02

Criando células customizadas

Transcrição

Acabamos de trocar o *Array* do projeto, que estava com valores fixos, pelo `ViagemDAO()`, arquivo que retorna as viagens para nós. No entanto, as células do nosso app ainda estão diferentes de como foi solicitado pela equipe de design. Mesmo definindo a altura de cada célula com `175`, nosso layout parece ter a mesma altura de antes.

Isso acontece porque, da mesma forma que alteramos o tamanho da célula marcando o *checkbox* de "Custom" no painel direito de *Table View Cell*, também precisamos passar um método de *Delegate* que retorne este valor. Assim, em `ViewController.swift` implementaremos o protocolo de *Delegate* após `UITableViewDataSource`, deixando a linha correspondente da seguinte forma:

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

Neste protocolo, há um método que define a altura que daremos a cada célula da nossa tabela, `heightForRowAt`:

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

Além disso, na célula temos uma imagem que ainda não está sendo usada, além das *labels*. Vamos criar um *Outlet* destes elementos! Todavia, por termos criado uma célula, o *View Controller* deixa de se responsabilizar por isso, ou seja, é necessário criar algo que cuide apenas da célula, imagens e *labels*.

Para isso, criaremos um `UITableViewCell.swift` clicando com o lado direito do mouse em "Alura Viagens" no painel de diretórios, e em "New File...". A subclasse será `UITableViewCell`, e a classe será denominada `TableViewCell`. Não usaremos os métodos `awakeFromNib()` e `setSelected()`, que vêm por padrão, portanto os deletaremos.

Criada a classe que gerenciará os elementos contidos na célula, setaremos isso no *storyboard* clicando em nossa célula e no terceiro ícone do menu superior no painel da direita, para configurarmos a customização da classe. Deixaremos "Class" com `TableViewCell`.

Então, precisaremos referenciar as *labels* inclusas na célula ao arquivo que a gerenciará. Vamos dividir a tela para facilitar nosso trabalho, e obtermos a visualização dos painéis com o *Controller View* e `ViewController.swift`. Em seu cabeçalho, é possível clicar em "Manual > Automatic (2)", e depois em "TableViewCell.swift".

Selecionaremos a primeira *Label* no painel de elementos, e a arrastaremos ao painel com o código para realizarmos o *Outlet*. O nome será `labelTitulo`. A segunda *Label*, da quantidade de dias do pacote de viagem, será arrastada ao código da mesma maneira, e chamaremos a conexão de `labelQuantidadeDias`. A última *Label* será `labelPreco` e, por fim, a imagem ficará como `imagemViagem`.

O código com as referências dos elementos ficará da seguinte maneira:

```
class TableViewCell: UITableViewCell {
    @IBOutlet weak var labelTitulo: UILabel!
    @IBOutlet weak var labelQuantidadeDias: UILabel!
```

```
    @IBOutlet weak var labelPreco: UILabel!
    @IBOutlet weak var imagemViagem: UIImageView!
}
```

Voltaremos a `ViewController.swift`, em que setamos apenas o título em uma *Label* que já vem por padrão no objeto `UITableViewCell`. Incluiremos `as! TableViewCell` no código para indicar que há uma classe para gerenciar a célula, logo após sua criação.

Já temos o nosso objeto, falta setarmos os valores. Deletaremos a linha `cell.textLabel?.text = viagemAtual.titulo`. Lembrando que por termos alterado a quantidade de dias para `Int`, é necessária uma **interpolação de *String***, isto é, colocaremos nosso valor e o concatenaremos com a palavra "dias".

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

    cell.labelTitulo.text = viagemAtual.titulo
    cell.labelQuantidadeDias.text = "\(viagemAtual.quantidadeDeDias) dias"
    cell.labelPreco.text = viagemAtual.preco

    return cell
}
```

Ao rodarmos o app, notaremos que a célula permanece pequena apesar da implementação do método `heightForRowAt`, e de termos alterado o valor da altura no *storyboard*. Isso ocorre porque da mesma forma que utilizamos os protocolos de `UITableViewDataSource`, no caso de `UITableViewDelegate` também precisamos passar os métodos que deverão ser implementados ao *Controller*.

Para isso, incluiremos a linha `self.tabelaViagens.delegate = self` em `viewDidLoad()`:

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

Feita a alteração, rodaremos a aplicação novamente e veremos que a célula preenche mais espaço do que antes, entretanto as *labels* ainda estão truncadas, muito pequenas. Se clicarmos duas vezes em `Main.storyboard`, poderemos aumentar o tamanho delas.

A caixa de texto se encontra exatamente do mesmo tamanho que o texto "Label", sendo que utilizaremos palavras maiores que, mantendo-se a atual configuração, não serão exibidas integralmente. Clicaremos e arrastaremos o limite direito da caixa de texto até a margem direita da imagem, `UIImageView`. Aumentaremos a largura da caixa de texto das demais *labels* também, e aproveitaremos para modificarmos o alinhamento da *Label* de preço para a direita.

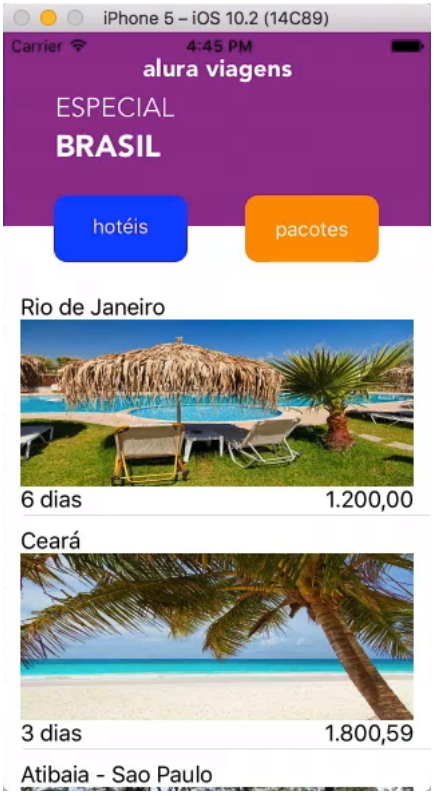
Falta incluirmos as imagens em nossas células, para o qual utilizaremos uma [pasta de imagens externa (https://s3.amazonaws.com/caelum-online-public/iOS-Layout/Arquivos+Projeto.zip)](https://s3.amazonaws.com/caelum-online-public/iOS-Layout/Arquivos+Projeto.zip) ("Assets"), a ser incluída no projeto.

Além disso, acrescentaremos a linha `cell.imagemViagem.image = UIImage(named: viagemAtual.caminhoDaImagem)` em `tableView()` do *Controller*.

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

Por utilizarmos `ViagemDAO().retornaTodasAsViagens()` , já temos o retorno dos objetos com as imagens desejadas.

Feito isso, rodaremos o app, e teremos o seguinte resultado no emulador:



Conseguimos implementar a tabela com todos os elementos necessários consumindo um arquivo DAO. A seguir, formataremos as *labels* e arredondaremos os cantos das imagens.