01

Listando o título das viagens

Transcrição

Nosso aplicativo está um pouco cru ainda, com formatação diferente do design do gabarito. Voltaremos a `Main.storyboard` para selecionar e alterar a *Label* com o nome do nosso app. Deixaremos o nome com letras todas em minúsculo ("alura viagens"). Faremos estas alterações no painel lateral direito com as configurações referentes aos textos. De acordo com o gabarito, suas especificações serão:

- Font `Custom`
- Family `Avenir`
- Style `Black`
- Size `18`

Se clicarmos na *Label* e a arrastarmos usando as setas do teclado, é exibida uma linha guia indicando o centro do layout. Isto nos ajudará a manter o texto centralizado.

Da mesma forma, "ESPECIAL", ficará com:

- Family `Avenir`
- Style `Book`
- Size `20`

Em "BRASIL", aplicaremos a mesma fonte das outras *labels*, com estilo `Black` e tamanho `23pt`. Estas duas *labels* devem ficar alinhadas à esquerda com o botão azul, de "hotéis".

Depois, selecionaremos as três *labels* mantendo-se a tecla "Shift" pressionada para alterar a cor ("Color") dos textos para branco (`White Color`). Tendo rodado o app para ver como ele está, arredondaremos os botões referenciando-as em nosso código. Criaremos *Outlets* acessando o *storyboard*, selecionando `View Button Hoteis` e clicando no ícone de duas circunferências se sobrepondo.

Mantendo "Cmd" (ou "Ctrl") pressionada, clicaremos em `View Button Hoteis` e arrastaremos o mouse até o *View Controller*. Poderemos nomeá-la de `viewHoteis`. Repetiremos o procedimento com a *View* de pacotes, `View Button Pacotes`, e o código ficará da seguinte forma:

```
class ViewController: UIViewController, UITableViewDataSource {

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

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

// código omitido
}
```

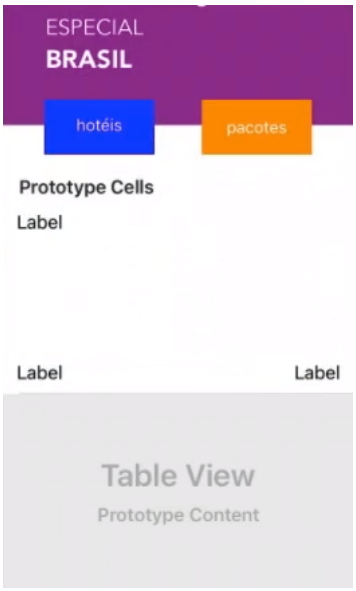
Então, abriremos `ViewController.swift` e, em `viewDidLoad()` chamaremos nossa *View* com `self.viewPacotes.layer.cornerRadius` para modificarmos o raio das *Views*, que deverá ser `10`, segundo o gabarito.

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

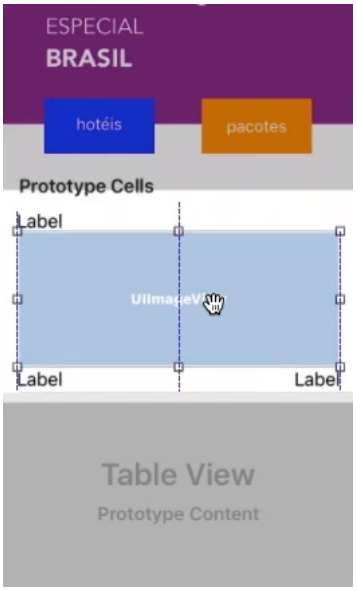
Vamos rodar o app novamente e conferir se tudo funciona como esperado?

Passando à tabela, nossas células estão mostrando apenas o título de cada viagem, sendo que precisamos incluir outros elementos. Voltaremos ao *storyboard* e usaremos a célula que incluímos anteriormente em nossa tabela. Ela deverá ter uma altura de `175`, que definiremos clicando no ícone de régua no painel direito, marcando o *checkbox* de "Custom".

Assim, temos mais espaço para incluir elementos. Precisaremos de uma *Label* com uma imagem ilustrativa da cidade de destino de viagem, de `125` de altura, e outras, uma para cada destino, indicando preço e quantidade de dias. Arrastaremos uma *Label* e faremos duas cópias dela, ficando uma no topo da célula, e duas na base, uma no canto direito e a outra no esquerdo:



Falta acrescentarmos as imagens, posicionadas com as mesmas margens das *labels* correspondentes, e com altura de `125`. Buscaremos por "image" no *Object Library*, clicaremos e arrastaremos a *Image View* para o layout no painel principal. Definiremos a sua altura (*Height*) no painel direito, enquanto a largura (*Weight*) seguirá o alinhamento das *labels* que acabamos de incluir:



Com isso, temos todos os elementos necessários para customizar a célula. E então precisaremos indicar ao app que queremos que a *cell* renderize uma viagem. Lembrando de alguns conceitos de Orientação a Objetos, é possível utilizar uma classe com todos estes atributos existentes na célula.

No painel com os nossos diretórios, clicaremos com o lado direito do mouse em "Alura Viagens", e em "New File...". Manteremos a configuração padrão na janela que se abre, com "Cocoa Touch Class", e pressionaremos "Next". Então, "Subclass of:" ficará com `NSObject`, e "Class" com `Viagem`. Clicaremos em "Next" e "Create".

Podemos clicar e arrastar o novo arquivo, `Viagem.swift` para cima, deixando-o junto aos demais de mesma extensão, por motivos de organização.

Nossas células são compostas por um título, uma imagem, seguida de uma *Label* que mostra a quantidade de dias e outra com o valor da viagem. Vamos criar estas variáveis em nossa classe. Quando formos lidar com a tabela, temos que indicar a localização e o nome da imagem utilizada, e por isto criaremos a variável `caminhoDaImagem`.

Além disso, implementaremos o método construtor e setaremos os valores:

```
import UIKit

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

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

Feito isso, vamos rodar o app para conferir a existência de possíveis erros de sintaxe.

Nossa tabela permanece aparentemente inalterada; apenas criamos uma estrutura que representará uma viagem dentro do app. Ao criarmos aplicativos, consumimos esses dados de algum lugar, normalmente de ***Web Services***, uma

das formas possíveis de consumir dados na web.

Como o foco do nosso curso não é este, e sim apenas montar o layout da aplicação, usaremos um arquivo que já vem com algumas viagens prontas, `ViagemDAO.swift` — disponível para download [neste link (https://s3.amazonaws.com/caelum-online-public/iOS-Layout/Arquivos+Projeto.zip)](https://s3.amazonaws.com/caelum-online-public/iOS-Layout/Arquivos+Projeto.zip) —, que selecionaremos e arrastaremos ao painel de diretórios do nosso projeto. Tal arquivo possui um método que retorna uma lista de viagens, e o usaremos para conseguirmos popular os dados do app.

Atualmente, no método `cellForRowAt` de `ViewController.swift` pegamos `listaViagens` com valores fixos, então teremos que alterar isso, e usar o arquivo que acabamos de incluir no projeto. Tipamos um `Array` de `Strings`, mas o colocaremos como sendo um `Array` de `Viagem`, chamando `ViagemDAO` com `retornaTodasAsViagens()`, que traz exatamente a lista de que precisamos:

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

Para fazermos um teste, criaremos uma variável neste mesmo arquivo para que se pegue a viagem que o método está retornando no momento, isto é, `viagemAtual`. Trocaremos `listaViagens[indexPath.row]` para `viagemAtual.titulo`, e clicaremos no ícone de *play*.

```
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
}
```

Isso resulta em um erro no nosso projeto, pois a quantidade de dias está como `String`, sendo que o esperado é `int` ("Cannot convert value of type 'Int' to expected argument type 'String'"). Teremos que alterar isso em nosso modelo, `Viagem.swift`:

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

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

Feito isso, rodaremos a aplicação novamente, com sucesso. Desta vez não lidamos com uma lista fixa que colocamos anteriormente. Falta começarmos a customizar a nossa célula, e faremos isso adiante.