04

Implementando Table View

Transcrição

Acabamos de implementar o nosso *Header*, falta formatarmos alguns elementos. Vamos rodar nosso projeto para verificar seu andamento?

Se compararmos o que fizemos com o que precisamos entregar para o cliente, notaremos a falta da tabela com os destinos e preços de viagens, organizadas em células customizadas. Vamos trabalhar nisso!

Inicialmente, inseriremos uma *Table* (tabela), e em seguida uma *Cell* (célula), que possuirá elementos. Vamos por partes: buscaremos por "table view" no *Object Library*, em que clicaremos e arrastaremos para dentro do nosso *View Controller*. Deixaremos ele um pouco abaixo dos botões coloridos, e puxaremos suas margens laterais de modo que ocupe toda a largura do layout. Enquanto isso, a margem de base será clicada e arrastada até a base do layout.

Em seguida, incluiremos uma *Table View Cell*, clicando e arrastando-a ao *Controller*, seguindo a especificação do gabarito. Para customizarmos estas células, teremos que fazer a nossa tabela funcionar — na parte com os diretórios do projeto, clicaremos em `ViewController.swift` para abri-lo, e criaremos uma lista simples com algumas viagens.

Implementaremos uma constante denominada `listaViagens`, que será um `Array` de `String`. Iremos inicializá-la com alguns valores, `Rio de Janeiro`, `Ceará` e `São Paulo`. Além disso, precisaremos indicar à tabela quais elementos precisam ser renderizados. Portanto, implementaremos o protocolo `UITableViewDataSource`, que obrigatoriamente necessita de alguns métodos.

Neste caso, temos três destinos, ou três valores para a célula. Assim, o primeiro método a ser implementado é exatamente a quantidade de elementos a serem renderizados. Usaremos o método `numberOfRowsInSection`, que precisa do retorno de um **inteiro**, que será o número de elementos contidos no nosso `Array` (no caso, três).

Como já informamos à tabela quantos elementos deverão ser renderizados, precisaremos indicar a célula que deverá ser exibida. Usaremos o método `cellForRowAt indexPath`, que espera que devolvemos uma `UITableViewCell`. Criaremos a célula `cell`, que será acessado via `tableView`.

```
class ViewController: UIViewController, UITableViewDataSource {

    let listaViagens: Array<String> = ["Rio de Janeiro", "Ceará", "São Paulo"]

    // código omitido

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return listaViagens.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCe
        let cell = tableView.
    }
}
```

Porém, ao trabalharmos com uma célula customizada, precisamos de um **identificador**. Voltaremos ao *storyboard* `Main.storyboard`, selecionaremos `Table View Cell` na parte com as *Views* e, em "Identifier" do painel lateral direito,

pode-se colocar qualquer *String* a ser referenciada em nosso código. Neste caso, denominaremos simplesmente de `cell`.

Voltaremos ao *View Controller*; a `tableView` possui um método chamado `dequeueReusableCell()`, que acessa o *storyboard* e busca uma célula com o identificador que passarmos com `withIdentifier`. Já que acabamos de colocar um identificador chamado `cell`, é exatamente este que devemos incluir:

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath)

    return cell
}
```

Com estes métodos, nossa *Table View* funcionará bem, mas, ao rodarmos o aplicativo para verificarmos seu funcionamento, ela não nos exibe nada. Por que isso acontece, dado que já indicamos quantos elementos ela deve renderizar, e qual célula deve ser buscada no *storyboard*?

Nos esquecemos de incluir textos em *Labels*, isto é, os itens de `listaViagens`, cada vez que passarmos por um método. Usaremos, portanto, o `textLabel`, e atribuiremos os textos que temos nas nossas *Arrays*, na linha em que o método estiver passando, setando-o em nossas *Labels*.

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

    return cell
}
```

Agora, precisaremos indicar o responsável pela implementação destes métodos da nossa *Table View*: o *View Controller*. Ou seja, para que tudo isso funcione, precisaremos realizar um *Outlet* da *Table View*. Clicaremos em cima dela e no ícone com dois círculos sobrepostos ao lado do ícone de alinhamento esquerdo para dividir a tela em dois.

Deste modo, criamos um *Outlet* em cima da *Table View* clicando nela, mantendo "Cmd" ou "Ctrl" pressionado e arrastando o mouse para a linha que antecede `let listaViagens: Array<String> = ["Rio de Janeiro", "Ceará", "São Paulo"]`, no painel com o código. Precisaremos definir um nome para a *Table*, e usaremos `tabelaViagens`. Em seguida, clicaremos em "Connect".

Teremos o seguinte:

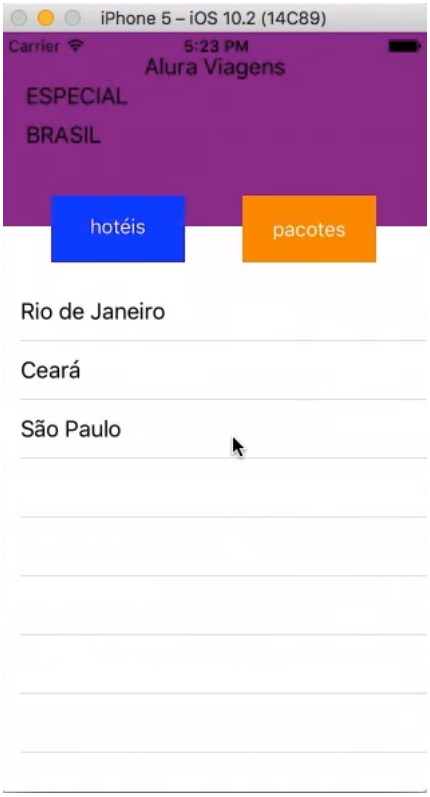
```
@IBOutlet weak var tabelaViagens: UITableView!

let listaViagens: Array<String> = ["Rio de Janeiro", "Ceará", "São Paulo"]
```

Voltaremos à tela com visualização única clicando no ícone com alinhamento esquerdo no menu superior do painel à direita, e em `ViewController.swift` no painel de diretórios. Agora que conseguimos referenciar nossa *Table View* no código, incluiremos uma linha em `viewDidLoad()` para que o `self` controle a tabela:

```
override func viewDidLoad() {
    super.viewDidLoad()
    self.tabelaViagens.dataSource = self
    // Do any additional setup after loading the view, typically from a nib.
}
```

Feito isso, rodaremos o app para conferir nossas alterações:



Estamos conseguindo renderizar algum conteúdo dentro da nossa *Table*. De acordo com o gabarito, falta customizarmos o design das células, o que faremos a seguir!