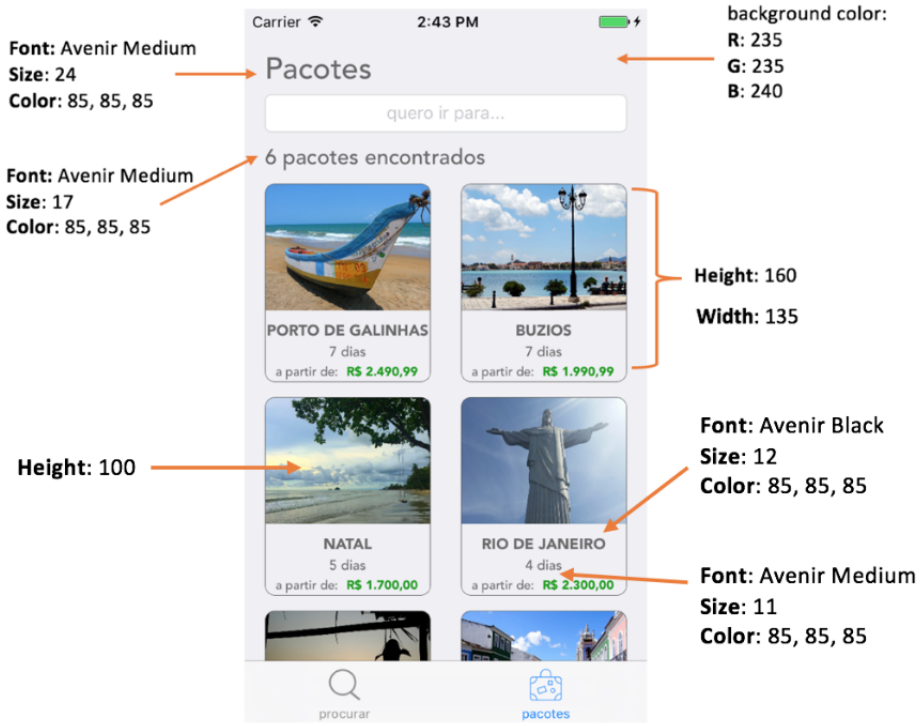
08

Finalizando a tela de pacotes de viagens

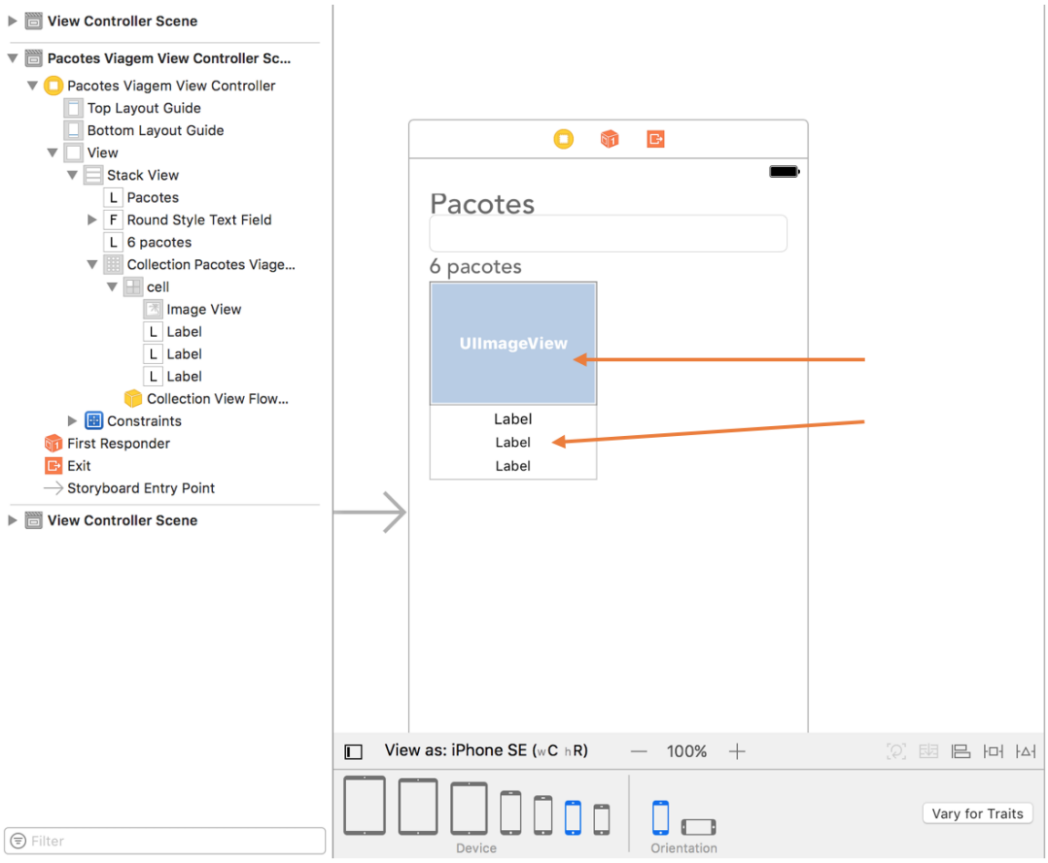
De volta ao projeto já temos a CollectionView funcionando. Agora precisamos fazer ela listar os pacotes de viagens.

Vamos analisar novamente o layout da tela de pacotes de viagens:

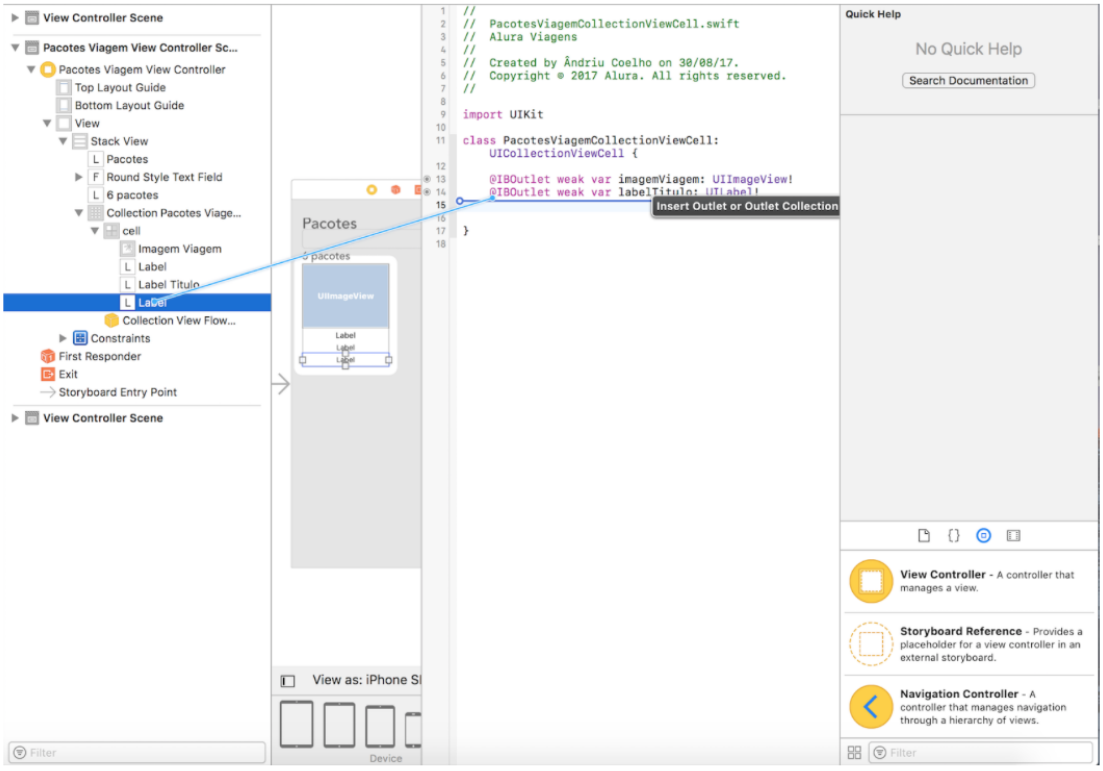


Acabamos de testar a implementação da CollectionView, colocando um fundo azul na célula. Porém agora é hora de customizarmos a célula da collection para que fique igual o layout proposto.

Então vamos voltar no storyboard e colocar os elementos necessários dentro da célula da collection:



Como precisamos acessar as labels e a imagem para setar o valor das viagens, precisamos criar os outlets desses elementos:



Para testar, vamos utilizar a mesma lista de viagens que utilizamos na primeira tela do nosso aplicativo.

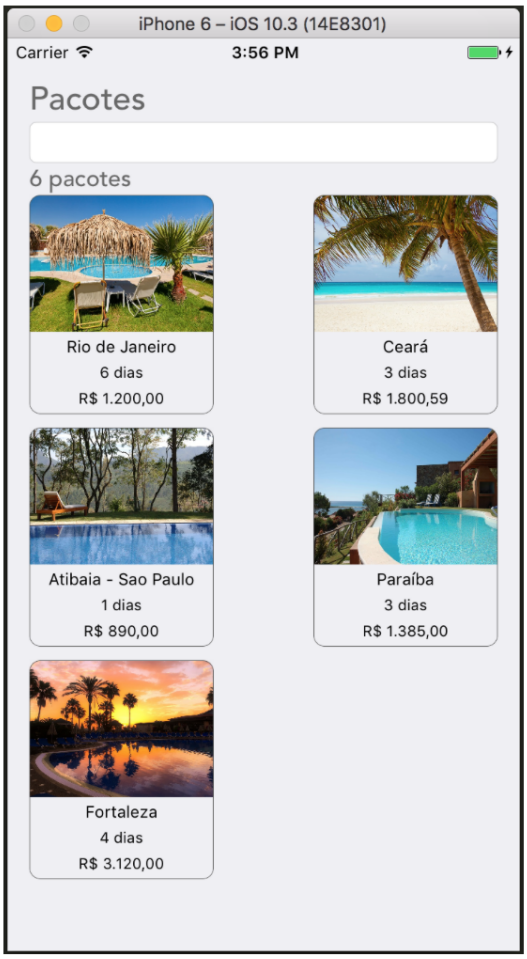
No método cellForItem, vamos setar os valores do objeto nos outlets da cell:

```swift
//
//  PacotesViagensViewController.swift
//  Alura Viagens
//
//  Created by Ândriu Coelho on 30/08/17.
//  Copyright © 2017 Alura. All rights reserved.
//

import UIKit

class PacotesViagensViewController: UIViewController, UICollectionViewDataSource {

    @IBOutlet weak var collectionPacotesViagens: UICollectionView!

    var listaViagens:Array<Viagem> = ViagemDAO().retornaTodasAsViagens()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.collectionPacotesViagens.dataSource = self
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    //MARK: - CollectionViewDataSource

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        return listaViagens.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) ->
        UICollectionViewCell {
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "cell", for: indexPath) as!
            PacotesViagemCollectionViewCell
        let viagemAtual = listaViagens[indexPath.row]

        cell.labelTitulo.text = viagemAtual.titulo
        cell.labelQuantidadeDias.text = "\(viagemAtual.quantidadeDeDias) dias"
        cell.labelPreco.text = "R$ \(viagemAtual.preco)"
        cell.imagemViagem.image = UIImage(named: viagemAtual.caminhoDaImagem)

        cell.layer.borderWidth = 0.5
        cell.layer.borderColor = UIColor(red: 85/255, green: 85/255, blue: 85/255, alpha: 1).cgColor
        cell.layer.cornerRadius = 8

        return cell
    }

}
```

Vamos rodar o app para conferir as alterações:

Agora é sua vez: Altere a cor de background do ViewController e da CollectionView conforme o gabarito.

Vamos rodar o app novamente:



O app já está com uma aparência bem melhor.

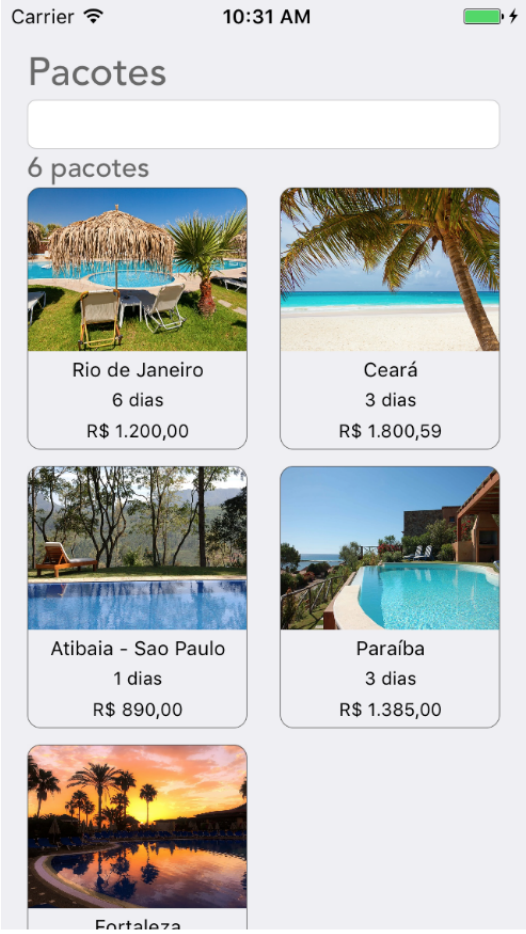
Se repararmos ainda há um espaço muito grande entre as células. Podemos melhorar isso implementando um método da collectionview que configura a altura e largura da célula.

Para utilizar esse método temos que implementar o protocolo `UICollectionViewDelegateFlowLayout`

No layout que recebemos, repare que cabe 2 células por linha na CollectionView. Então para aproveitar melhor o espaço entre as células podemos dividir a largura da collection por 2 e deixar um espaço para as margens. Assim conseguimos aproveitar mais os espaços entre as células.

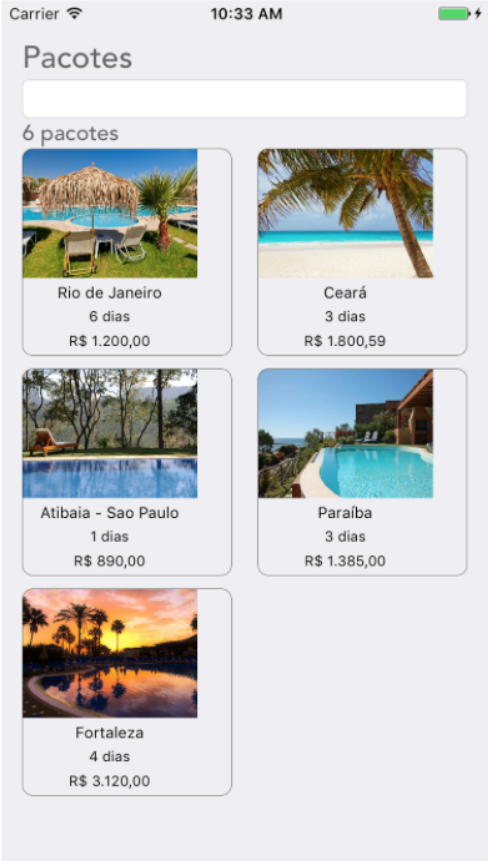
> Agora é sua vez: Implemente o método `sizeForItemAt indexPath` para que a collection renderize 2 células deixando um espaçamento menor entre elas

Vamos rodar o app novamente:



Ótimo, com essa implementação conseguimos aproveitar melhor o espaço entre as células.

Vamos fazer outro teste, rodando o app no simulador do iPhone 6:

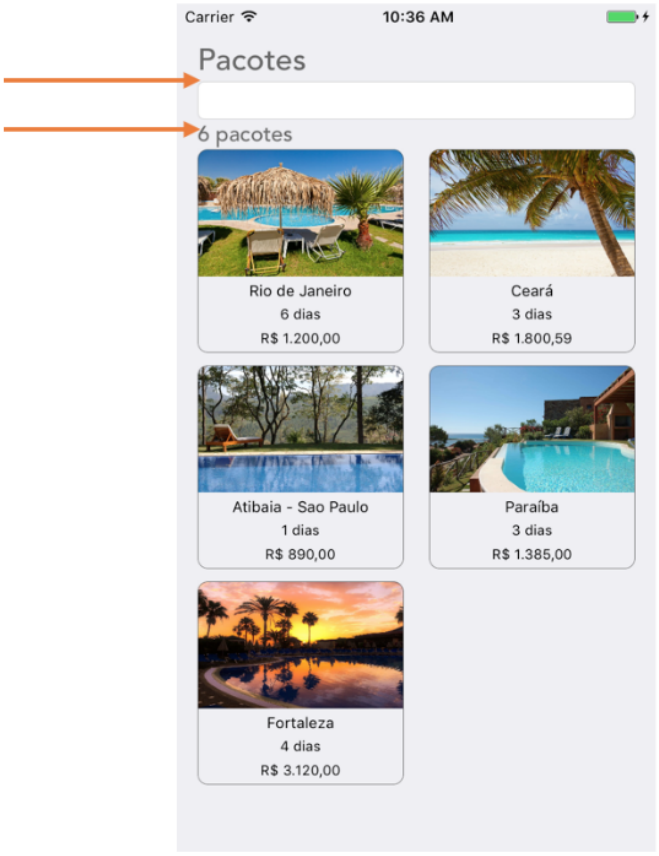


Repare que a imagem e as labels estão em uma posição fixa dentro da célula. Ou seja, se a célula aumentar ou diminuir de tamanho, os elementos não vão se reposicionar.

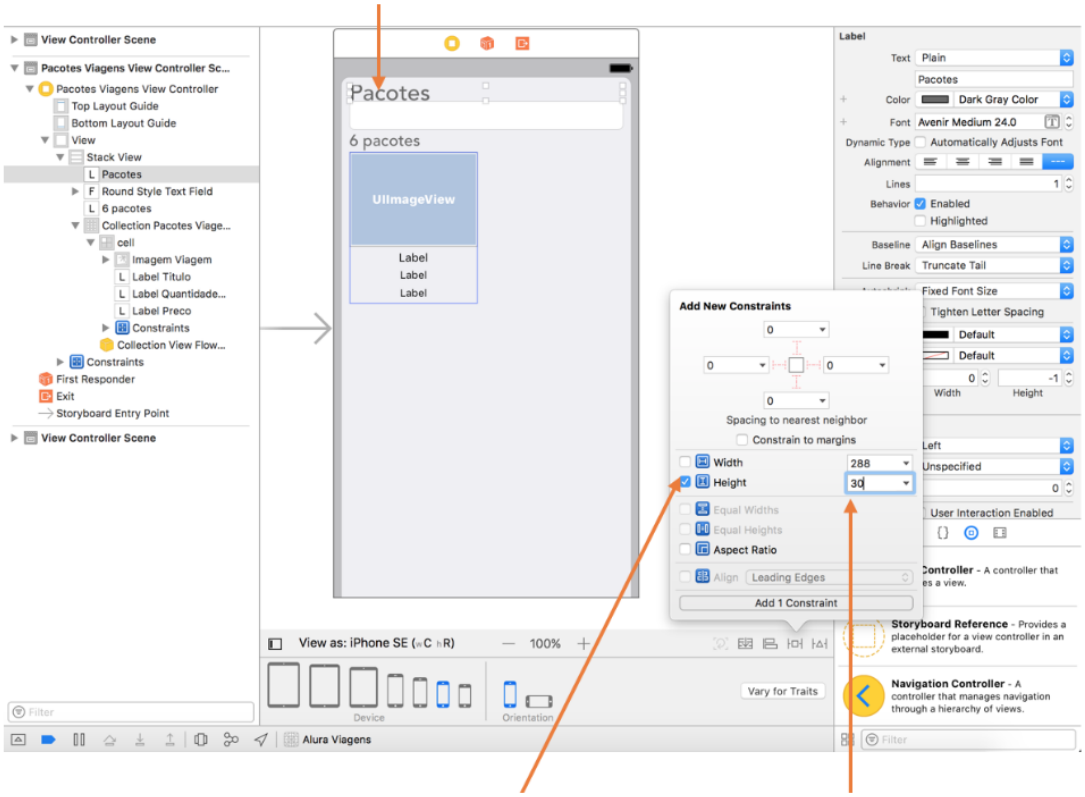
Agora é sua vez: Aplique as constraints necessárias nos elementos de dentro da célula

Vamos rodar o aplicativo no simulador do iPhone 6 novamente:

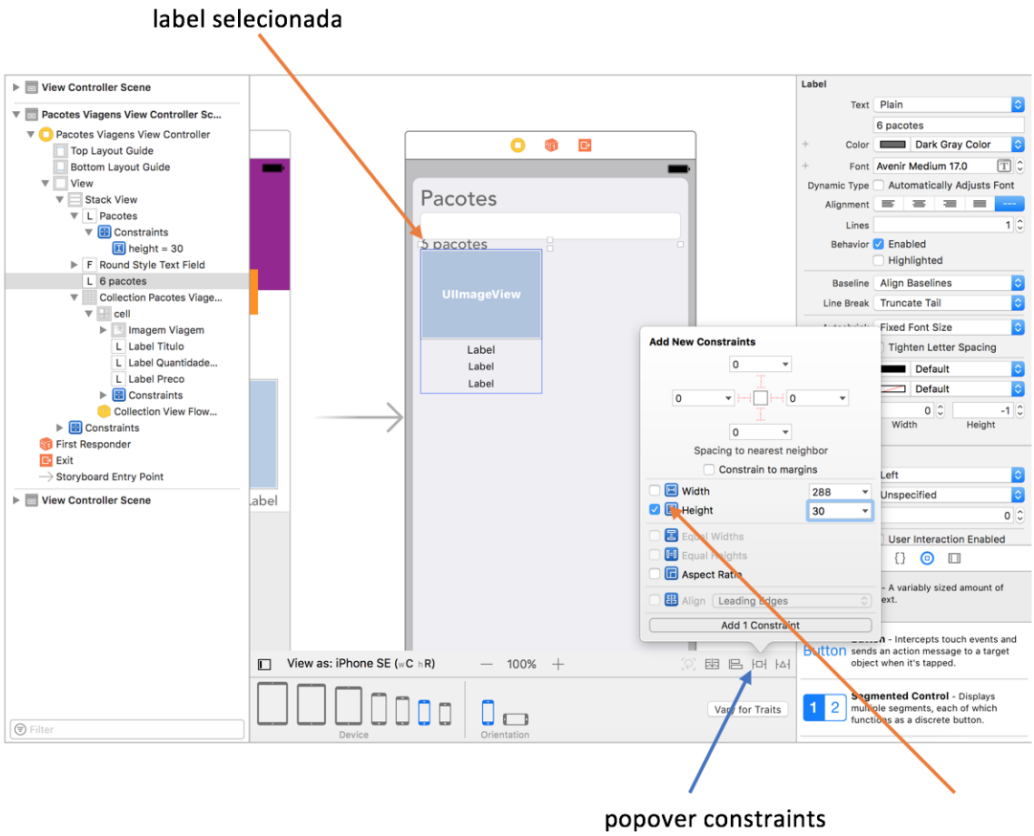
Para finalizar, vamos corrigir os espaçamentos entre os elementos. Perceba que eles estão bem grudados um no outro:



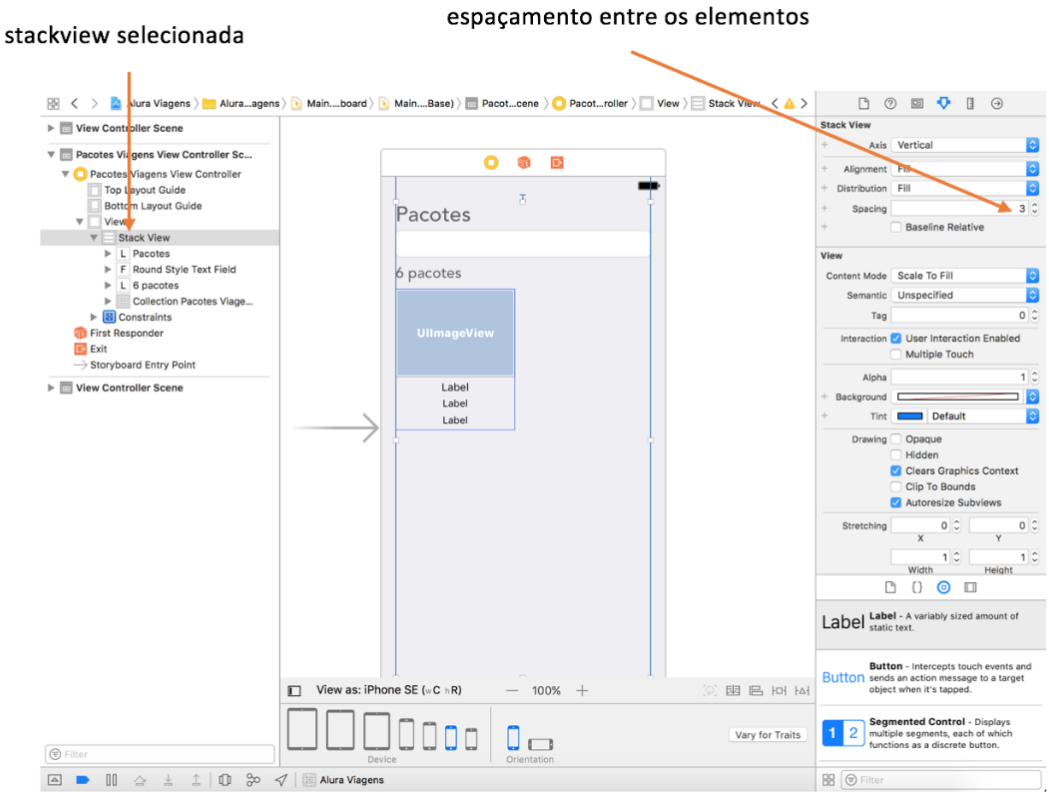
Para deixar um espaço entre a label e o elemento abaixo, vamos setar uma constraint de 30 de altura:



Para arrumar a altura da label de contador de pacotes, vamos setar uma constraint de altura:



Agora que colocamos uma restrição de altura mínima para as labels, podemos aumentar o espaçamento de todos os elementos de uma só vez, através da propriedade 'Spacing':



Ótimo, com isso terminamos a implementação da segunda tela de pacotes de viagens. Ainda tem algumas modificações que precisamos fazer como: alterar a label de contador de pacotes, que por enquanto deixamos fixo e implementar a lista com os pacotes de viagens, ao invés dessa que utilizamos na primeira tela.