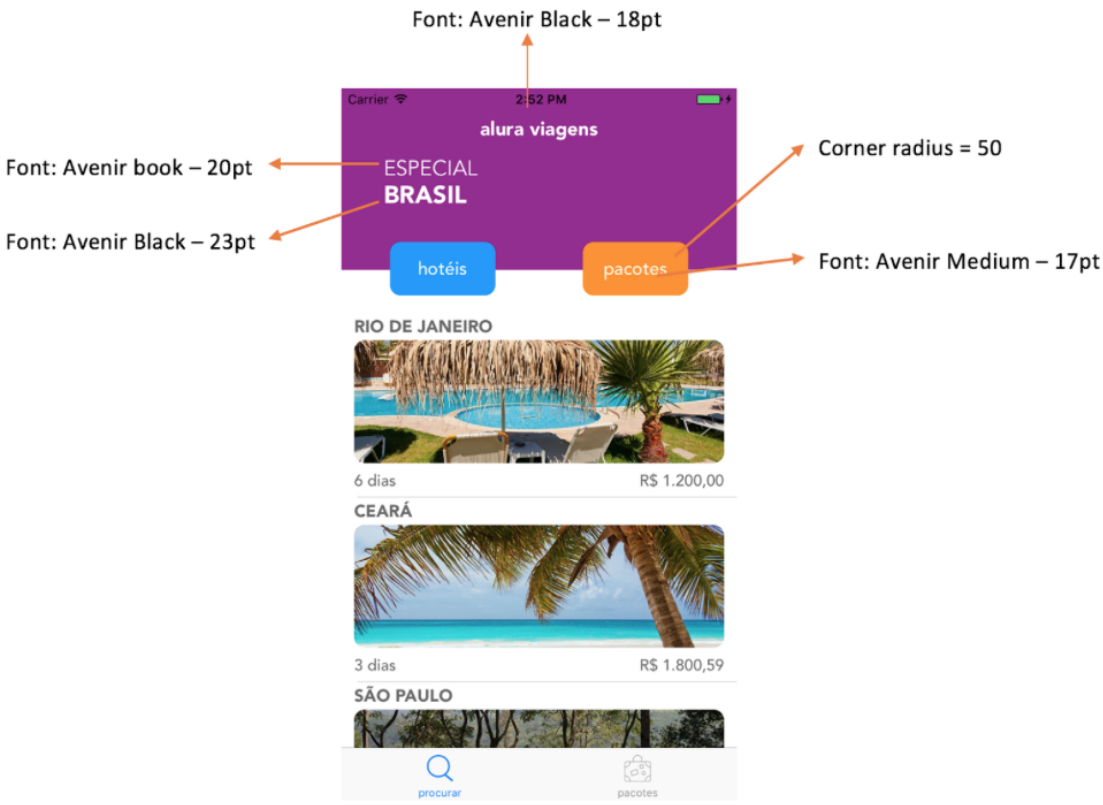
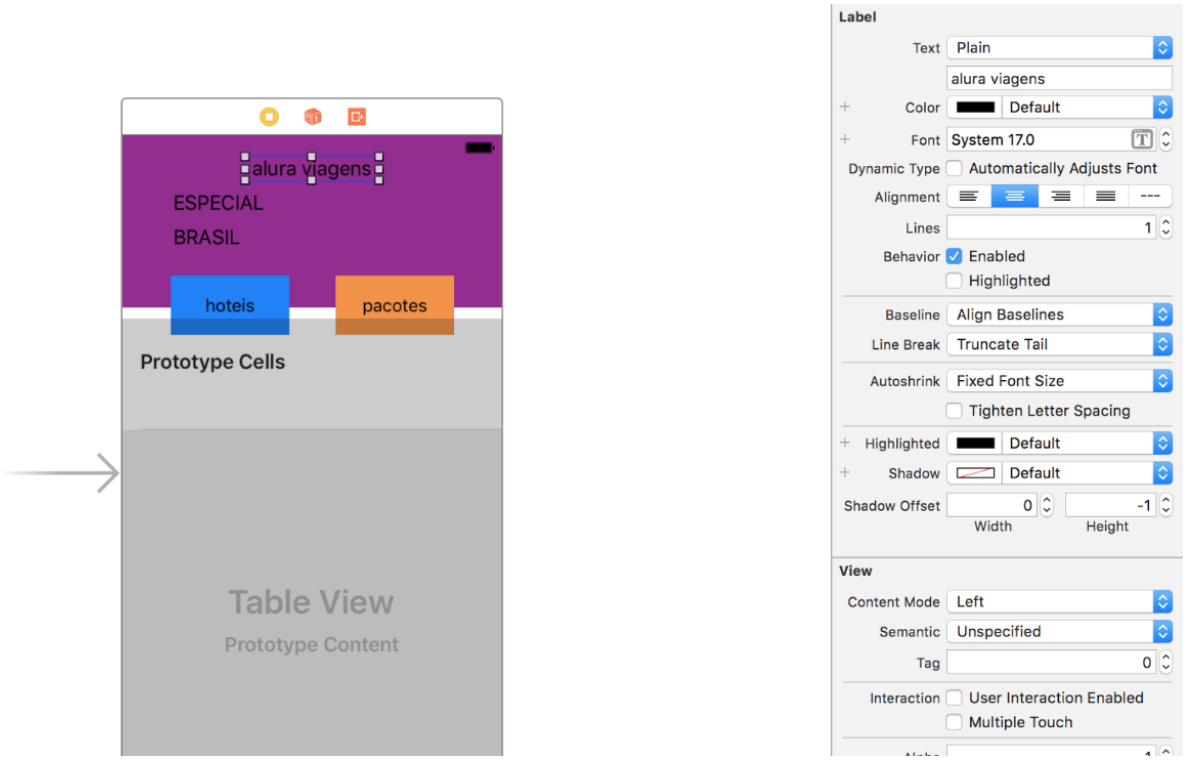
05

Listando os títulos da viagem na TableView

Acabamos de implementar a *Table View* com um array de valores fixos, mas o layout ainda está bem diferente do proposto. Vamos consultar o gabarito de design e para deixar exatamente como foi solicitado.

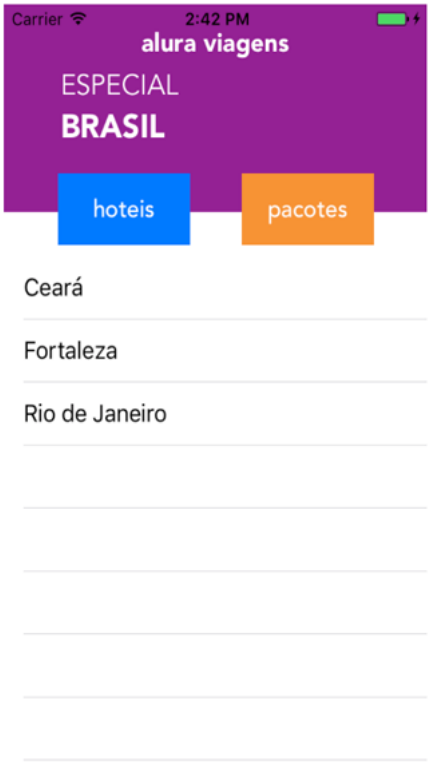


Abriremos o *storyboard* para padronizar as *labels*, primeiro com aquela que possui o nome do nosso app. Repare que ao clicarmos em cima da *Label*, o menu "Attributes Inspector" (lado direito) se modifica, trazendo as propriedades do objeto selecionado:



Uma das primeiras propriedades é a "Color", que mudaremos para branco. Embaixo, dentre as opções de fonte, escolheremos a `Avenir` com estilo `Black` e tamanho de `18` pt.

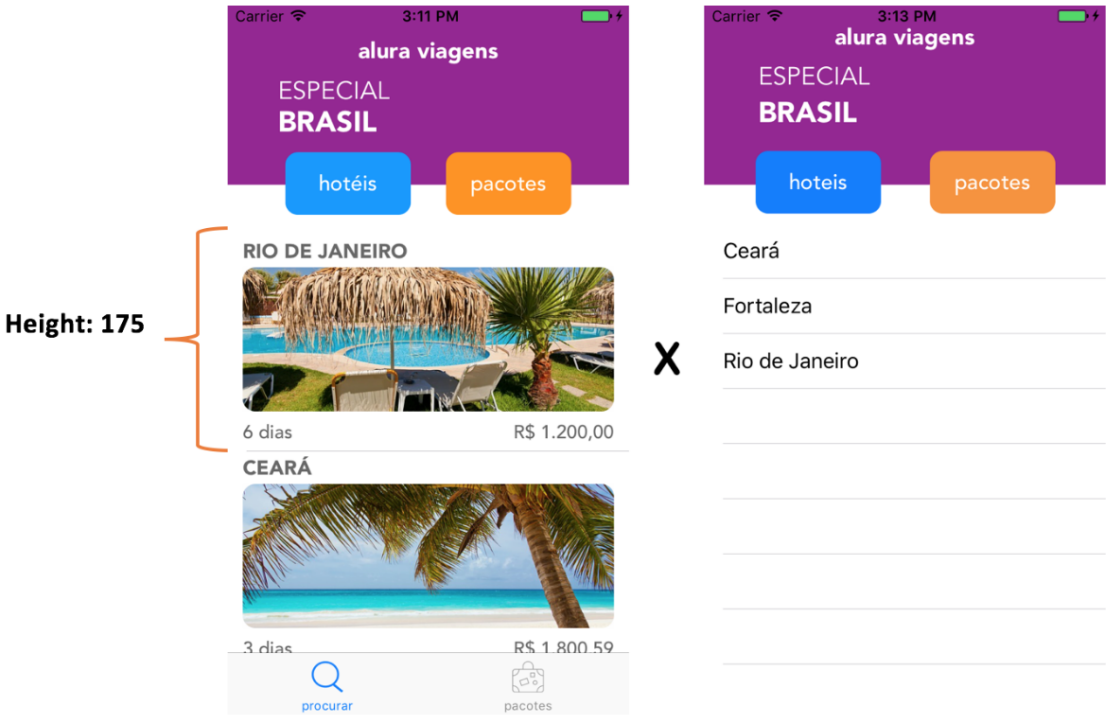
Por fim, faremos um *build* do app para conferir as alterações:



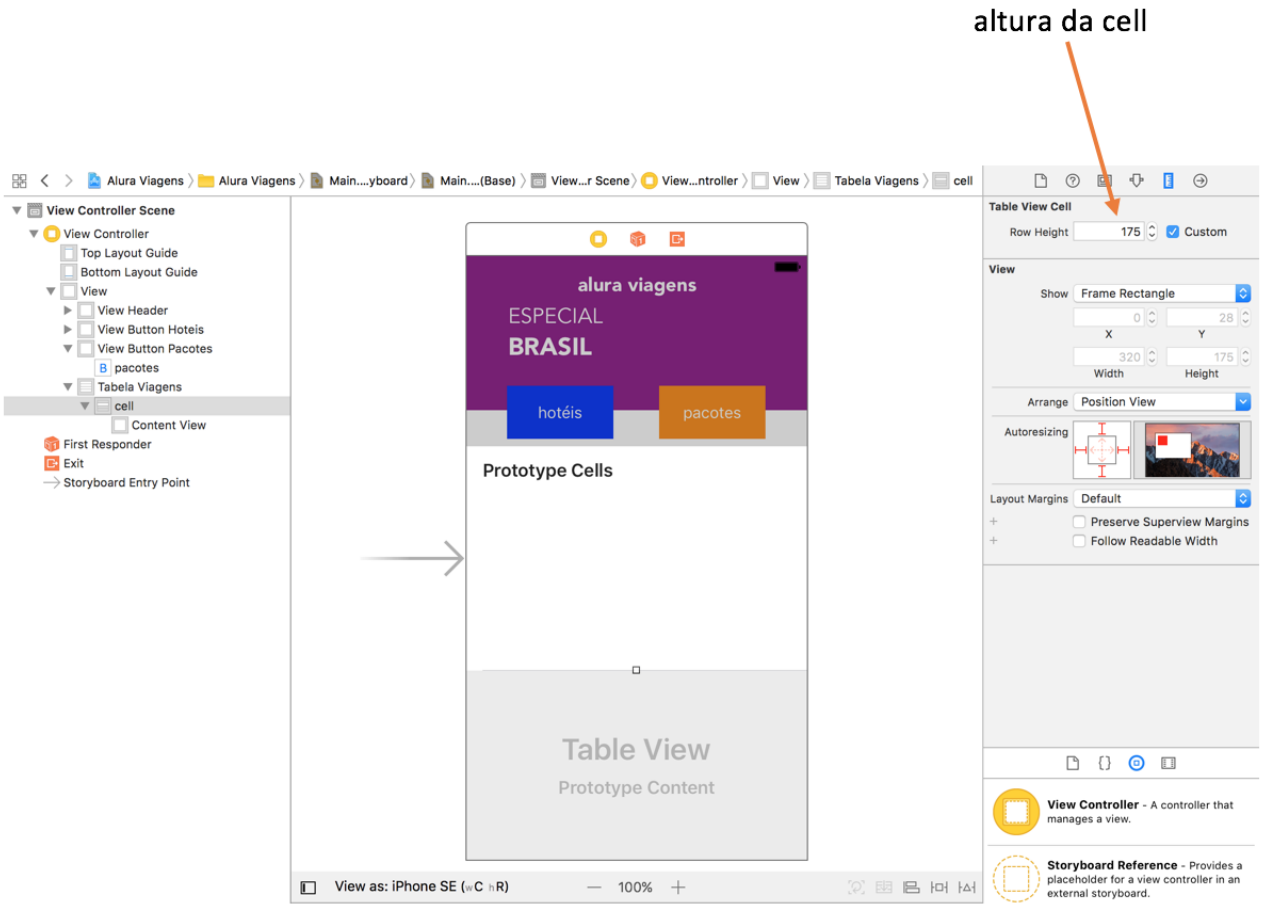
Agora vamos arredondar as bordas das *Views* "hoteis" e "pacotes", com o seguinte trecho de código:

```
override func viewDidLoad() {
    super.viewDidLoad()
    self.tabelaViagens.dataSource = self
    self.viewPacotes.layer.cornerRadius = 10
    self.viewHoteis.layer.cornerRadius = 10
}
```

Comparando o que já implementamos até agora com o design proposto, há uma grande diferença entre as alturas das células. No design que nos foi passado a *cell* tem altura de `175` e contém vários elementos. Então abriremos novamente o *storyboard* para continuarmos as alterações.



Modificaremos a altura da célula:



Já alteramos a altura da célula pelo *storyboard*. Isso nos dá espaço suficiente para colocarmos os elementos de que precisamos. Porém, fazer esta alteração pelo *storyboard* não soluciona o problema, pois aumentamos a área de visualização para encaixar os elementos necessários.

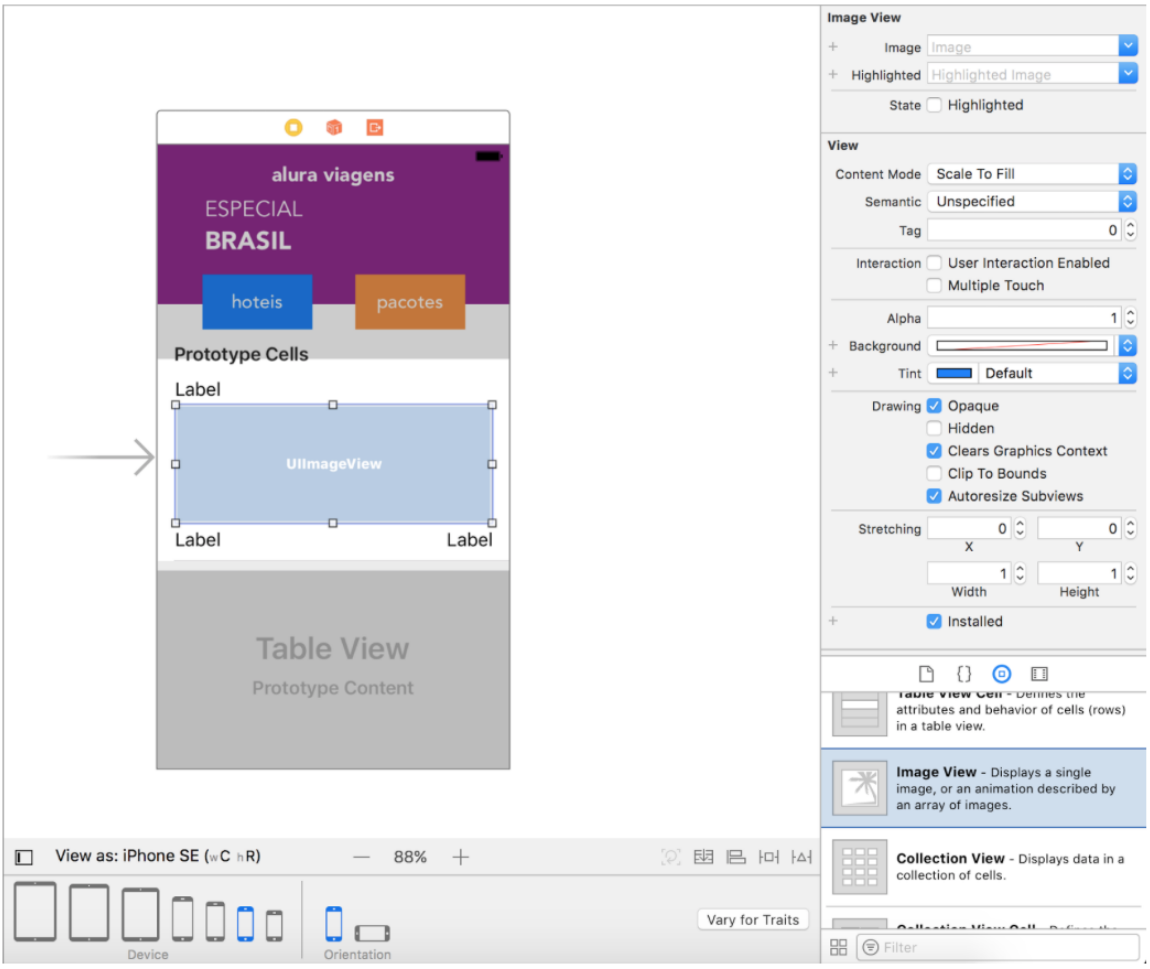
Para efetivamente aumentar a altura da célula, precisamos implementar o método de *delegate* `heightForRowAt` da *Table View*. Neste método, retornaremos o valor da altura da célula que queremos implementar, usando o protocolo de `UITableViewDelegate` no *View Controller*:

```
import UIKit

class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
    /*...*/
}
```

O protocolo implementado nos habilita a utilizar os métodos de *delegate* da *Table View*, portanto, implementaremos o método `heightForRowAt` :

```
func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
    return 175
}
```

Vamos continuar a implementação da *cell* acrescentando 3 *labels* e uma imagem:



Temos a implementação de um array com alguns nomes de destinos fixos para preencher temporariamente nossa tabela. Porém, é necessário refatorarmos nosso código para que a *Table View* não liste as *strings* fixas ("Ceará", "Fortaleza" e "Rio de Janeiro") do array, e sim, as viagens com título, preço, e imagem correspondente, assim como no gabarito.

Precisaremos criar algo que represente esses elementos da célula em nosso app. Como podemos fazer isso?

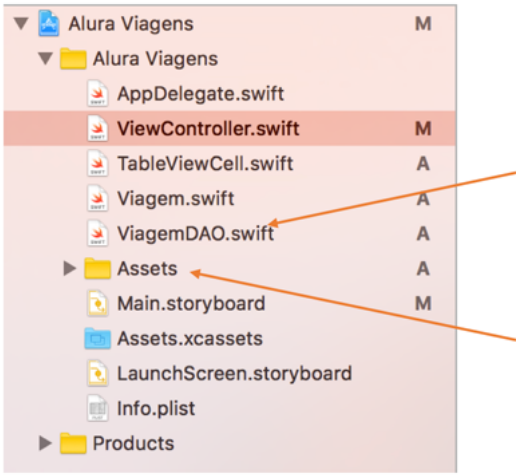
Podemos representá-los por meio de uma classe. Vamos criar a classe `Viagem`, com os seguintes atributos:

```
import UIKit

class Viagem: NSObject {
    let titulo:String
    let quantidadeDeDias:Int
    let preco:String
    let caminhoDaImagem:String

    int(titulo:String, quantidadeDeDias:Int, preco:String, caminhoDaImagem:String) {
        self.titulo = titulo
        self.quantidadeDeDias = quantidadeDeDias
        self.preco = preco
        self.caminhoDaImagem = caminhoDaImagem
    }
}
```

Com isso, temos uma estrutura que representa uma viagem em nosso aplicativo. Em seguida, iremos consumir as informações de viagem (nome, destino, data, preço) de algum lugar. Como queremos montar todo o layout do app, não acessaremos um webservice, e sim utilizar os arquivos DAO que trarão esses dados em tempo de execução.

Para isso, faremos o download (https://s3.amazonaws.com/caelum-online-public/iOS-Layout/Arquivos+Projeto.zip) dos arquivos `ViagemDAO.swift` e `Assets` do site e os colocaremos na pasta raiz do projeto.

O arquivo `ViagemDAO.swift` retornará uma lista de viagens com as informações que precisaremos exibir na *Table View*. A pasta "Assets" contém as imagens e ícones que utilizaremos no projeto. Agora, falta referenciarmos esses arquivos no projeto. Com *drag-and-drop*, clicaremos e arrastaremos os arquivos da pasta raiz para o Xcode.



Assim, podemos alterar os valores fixos do nosso array. Abriremos o *View Controller* e faremos a seguinte modificação:

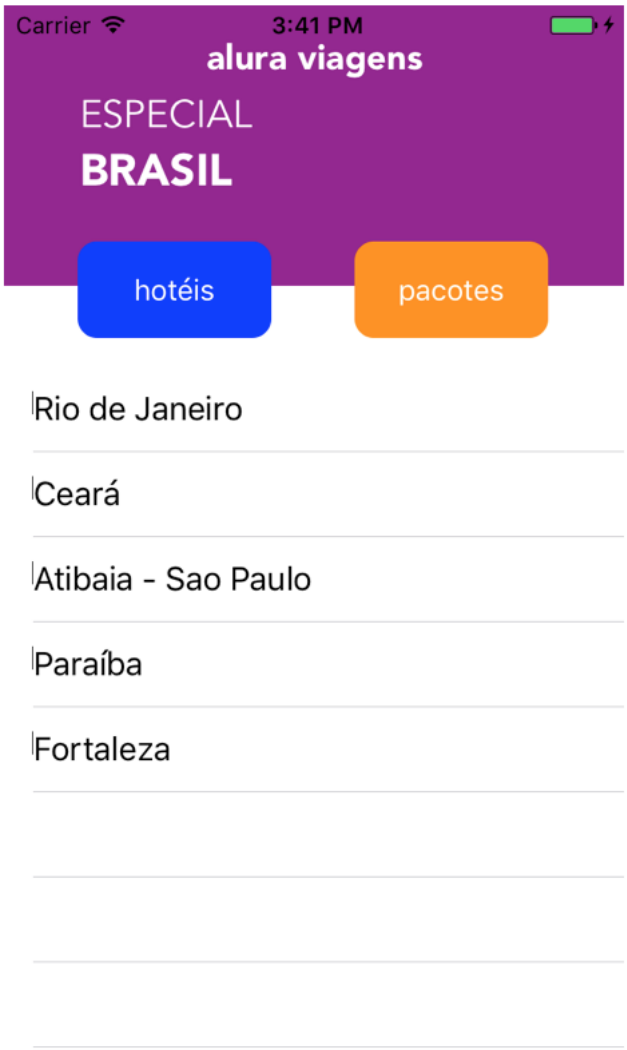
```
class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {

    @IBOutlet weak var tabelaViagens: UITableView!
    @IBOutlet weak var viewHoteis: UIView!
    @IBOutlet weak var viewPacotes: UIView!

    let listaViagens:Array<Viagem> = ViagemDAO().retornaTodasAsViagens()
    /*...*/
}
```

Repare que agora o nosso array está tipado com o objeto `Viagem`, que criamos. Ou seja, o método `retornaTodasAsViagens()` traz uma lista de objetos `Viagem` com todas as propriedades necessárias (título, imagem, preço, quantidade de dias) para preencher a célula da *Table View*.

Vamos fazer um teste refatorando o método `cellForRow()` para setarmos a *Label* da *cell* com o título das viagens do objeto que consumimos do arquivo `ViagemDAO`.

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath)
    let viagemAtual = listaViagens[indexPath.row]
    cell.textLabel?.text = viagemAtual.titulo

    return cell
}
```

E então faremos um *build* para conferir as alterações:



Ótimo! Desta vez a *Table View* está pegando os títulos corretos. A seguir, continuaremos customizando a sua célula.