

TableView

Utilizando StoryBoard

X-Code com ObjC

Prof. Agesandro Scarpioni

agesandro@fiap.com.br

Table View

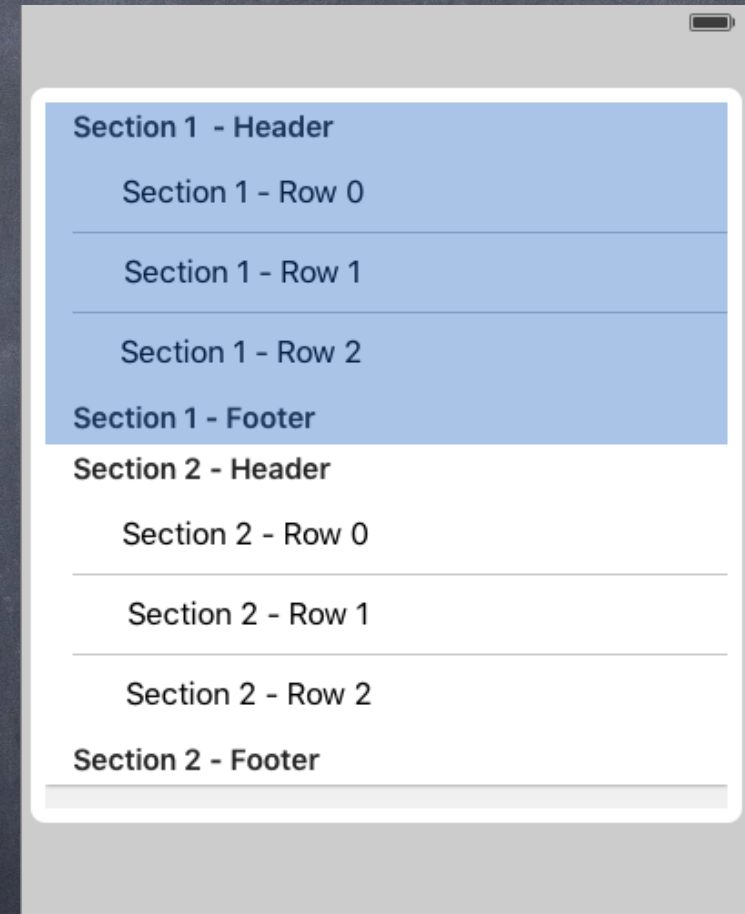
- Veja como é fácil utilizar o TableView um objeto muito utilizado para exibir e manipular listas.

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

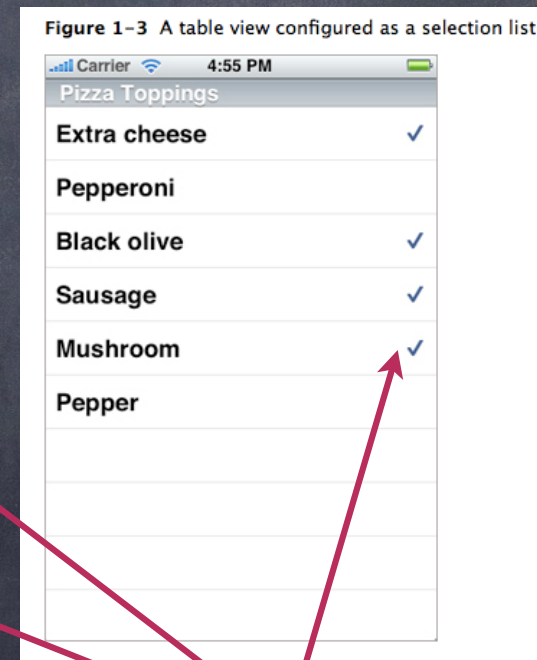
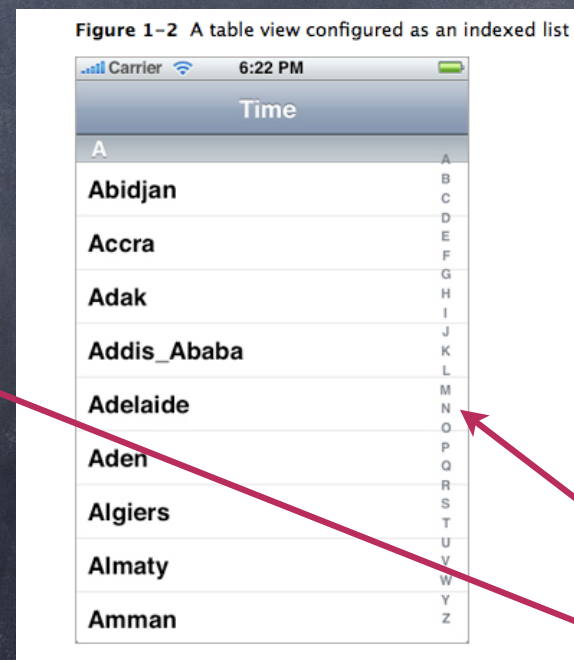
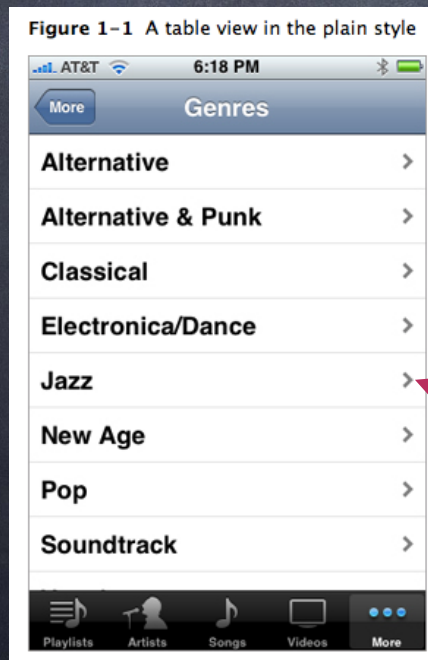
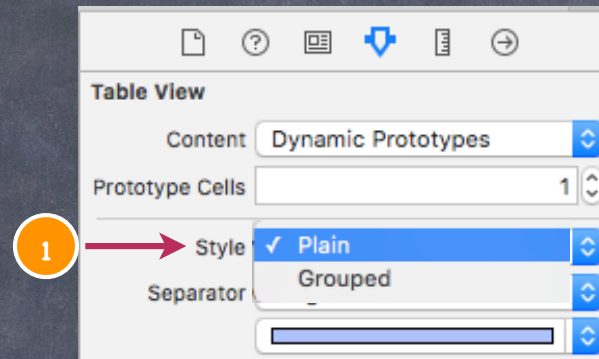
- As tabelas possuem os seguintes elementos: Section, header, footer e row.
- As seções são grupos de linhas de uma tabela.
- O header é o cabeçalho de cada seção, por exemplo: imagine que as seções contenham linhas com nomes de jogos separadas por tipo de consoles, no header será exibido o nome do console 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

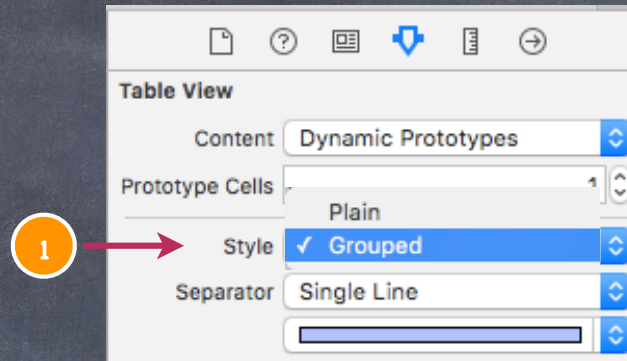
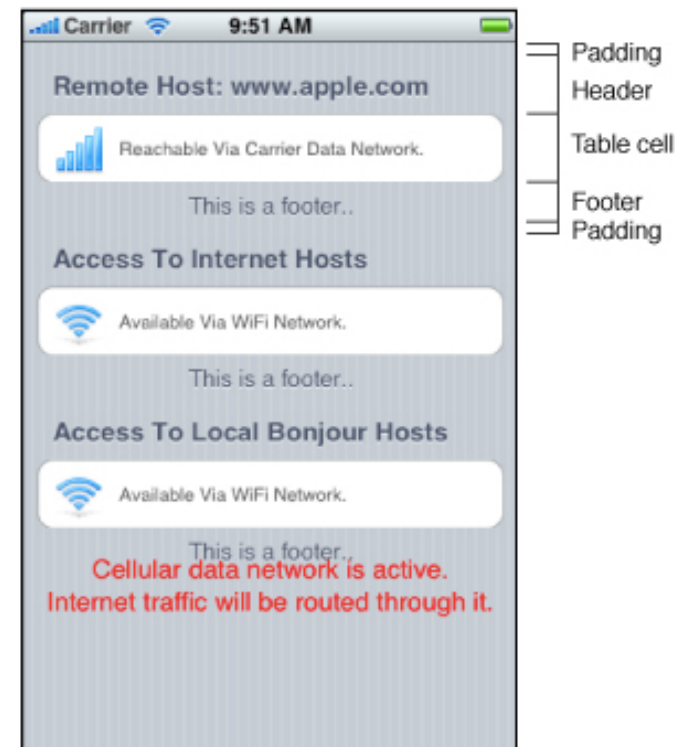


Figure 1-4 A table view in the grouped style



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:

```
18
19 //retorna número de seções da tabela - método obrigatório
20 -(NSInteger) numberOfSectionsInTableView:(UITableView *)tableView;
21
22 //retorna o número de linhas por seção - método obrigatório
23 -(NSInteger) tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section;
24
25 //retorna uma célula da tabela para o índice informado - método obrigatório
26 -(UITableViewCell *) tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath;
27
28 //Título de cabeçalho de cada seção
29 -(NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section;
30
31 //Título de rodapé de cada seção
32 -(NSString *)tableView:(UITableView *)tableView titleForFooterInSection:(NSInteger)section;
33
34 //Edição da tabela pelo usuário, delete ou insert de linhas na tabela
35 -(void) tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath;
36
```

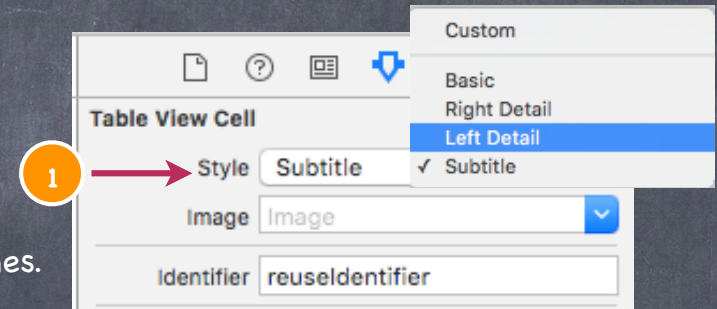

UITableViewDelegate

- 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.

```
37
38 //seleção de um item na tabela
39 -(void) tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath;
40
41 //seleção do botão de acessório
42 -(void) tableView:(UITableView *)tableView accessoryButtonTappedForRowWithIndexPath:(NSIndexPath *)indexPath;
43
44 //alteração de altura (height) de linha (row), cabeçalho (header) e rodapé (footer)
45 -(CGFloat) tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath;
46
47 -(CGFloat) tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section;
48
49 -(CGFloat) tableView:(UITableView *)tableView heightForFooterInSection:(NSInteger)section;
50
51 //customização da view de cabeçalho e rodapé
52 -(UIView *) tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section;
53
54 -(UIView *) tableView:(UITableView *)tableView viewForFooterInSection:(NSInteger)section;
55
```


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

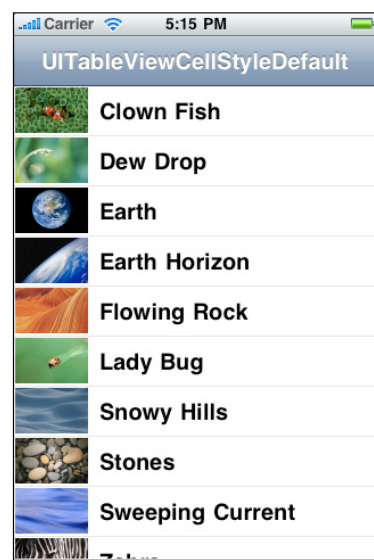


Figure 1-7 Table row style with a subtitle under the title

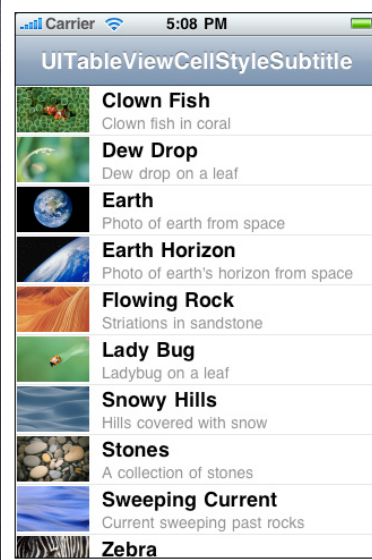


Figure 1-8 Table row style with a right-aligned subtitle

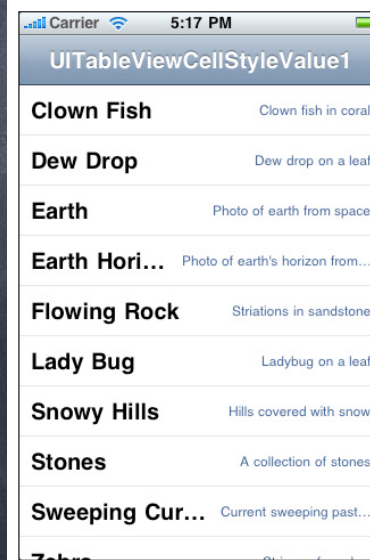
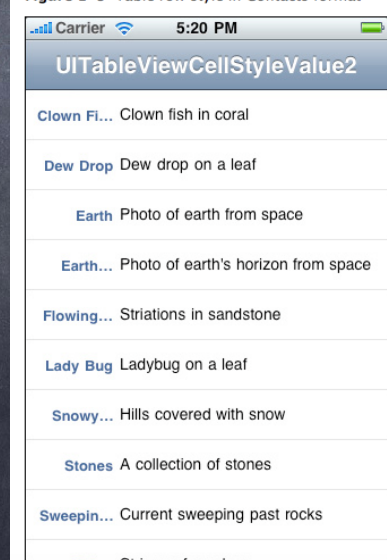
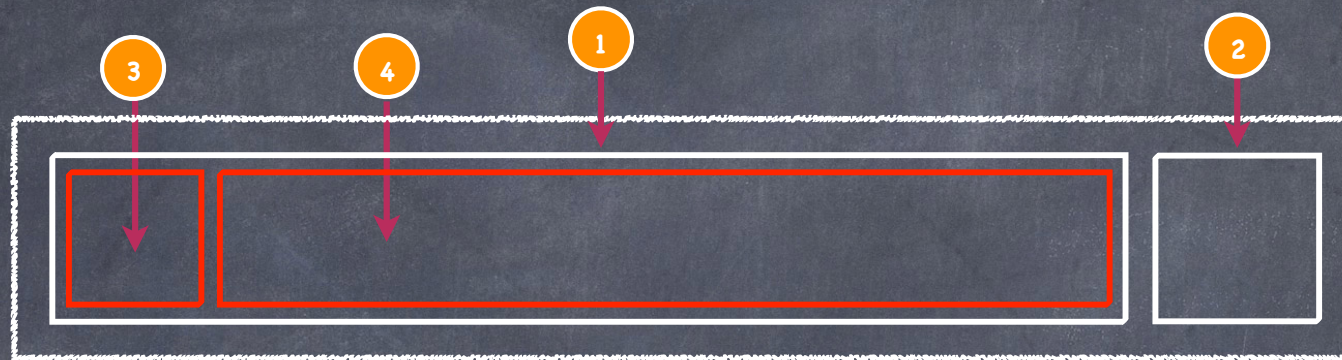


Figure 1-9 Table row style in Contacts format



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

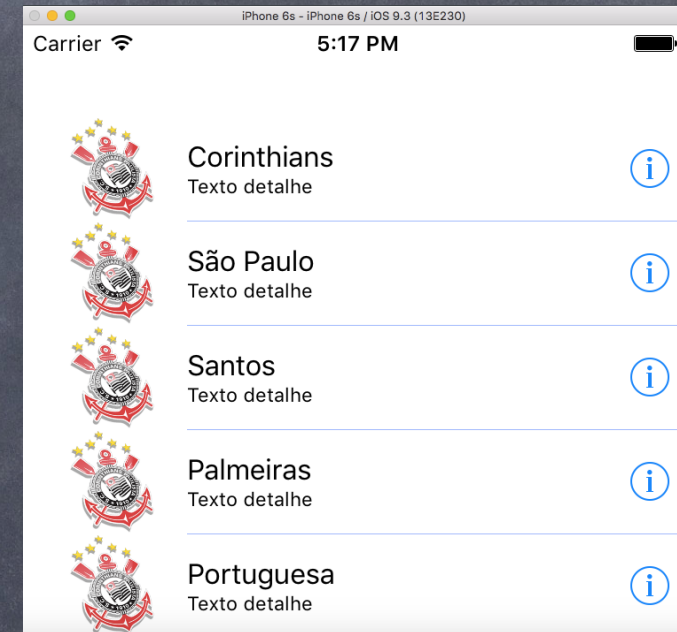
Table View Cell



- 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.



```
50   celula.textLabel.text = [time objectAtIndex:indexPath.row];
51   celula.imageView.image = [UIImage imageNamed:@"corinthians.png"];
52   celula.detailTextLabel.text = @"Texto detalhe";
53   celula.accessoryType= UITableViewCellStyleAccessoryDetailButton;
```

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).

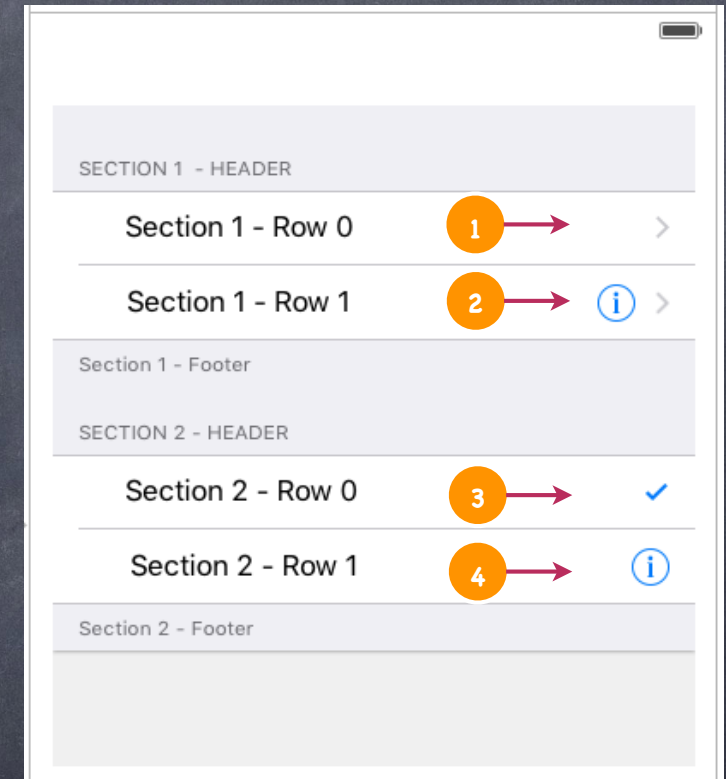


Table View

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

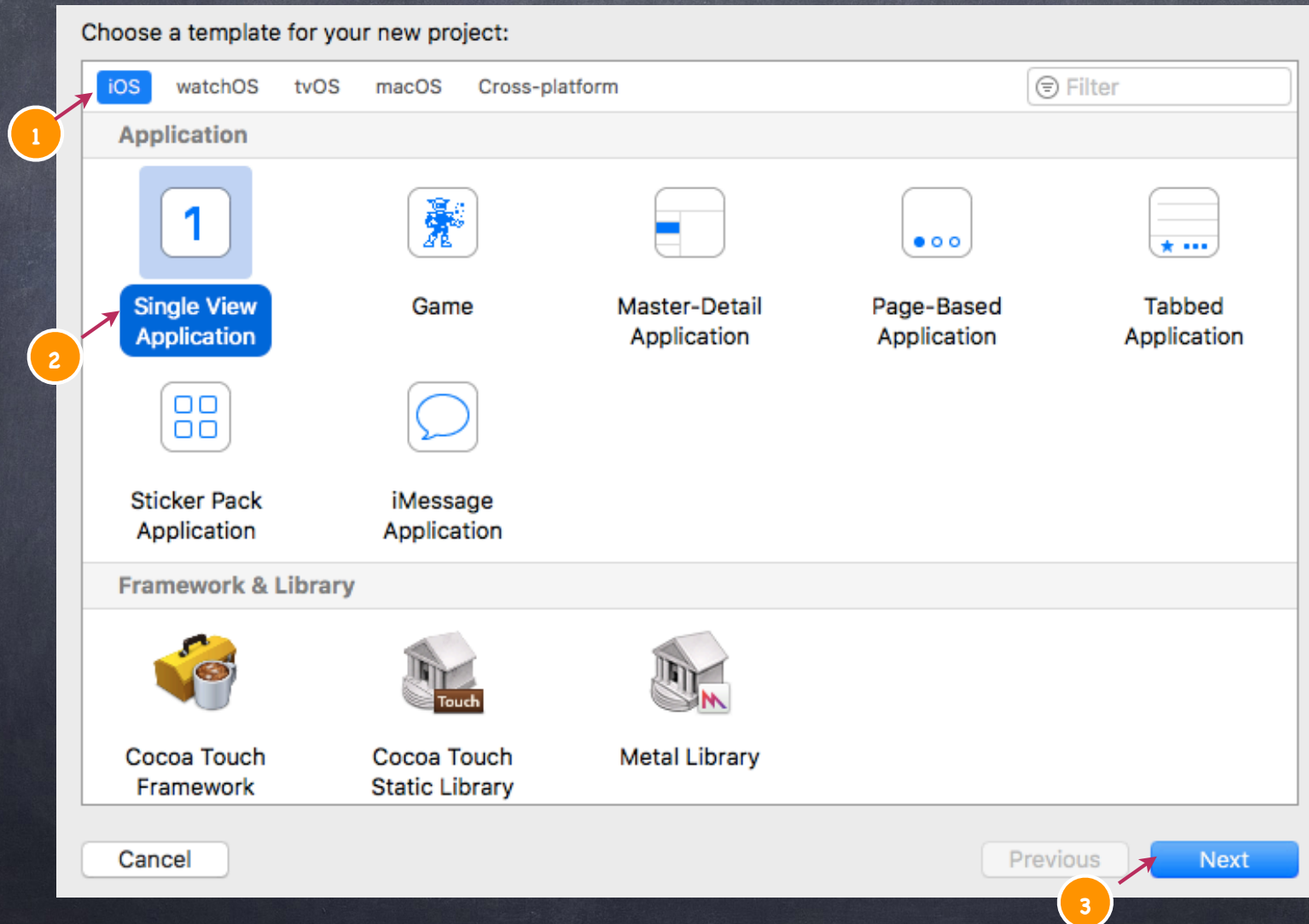


Table View

- Nomeie o projeto como: "Exemplo1_TableView" (4), escolha a linguagem Objective-C (5), depois selecione o 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

FIAP

- 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.h.

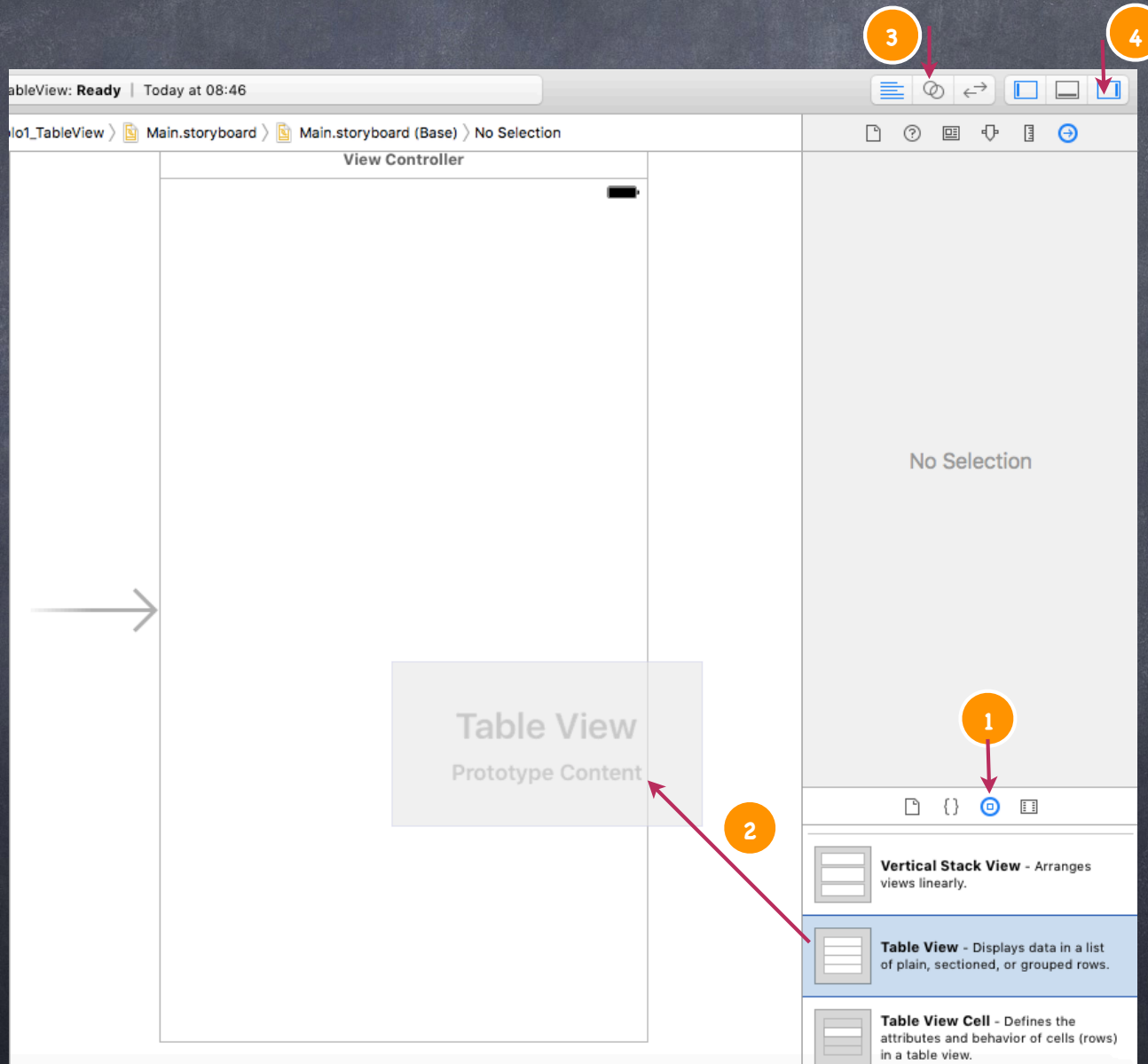


Table View

- Abra chaves no seu arquivo.h, vamos criar um Outlet da TableView clicando com o botão direito sobre o objeto, escolha New Referencing Outlet(1) e arraste até a parte abaixo das chaves, nomeie seu Outlet como minhaTableView(2).

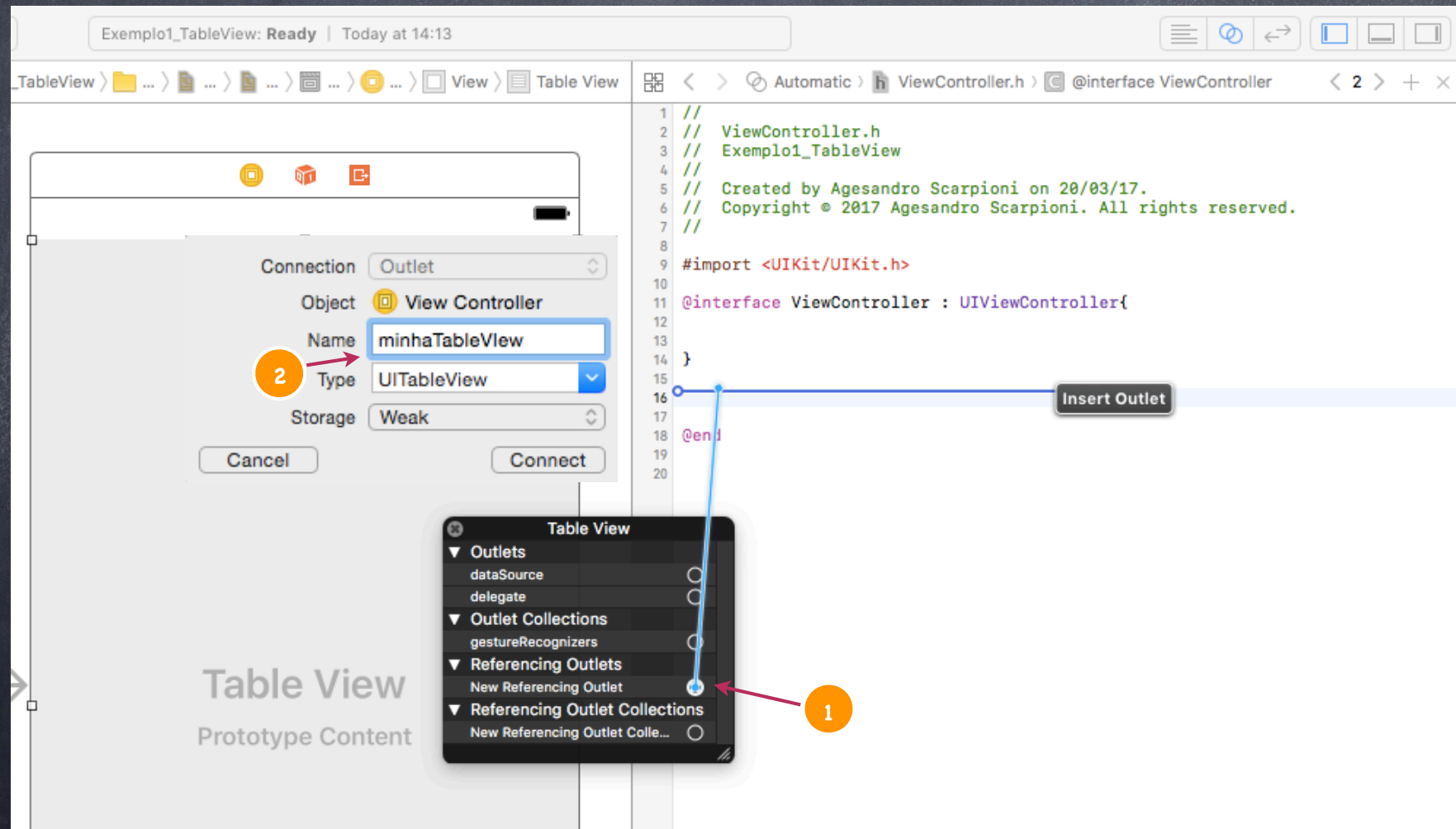


Table View

- No arquivo .h, 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.h
3  // Exemplo1_TableView
4  //
5  // Created by agesandro scarpioni on 15/03/15.
6  // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7  //
8
9  #import <UIKit/UIKit.h>
10
11 @interface ViewController : UIViewController <UITableViewDataSource, UITableViewDelegate> {
12
13
14 }
15
16 @property (weak, nonatomic) IBOutlet UITableView *minhaTableView;
17
18 @end
19
20
```

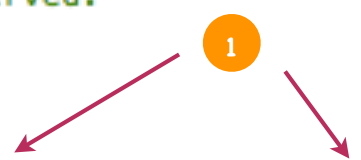


Table View

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

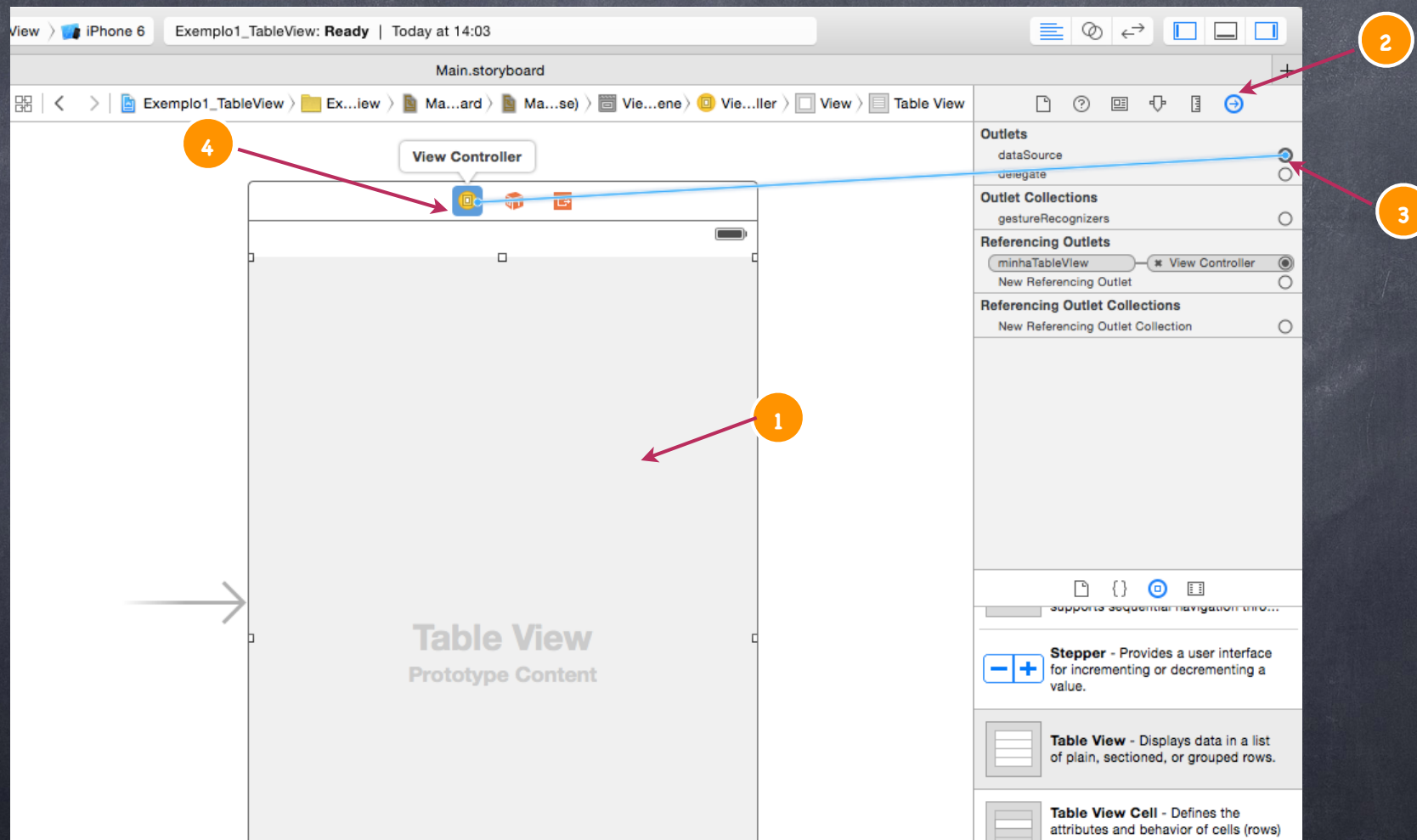


Table View

- Implemente dois protocolos de TableView, implemente inicialmente apenas os métodos do protocolo UITableViewDataSource(4) no arquivo .h, neste local faremos apenas a declaração desses métodos (1) (2) e (3). Lembre-se, esse primeiro protocolo(4) vai fornecer um meio para os dados serem apresentados na lista.

```
1 //
2 // ViewController.h
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/UIKit.h>
10
11 @interface ViewController : UIViewController <UITableViewDataSource, UITableViewDelegate>{
12
13 }
14
15 @property (weak, nonatomic) IBOutlet UITableView *minhaTableView;
16
17 //quantas seções a lista deve possuir - método obrigatório
18 -(NSInteger) numberOfSectionsInTableView:(UITableView *)tableView; ← 1
19
20 //número de linhas da lista - método obrigatório
21 -(NSInteger) tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section; ← 2
22
23 //retorna uma célula da tabela para o índice informado - método obrigatório
24 -(UITableViewCell *) tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath; ← 3
25
26 @end
27
```


Table View

- Copie os métodos do arquivo.h para o arquivo.m, retire o ";", coloque as chaves "{ }", e faça a implementação.

```
32 //implementando os métodos do protocolo UITableViewDataSource
33
34 //Retorna quantas seções terá a lista - método obrigatório
35 -(NSInteger) numberOfSectionsInTableView:(UITableView *)tableView{
36     return 1;
37 }
38 //Retorna o número de linhas da lista - método obrigatório
39 -(NSInteger) tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{
40     return 5; //caso você retorne de um array, dicionário ou plist, você deve dar um count no objeto
41 }
42 //retorna uma célula para cada índice informado - método obrigatório
43 -(UITableViewCell *) tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath{
44     //essa função é chamada para cada item do nosso numberOfRowsInSection
45     //aqui estamos criando a célula(UITableViewCell) para a linha do tableView(tabela)
46     UITableViewCell *celula = [tableView dequeueReusableCellWithIdentifier:@"Celula"];
47     if (celula == nil){
48         //faz cache da célula para evitar criar muitos objetos durante o scroll
49         celula = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"Celula"];
50     }
51     celula.textLabel.text = [NSString stringWithFormat:@"Item número %ld", (long)indexPath.row];
52     celula.imageView.image = [UIImage imageNamed:@"corinthians.png"];
53     return celula;
54 }
55 }
```


Table View

- Copie os métodos do arquivo.h para o arquivo.m, retire o ";", coloque as chaves "{ }", e faça a implementação.

```

1 //
2 // ViewController.m
3 // Exemplo1_TableView
4 //
5 // Created by agesandro scarpion.
6 // Copyright (c) 2013 Agesandro.
7 //
8
9 #import "ViewController.h"
10
11 @interface ViewController ()
12
13 @end
14
15 @implementation ViewController
16
17 - (void)viewDidLoad
18 {
19     [super viewDidLoad];
20     // Do any additional setup after loading the view.
21 }
22
23 - (void)didReceiveMemoryWarning
24 {
25     [super didReceiveMemoryWarning];
26     // Dispose of any resources that can be recreated.
27 }
28
29 //vamos implementar nossos métodos do protocolo
30 -(NSInteger) numberOfSectionsInTableView:(UITableView *)tableView {
31     return 1;
32 }
33
34 -(NSInteger) tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
35     return 5;
36 }
37
38 -(UITableViewCell *) tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
39     //essa função é chamada para cada item de nosso numberOfRowsInSection
40     //criamos a célula para a linha da tabela.
41     static NSString *idCélula = @"Célula";
42     UITableViewCell *celula = [tableView dequeueReusableCellWithIdentifier:idCélula];
43     if (celula==nil){
44         //faz cache da célula para evitar criar muitos objetos durante o scroll
45         celula = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:idCélula];
46     }
47     celula.textLabel.text = [NSString stringWithFormat:@"Item número %d", indexPath.row];
48     celula.imageView.image = [UIImage imageNamed:@"corinthians.png"];
49     return celula;
50 }
51
52 @end

```

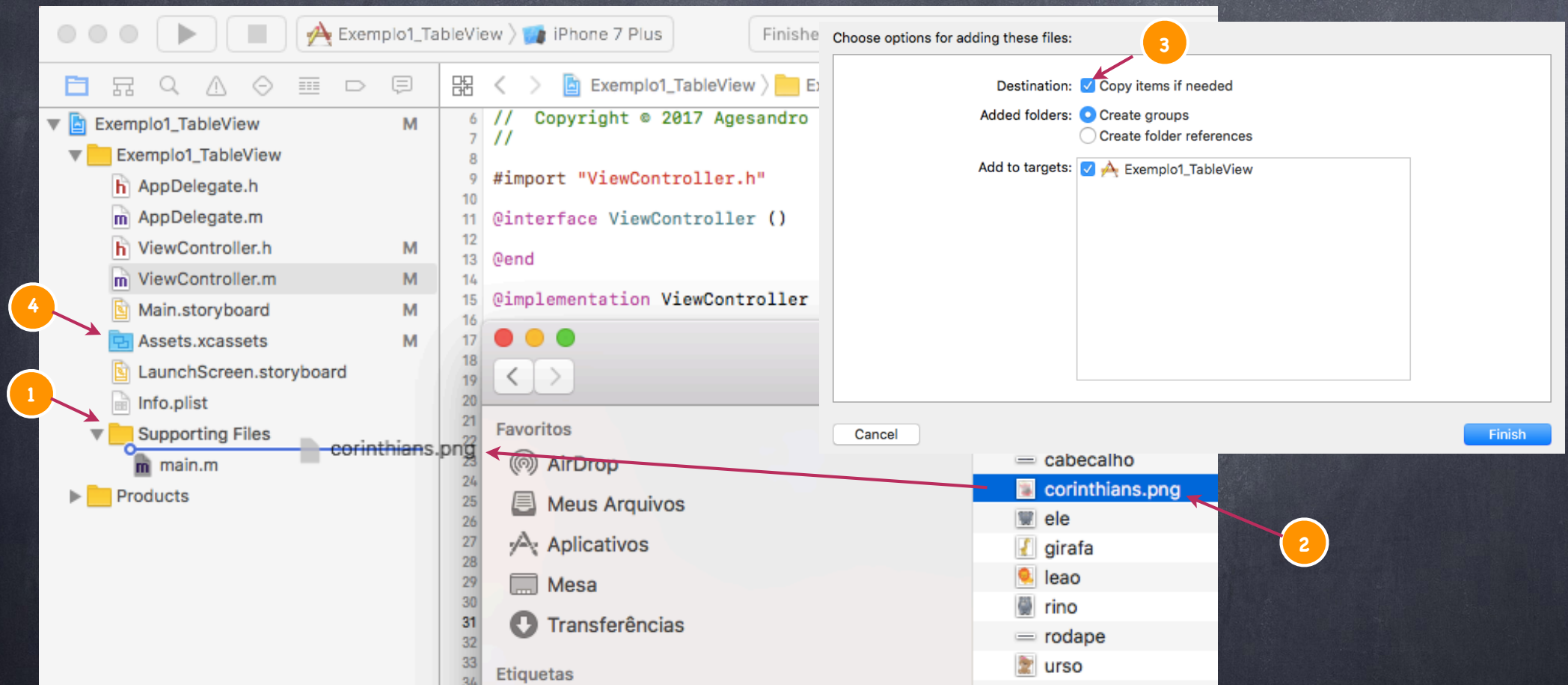
```

26 //implementando os métodos do protocolo UITableViewDataSource
27 //Retorna quantas seções terá nossa lista - método obrigatório
28 -(NSInteger) numberOfSectionsInTableView:(UITableView *)tableView{
29     return 1;
30 }
31 //Retorna o número de linhas da lista - método obrigatório
32 -(NSInteger) tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{
33     return 5; //caso você retorne de um array, dicionário ou plist, você deve dar um count no objeto
34 }
35 //
36 -(UITableViewCell *) tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath{
37     //essa função é chamada para cada item do nosso numberOfRowsInSection
38     //aqui estamos criando a célula(UITableViewCell) para a linha do tableView(tabela)
39     UITableViewCell *celula = [tableView dequeueReusableCellWithIdentifier:@"Célula"];
40     if (celula == nil){
41         celula = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"Célula"];
42     }
43     celula.textLabel.text = [NSString stringWithFormat:@"Item número %ld", (long)indexPath.row];
44     celula.imageView.image = [UIImage imageNamed:@"corinthians.png"];
45     return celula;
46 }
47
48 }

```


Table View

- Dentro de supporting files (1), e arraste via finder (2) a imagem para dentro da pasta, marque o primeiro checkbox (3) para que uma cópia da imagem vá para a pasta, caso contrário apenas teremos um link.



OBS: Existe outra forma de inserir a imagem no projeto pelo images.xcassets(4), veja slide TableView_Swift pág 23

Table View

- Command + R, execute e veja o resultado.

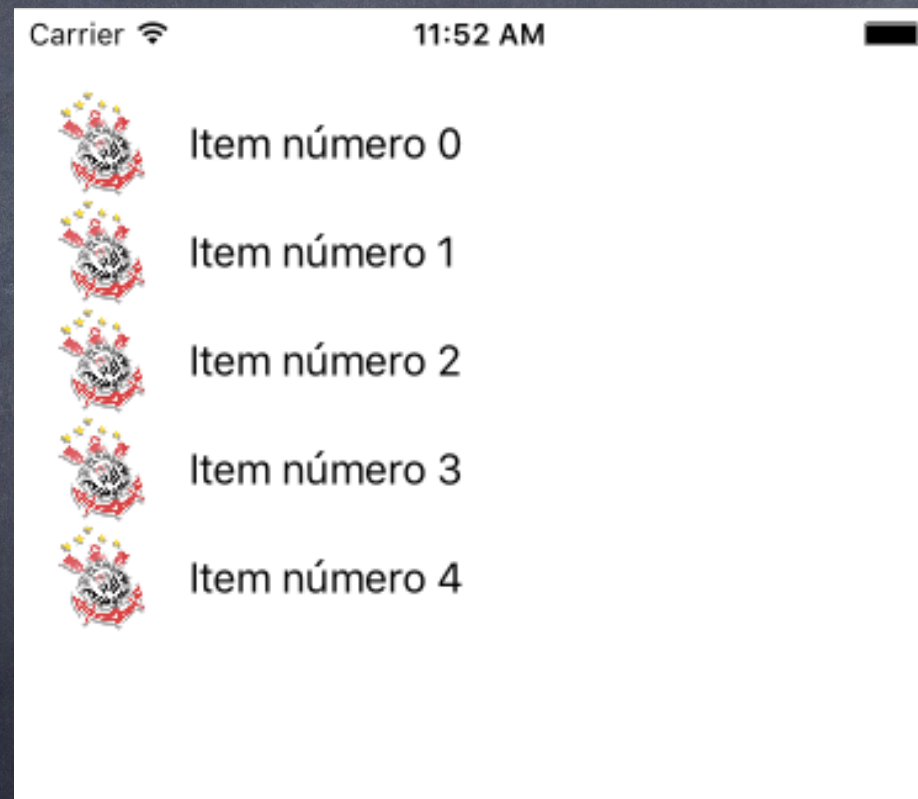


Table View

- No arquivo .h crie um array mutável.

```
1  //
2  //  ViewController.h
3  //  Exemplo1_TableView
4  //
5  //  Created by agesandro scarpioni on 15/03/15.
6  //  Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7  //
8
9  #import <UIKit/UIKit.h>
10
11 @interface ViewController : UIViewController <UITableViewDataSource, UITableViewDelegate> {
12     NSMutableArray *animal; ← 1
13 }
14
15
16 * @property (weak, nonatomic) IBOutlet UITableView *minhaTableView;
17
18 //quantas seções terá nossa lista - método obrigatório
19 -(NSInteger) numberOfSectionsInTableView:(UITableView *)tableView;
20
21 //número de linhas da Lista - método obrigatório
22 -(NSInteger) tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section;
23
24 //retorna uma célula da tabela para o índice informado - método obrigatório
25 -(UITableViewCell *) tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath;
26
27
28 @end
29
```


Table View

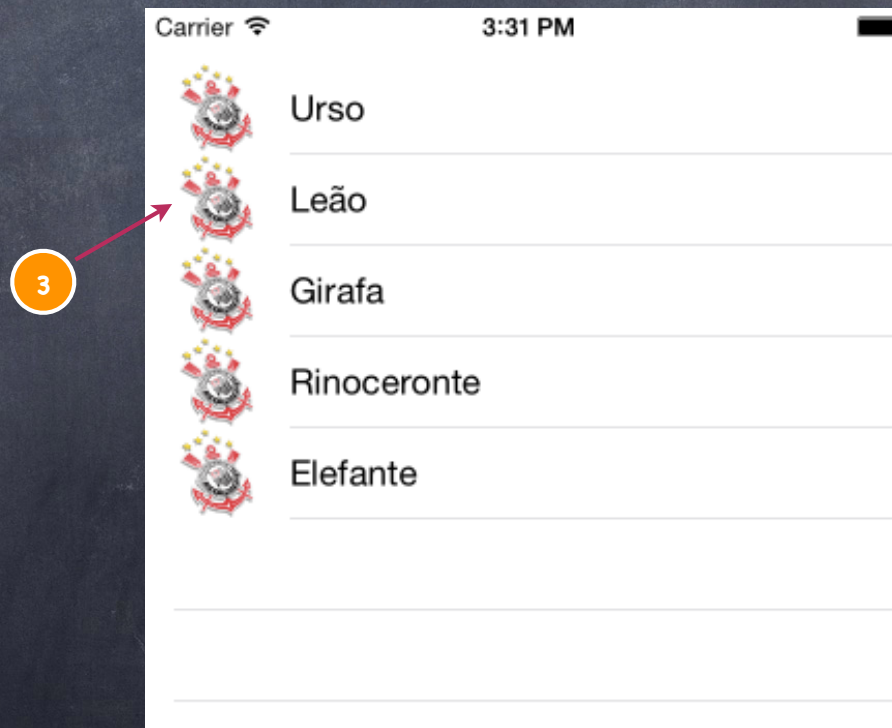
- No arquivo.m , popule o array mutável como demonstrado abaixo, dentro de viewDidLoad.

```
1 //
2 // ViewController.m
3 // Exemplo1_TableView
4 //
5 // Created by agesandro scarpioni on 15/03/15.
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import "ViewController.h"
10
11 @interface ViewController ()
12
13 @end
14
15 @implementation ViewController
16
17 - (void)viewDidLoad {
18     [super viewDidLoad];
19
20     animal = [[NSMutableArray alloc] init];
21     [animal addObject:@"Urso"];
22     [animal addObject:@"Leão"];
23     [animal addObject:@"Girafa"];
24     [animal addObject:@"Rinoceronte"];
25     [animal addObject:@"Elefante"];
26
27 }
```


Table View

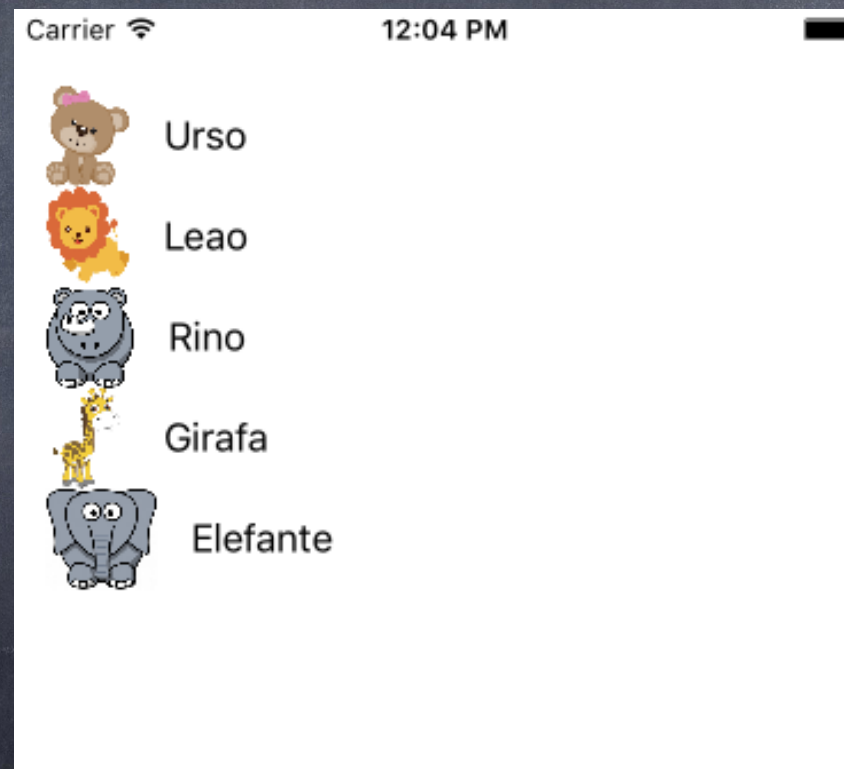
- Ainda no arquivo.m, coloque um comentário na linha da implementação(1) que movimenta os números de ítems para a TableView, em seguida insira a linha indicada abaixo(2) para que os dados do array sejam exibidos na tabela. Execute o programa e verifique o resultado (3).

```
54 1 //celula.textLabel.text = [NSString stringWithFormat:@"Item número %ld", (long)indexPath.row];  
55   celula.textLabel.text = [animal objectAtIndex:indexPath.row]; ← 2  
56   celula.imageView.image = [UIImage imageNamed:@"corinthians.png"];  
57   return celula;
```



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 slide Aula_13_201X_Array_objC 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.

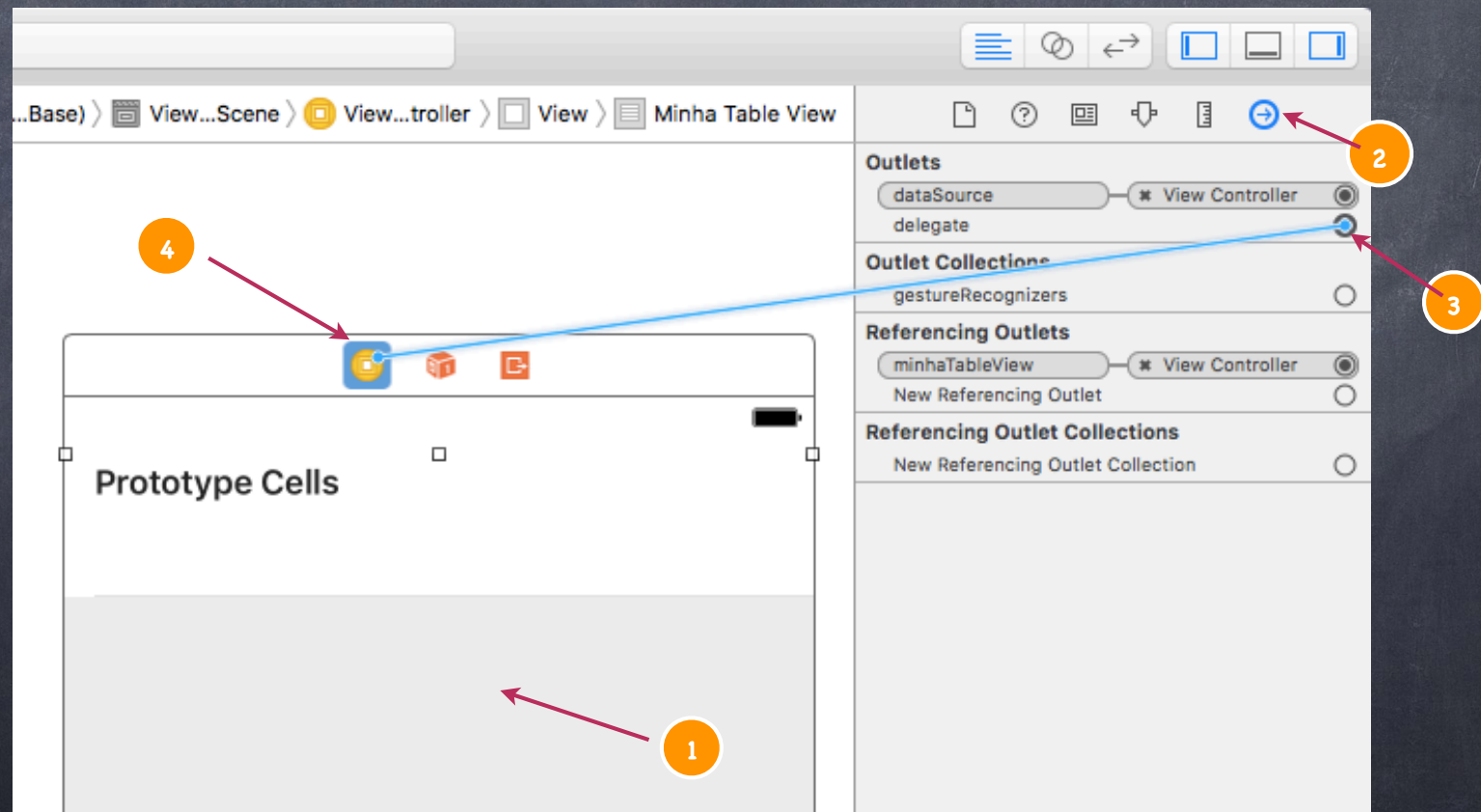


Table View

- No arquivo .h vamos colocar a seguinte assinatura :

```
28 // Implementando o protocolo UITableViewDelegate
29 // O método abaixo é chamado quando selecionamos uma linha da tableview
30 -(void) tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath;
31
```

ATENÇÃO: O método acima é didSelect e não didDeselect

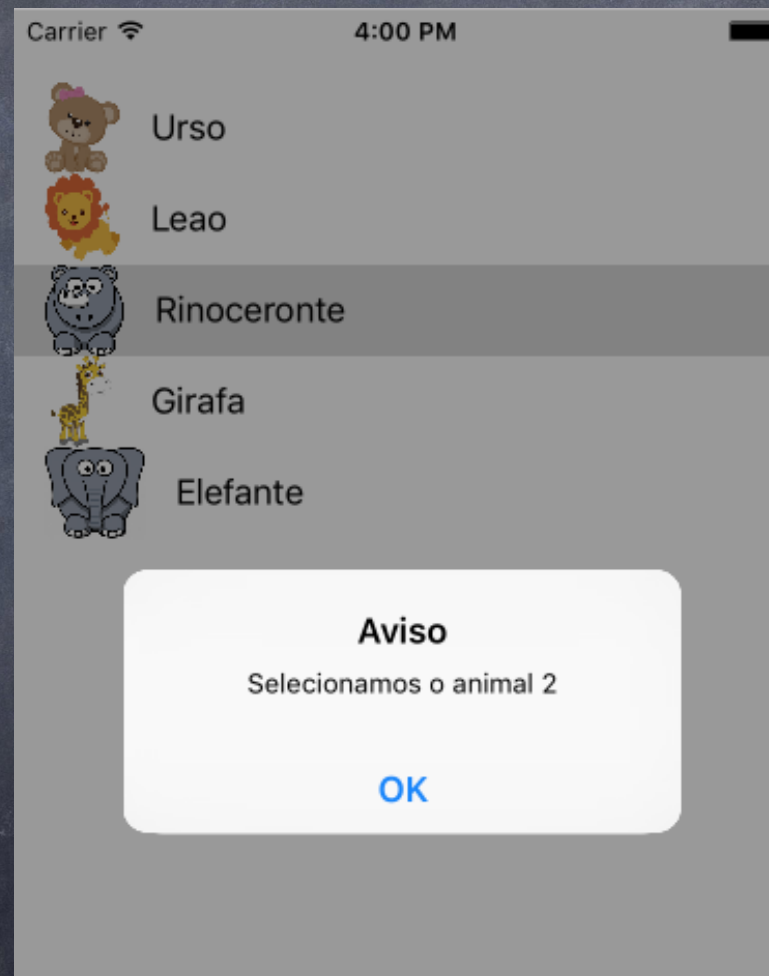
- No arquivo .m vamos fazer a seguinte implementação :

```
55 //assinatura do protocolo delegate
56 -(void) tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath{
57     NSString *msg = [NSString stringWithFormat:@"Selecionamos o animal %ld", (long)indexPath.row];
58
59     UIAlertController *alerta = [UIAlertController
60                                 alertControllerWithTitle:@"Aviso"
61                                 message:msg
62                                 preferredStyle:UIAlertControllerStyleAlert];
63
64     UIAlertAction *ok = [UIAlertAction
65                         initWithTitle:@"OK"
66                         style:UIAlertActionStyleDefault
67                         handler:^(UIAlertAction * _Nonnull action) {
68                             [alerta dismissViewControllerAnimated:YES completion:nil];
69                         }];
70
71     [alerta addAction:ok];
72     [self presentViewController:alerta animated:YES completion:nil];
73 }
74 //Lembre-se que o método acima só funciona porque ligamos o protocolo delegate ao ViewController
75
```

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 57 foi duplicada e comentada, a linha 58 foi adaptada.

```
55 //assinatura do protocolo delegate
56 -(void) tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath{
57     //NSString *msg = [NSString stringWithFormat:@"Selecionamos o animal %ld", (long)indexPath.row];
58     NSString *msg = [NSString stringWithFormat:@"Selecionamos o animal %@", [animal objectAtIndex:indexPath.row]];
59
60     UIAlertController *alerta = [UIAlertController
61                                 alertControllerWithTitle:@"Aviso"
62                                 message:msg
63                                 preferredStyle:UIAlertControllerStyleAlert];
64
65     UIAlertAction *ok = [UIAlertAction
66                         actionWithTitle:@"OK"
67                         style:UIAlertActionStyleDefault
68                         handler:^(UIAlertAction * _Nonnull action) {
69                             [alerta dismissViewControllerAnimated:YES completion:nil];
70                         }];
71
72     [alerta addAction:ok];
73     [self presentViewController:alerta animated:YES completion:nil];
74 }
75 //Lembre-se que o método acima só funciona porque ligamos o protocolo delegate ao ViewControler
76
```


Table View

- Command + R para executar:

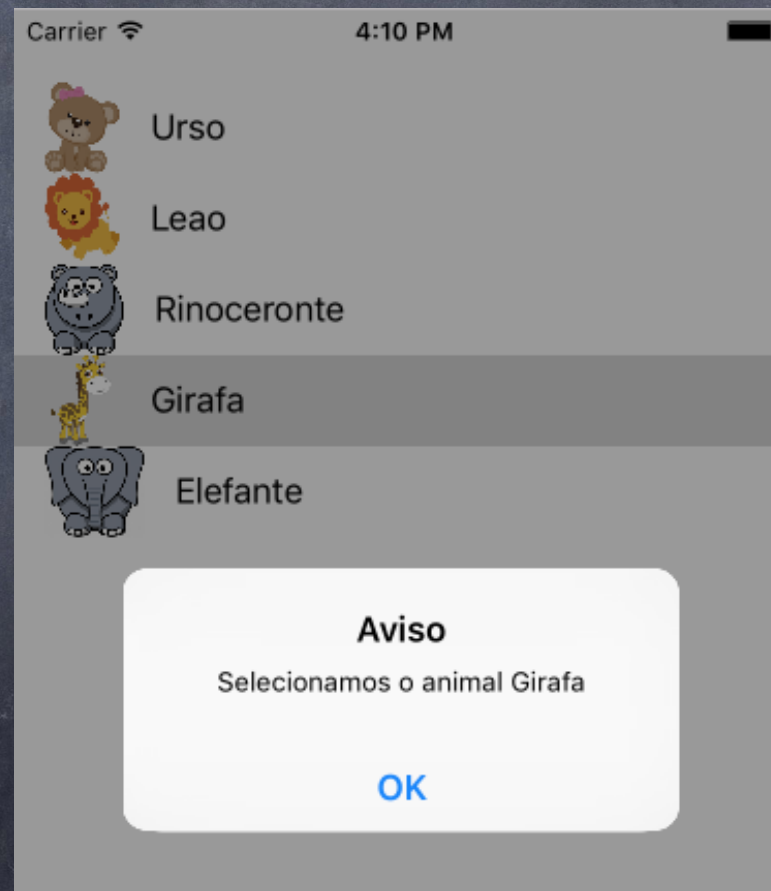


Table View

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

```
79
80 -(UIView *)tableView:(UITableView *)tableView viewForFooterInSection:(NSInteger)section{
81     UIImageView *img = [[UIImageView alloc] initWithImage:[UIImage imageNamed:@"rodape"]];
82     return img;
83 }
84 -(CGFloat) tableView:(UITableView *)tableView heightForFooterInSection:(NSInteger)section{
85     return 10;
86 }
87
88 -(UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section{
89     UIImageView *img = [[UIImageView alloc] initWithImage:[UIImage imageNamed:@"cabecalho"]];
90     return img;
91 }
92
93 -(CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section{
94     return 70;
95 }
96
```


Table View

- Command + R para executar:

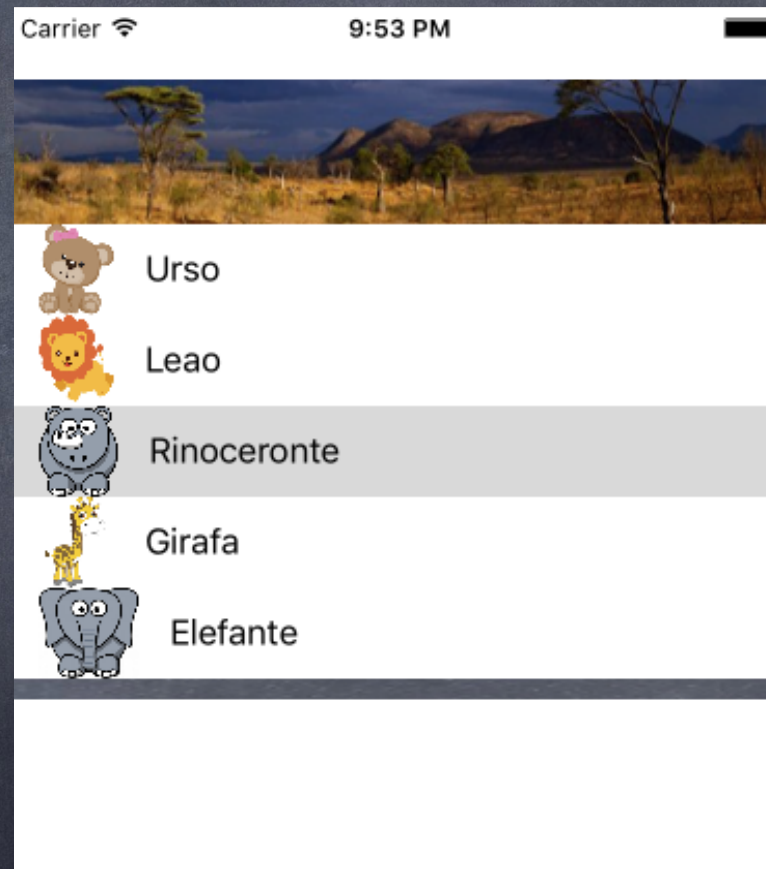
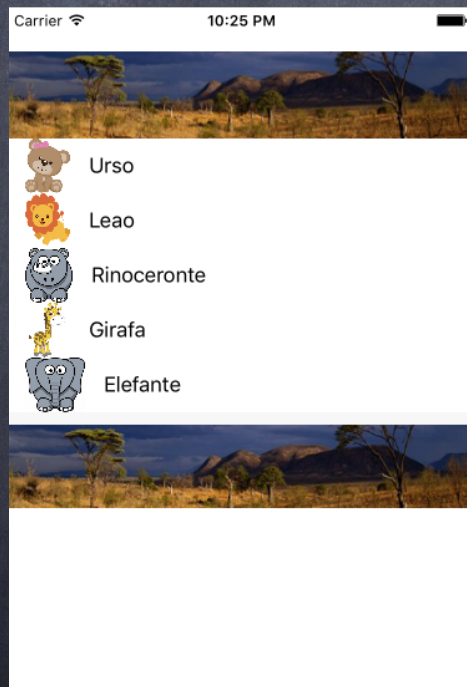


Table View

- 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:

```
35 self.minhaTableView.tableFooterView = [[UIImageView alloc] initWithImage:[UIImage imageNamed:@"cabecalho"]];
```



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