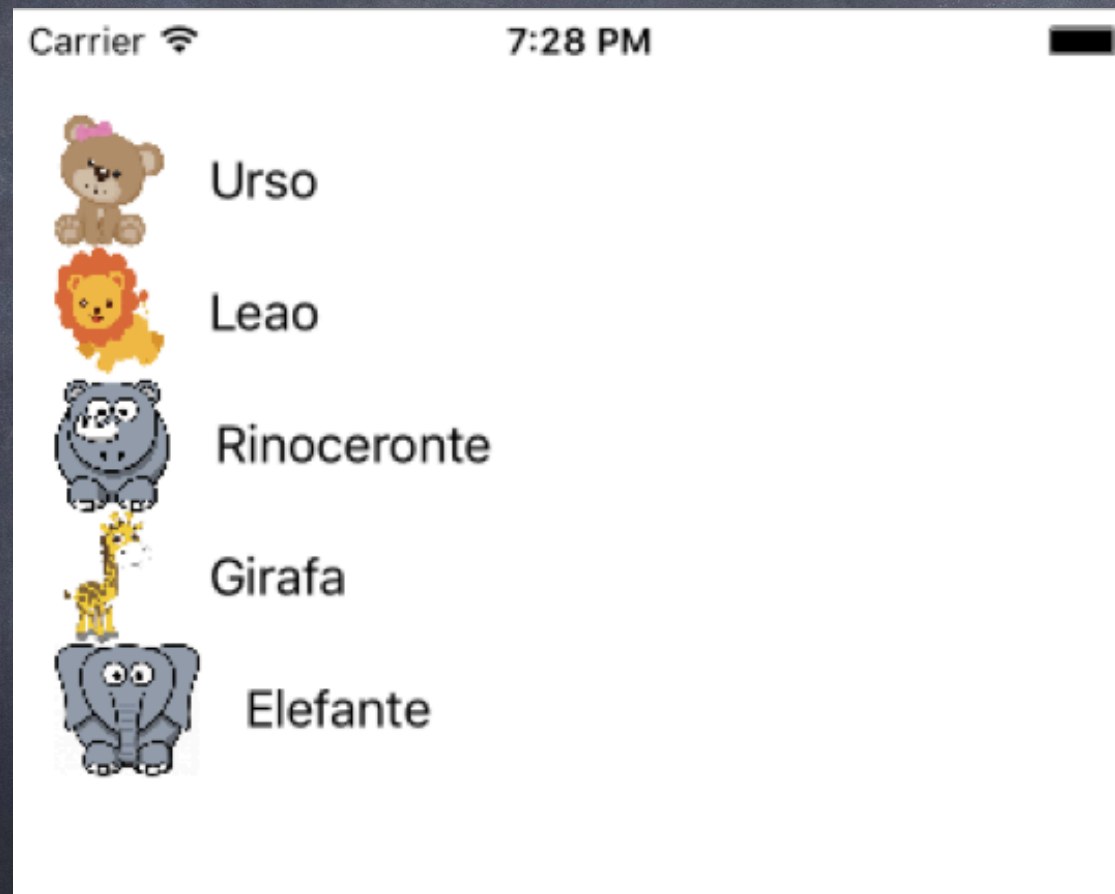

Table View

- Execute o programa e verifique o resultado.



Atividade

- Busque na internet ou baixe do portal algumas imagens de animais do tipo png, monte um array mutável chamado foto e insira os nomes das imagens nesse array, faça aparecer os animais na tableView.



Dica: No arquivo MyPlayground_Array do portal você encontra o código para acessar os itens de um array.

Table View

- Selecione o TableView (1) ligue via connections inspector(2) o protocolo Delegate(3) ao ViewController(4), assim é possível implementar a funcionalidade para exibir uma mensagem quando ocorre um "tap" na lista. O Protocolo Delegate é o responsável pelos métodos chamados quando ocorre algum tipo de toque em uma linha do TableView.

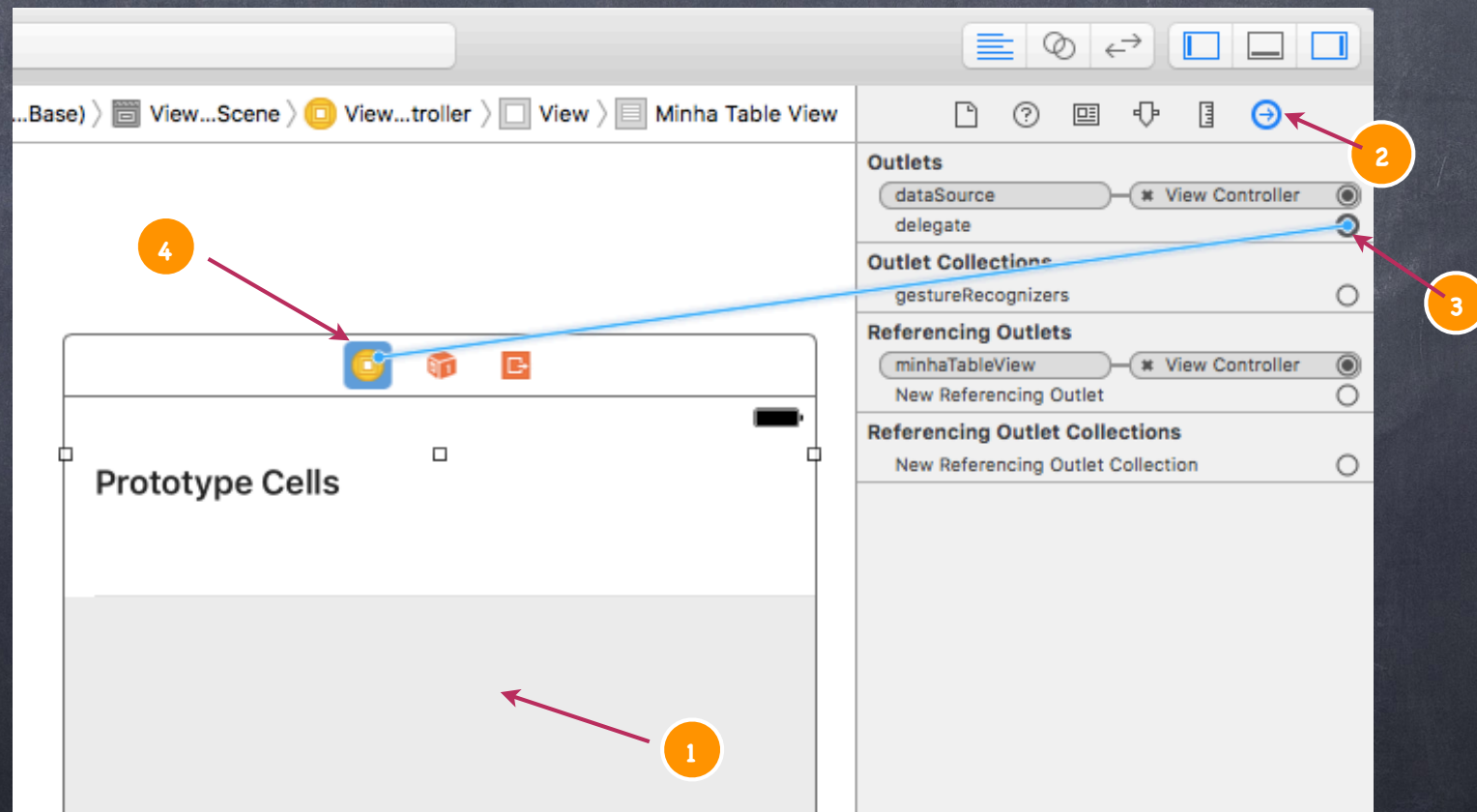


Table View

- No arquivo .swift vamos fazer a seguinte implementação :

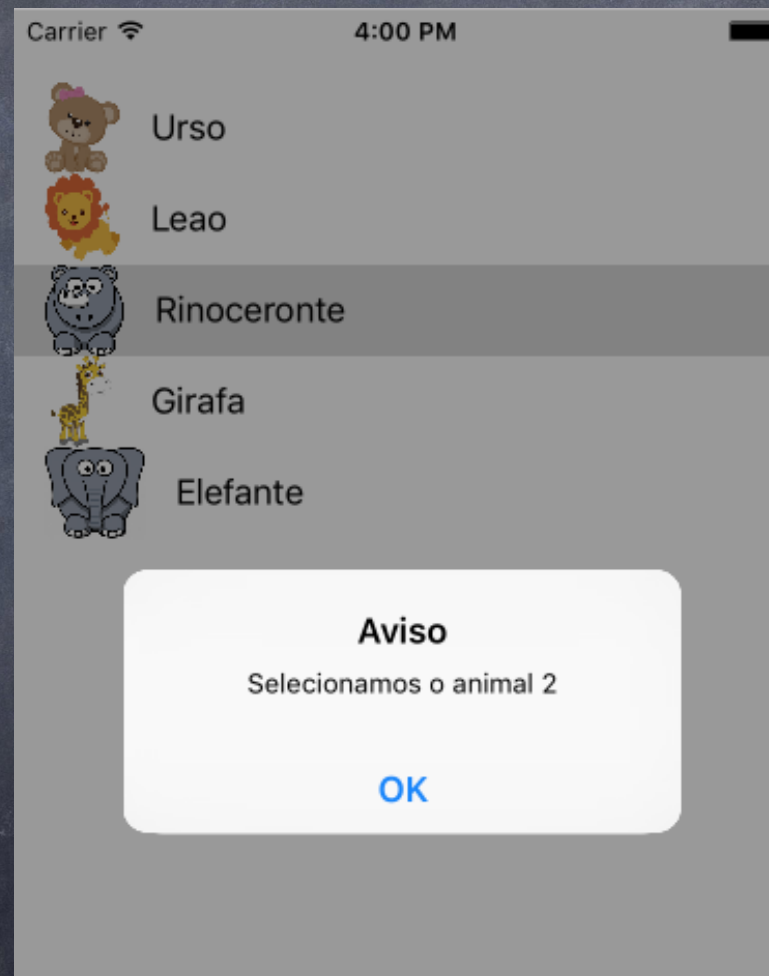
```
37 func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {  
38     let msg = "Selecionamos o animal \(indexPath.row)"  
39  
40     let alerta = UIAlertController(title: "Aviso",  
41                                   message: msg,  
42                                   preferredStyle: .alert)  
43  
44     alerta.addAction(UIAlertAction(title: "OK",  
45                                   style: UIAlertActionStyle.default,  
46                                   handler: nil))  
47  
48     present(alerta, animated: true, completion: nil)  
49  
50 }
```

ATENÇÃO: O método acima é didSelect e não didDeselect

Dica: Você pode criar uma classe só para receber o texto e enviar as mensagens, faremos isso em slides futuros.

Table View

- Command + R para executar:



Obs: Foi apresentado o índice do animal no array, para exibir a posição do animal na tableView adicione 1 ao índice

Table View

- No exemplo abaixo será mostrado o nome do animal ao invés do índice do array, note que a linha 40 foi duplicada e comentada, a linha 39 foi adaptada.

```
36
37 func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
38     //let msg = "Selecionamos o animal \(indexPath.row)"
39     let msg = "Selecionamos o animal \(animal[indexPath.row])"
40
41
42     let alerta = UIAlertController(title: "Aviso",
43                                   message: msg,
44                                   preferredStyle: .alert)
45
46     alerta.addAction(UIAlertAction(title: "OK",
47                                   style: UIAlertActionStyle.default,
48                                   handler: nil))
49
50     present(alerta, animated: true, completion: nil)
51
52 }
```

Obs: O método acima só funciona porque foi ligado o protocolo Delegate ao ViewController pelo Connections Inspector na página 28

Table View

- Command + R para executar:

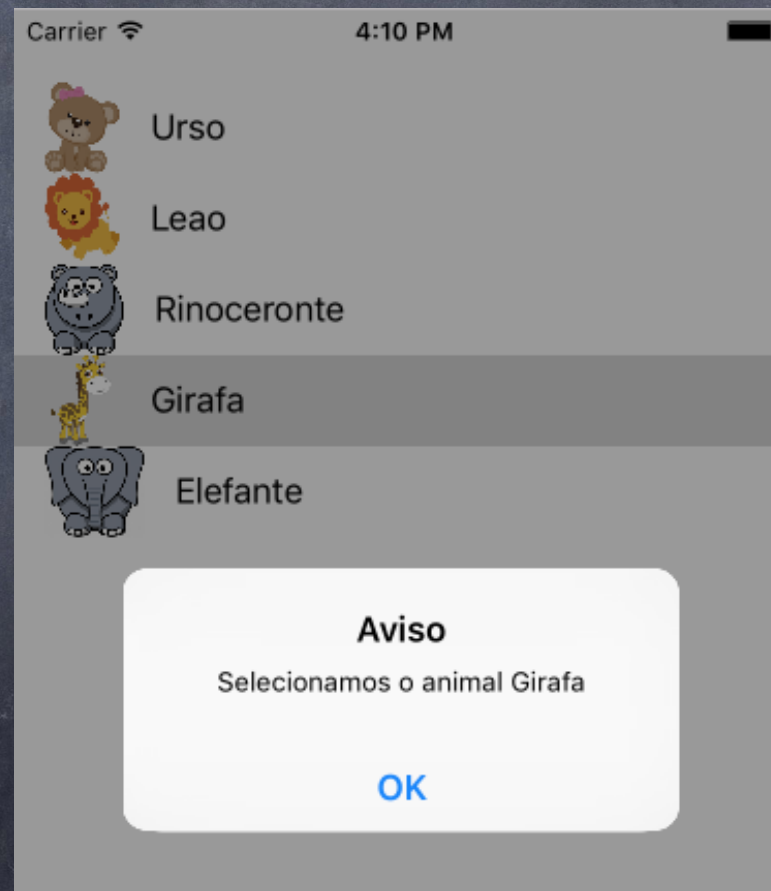


Table View

- Nas linhas abaixo utilize outros métodos para configurar a header e o footer da tabela.

```
53
54 func tableView(_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {
55     let footer = UIImageView(image:UIImage(named:"rodape"))
56     return footer
57 }
58
59 func tableView(_ tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {
60     return 10
61 }
62
63 func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
64     let cabec = UIImageView(image:UIImage(named:"cabecalho"))
65     return cabec
66 }
67
68 func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
69     return 70
70 }
71
```


Table View

- Command + R para executar:

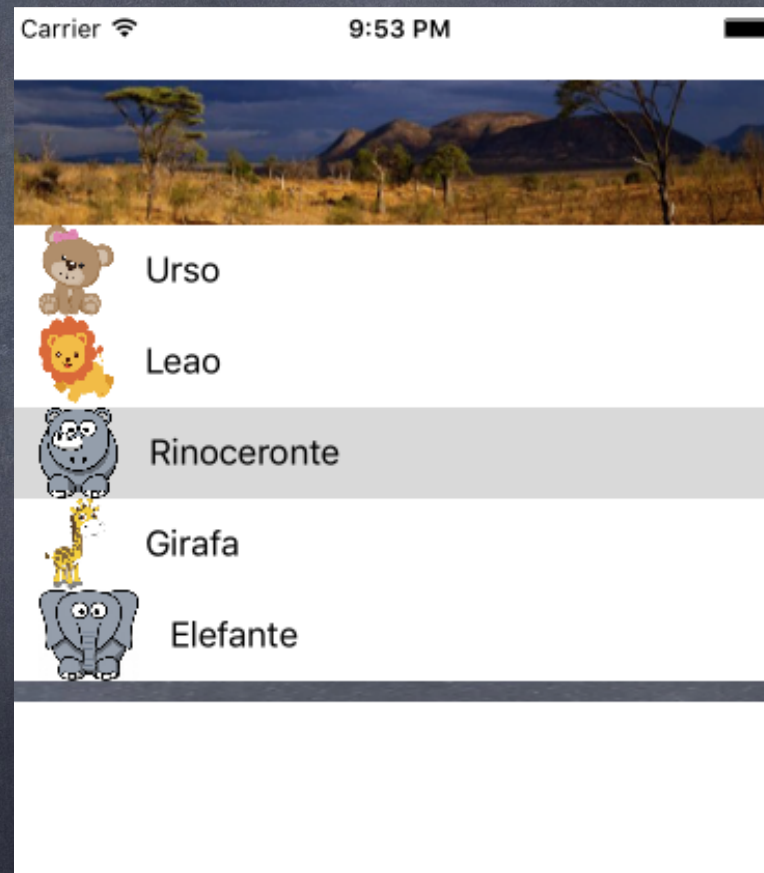
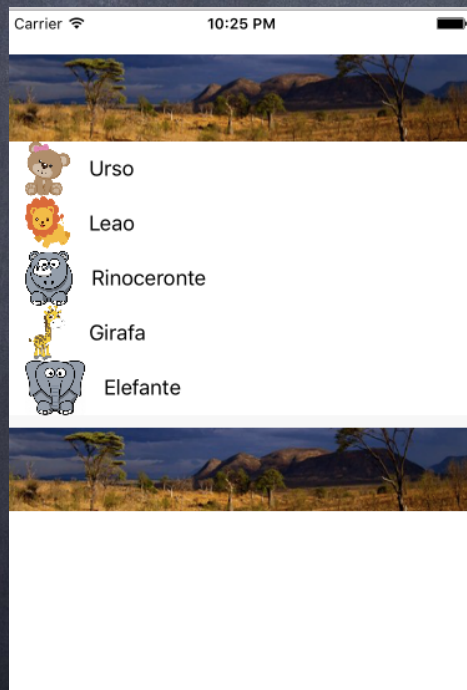


Table View

- Comente a linha que exibe a imagem cinza do rodapé método `viewForFooterInSection`, veja que também é possível atribuir uma imagem pelo atributo `.tableFooterView` no `viewDidLoad`, atribua no footer a mesma imagem do header digitando a linha abaixo:

```
107 minhaTableView.tableFooterView = UIImageView(image: UIImage(named: "cabecalho"))
```



- Obs: para alterar a imagem do header também é possível pelo atributo `.tableHeaderView`.