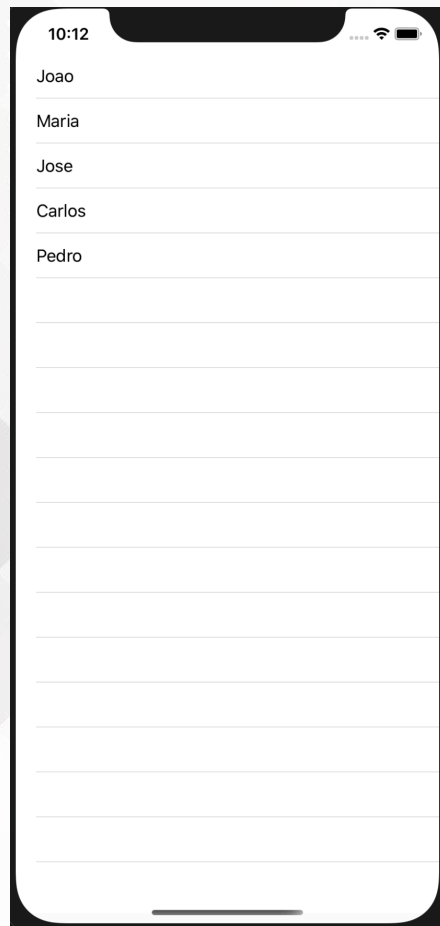




TableView Avançado



Para este exemplo, vamos deixar pronto uma tableView carregando 5 nomes
"Joao" "Maria" "Jose" "Carlos" "Pedro"





Detectando o click do usuário



Implementamos o delegate `didSelectRowAt` para capturar aonde o usuário clicou na célula. Neste exemplo, imprimimos o nome da pessoa que foi clicada.


```
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {  
    print("clicou em: \(arrayNomes[indexPath.row])")  
}
```





Habilitar para remover
um item da célula

Responsável por gerenciar as interações do usuário com a TableView, possui os métodos para gerenciar seleções, configurar cabeçalhos e rodapés de seção, excluir e reordenar células e executar outras ações em uma TableView.



Primeiro precisamos liberar pro usuário poder editar a tableView. Para isso implementamos o delegate `canEditRowAt` e retornamos `true`

```
func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool {  
    return true  
}
```



Após isso, vamos capturar que o usuário clicou no delete, implementando o delegate editingStyle e vamos remover o usuario do array e vamos da o reloadData() para atualizar a tabela novamente

```
func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle,
forRowAt indexPath: IndexPath) {
    arrayNomes.remove(at: indexPath.row)
    tableView.reloadData()
}
```

No final, teremos este código

```
func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool {  
    return true  
}  
  
func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle,  
    forRowAt indexPath: IndexPath) {  
    arrayNomes.remove(at: indexPath.row)  
    tableView.reloadData()  
}
```

10:41



Joao

Maria

Jose

Carlos

Pedro



Criando uma célula customizada



Crie uma nova classe que herda de
UITableViewCell com o nome de CustomCell

Class: CustomCell

Subclass of: UITableViewCell

☐ Also create XIB file

Language: Swift



Crie um IBOutlet de uma label
programaticamente no código, dentro da
classe CustomCell

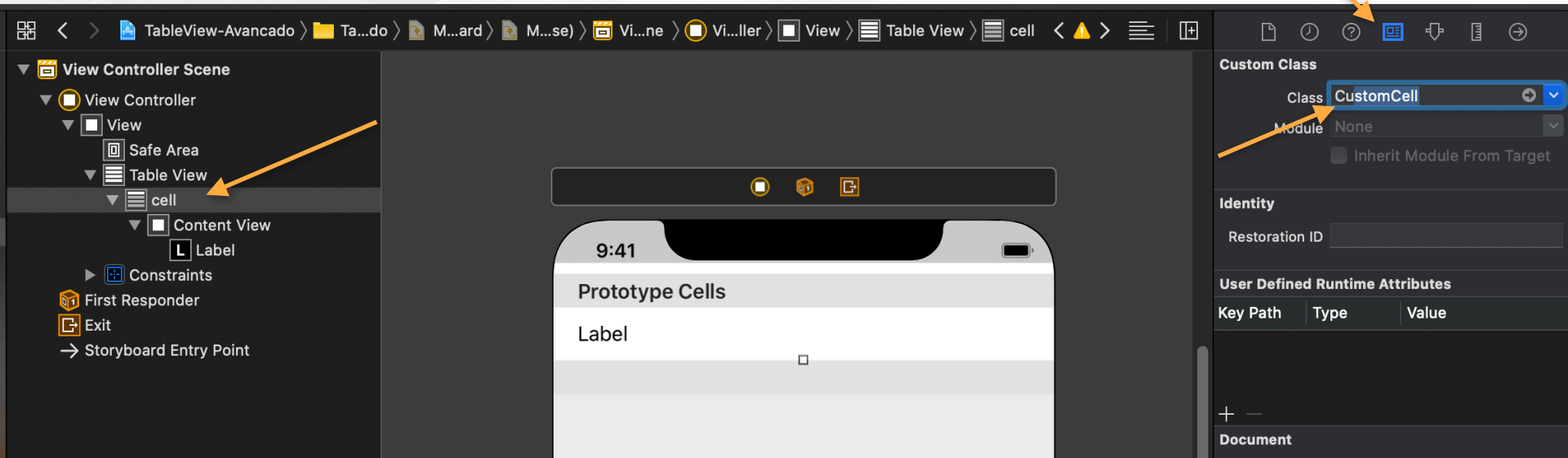
```
10  
11 class CustomCell: UITableViewCell {  
12  
13     @IBOutlet weak var myLabel: UILabel!|  
14
```



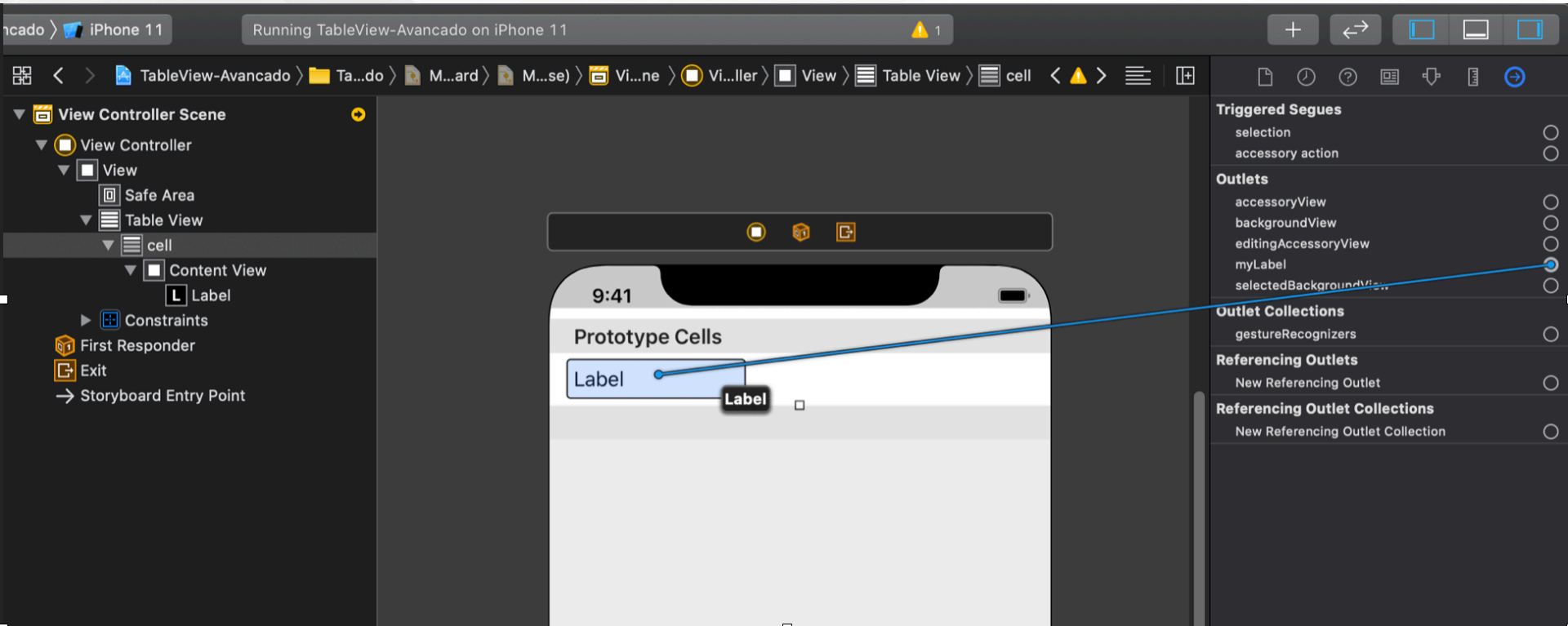
Crie uma função setup dentro da classe CustomCell pedindo como parâmetro um nome e popule este nome na sua label

```
func setup(nome: String) {  
    myLabel.text = nome  
}
```


Dentro do storyboard, atribua a classe CustomCell para controlar a sua célula.



Após atribuir a classe para controlar a sua célula, faz a conexão do outlet a sua label



No final sua classe irá esta assim

Label
conectada

```
8
9  import UIKit
10
11  class CustomCell: UITableViewCell {
12
13      @IBOutlet weak var myLabel: UILabel!
14
15      override func awakeFromNib() {
16          super.awakeFromNib()
17          // Initialization code
18      }
19
20      override func setSelected(_ selected: Bool, animated: Bool) {
21          super.setSelected(selected, animated: animated)
22
23          // Configure the view for the selected state
24      }
25
26      func setup(nome: String) {
27          myLabel.text = nome
28      }
29
30  }
```



Agora voltando para sua ViewController,
vamos fazer o casting(converter) a sua célula
para o tipo CustomCell

```
let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath) as?  
CustomCell
```



Após converter, vamos usar o método setup e seu método setup para configurar o item da célula. O código no final ficará assim

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
  
    let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath) as? CustomCell  
  
    cell?.setup(nome: arrayNomes[indexPath.row])  
  
    return cell ?? UITableViewCell()  
}
```



Bora Codar :)