Adaptive Interfaces

Tutorial 1 - Size Classes

- Abra o projeto **Size Classes**
- Descompacte e veja o conteúdo da pasta **Size Classe Images**
- Copie a imagem *ipad-portrait.png* para o XCAssets do projeto
- Renomeie a imagem para **Device**
- No *storyboard* a imagem deve aparecer
- Volte ao arquivo **xcassets**
- Selecione a imagem **Device**
- No *Attribute Inspector* (do lado direito da tela) procure **Width class** e **High Class** altere o valor para *Any & Compact* (para ambas)
- A imagem agora deve ter mais "*slots*" nomeados "*Any Width Any Height*, "*Any Width Compact Height*", "*Compact Width Any Height*" e "*Compact Width Compact Height*". O recurso efetivamente usado será o mais específico possível para uma *size class*.
- Vamos incluir como "*Compact Width Compact Height*" a imagem *iphone-landscape.png*.
- Vamos incluir como "*Compact Width Any Height*" a imagem *iphone-portrait.png*.
- Caso a classe da tela seja "*Compact* x *Compact*" a imagem usada será o iPhone em modo landscape. Para a classe "*Compact* x Qualquer outra classe (no caso apenas *Regular*)" a imagem utilizada será o iPhone em modo portrait.
- Da mesma forma, a imagem do iPad é utilizada para as classes "*Regular* x *Compact*" e "*Regular* x *Regular*". Vamos alterar a imagem "*Any* x *Compact*" de forma que para qualquer classe de largura não definida com altura "*Compact*" a utilize. Como já temos uma imagem para "*Compact* x *Compact*", efetivamente a única classe que utilizará esta imagem será "*Regular* x *Compact*". A imagem que representa esta classe é o iPhone plus em modo *landscape*.
- Vamos alterar a imagem "Any Width x Compact Height" para *iphoneplus-landscape.png*.

Com isto, temos uma imagem diferente para cada uma das size classes. Se executarmos esta aplicação para diversos dispositivos e alternarmos a orientação ou se utilizarmos o *Preview* (selecione o *storyboard*, abra o *Assistant Editor* e selecione *Preview*) ou mudando a visualização do *storyboard* para outro dispositivo - utilizando o modo "*View As:*" na parte inferior da tela) veremos que a imagem se alterna de acordo com o *Size Class*.

Isto mostra que podemos utilizar recursos diferentes dependendo da tela que utilizamos. O mesmo pode ser feito para telas retina com imagens distintas para 1x, 2x e 3x. Isto permite o uso de imagens de maior ou menor resolução para telas distintas.

Tutorial 2 - Trait Collection and Size Change

No tutorial anterior temos um problema. As mudanças de size classes servem como indicativo de grandes mudanças visuais (como de uma tela de iPhone para uma tela de iPad). Porém, no caso de um iPad a mudança não é drástica da orientação *portrait* para a *landscape*, porém algum ajuste visual pode ser necessário. No nosso caso, gostaríamos de colocar a imagem do iPad em modo *landscape* e *portrait* como fizemos com os iPhones.

Um primeiro método para detectar a rotação do dispositivo é *traitCollectionDidChange*:

```
override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollecti
on?) {
        super.traitCollectionDidChange(previousTraitCollection)

        print("Change from \(previousTraitCollection) to \(traitCollection)")
}
```

Este método permite executar algo quando a *traitCollection* é modificada. Porém, se executarmos este código para um iPad veremos que ele executa apenas da primeira vez, quando a *traitCollection* muda de **nil** para *Regular* x *Regular*.

Para o caso específico do iPad podemos utilizar o método *viewWillTransition* que captura a transição entre telas (ou da mesma tela, quando existe uma animação):

```
override func viewWillTransition(to size: CGSize, with coordinator: UIViewControl
lerTransitionCoordinator) {
        super.viewWillTransition(to: size, with: coordinator)

        if (self.traitCollection.userInterfaceIdiom == .pad) {
            if (size.width > size.height) {
                deviceImage.image = UIImage(named: "Device-landscape")
            } else {
                deviceImage.image = UIImage(named: "Device")
            }
        }
}
```

Vamos agora incluir a imagem *ipad-landscape.png* no XCassets e renomeá-la como "Device-landscape". Agora podemos executar esta aplicação para este tipo de dispositivo e ver a imagem alterando conforme rotacionamos o dispositivo.

Tutorial 3 - Adaptive Interface

Abra o projeto **AdaptiveInterface**. Iremos trabalhar sobre este projeto (um catálogo de cervejas) que se adapte a diferentes tipos de tela.

Passo 1 - Instalar e desinstalar *views*

Queremos que a imagem de *thumbnail* apareça apenas nos iPhones e que a imagem grande apareça apenas no iPad, onde temos espaço suficiente para exibir uma imagem maior.

Selecione o *storyboard* e na tela **Beer Detail** selecione a imagem **Thumbnail**. No *Attributes Inspector* (do lado direito) vá até o final e veja que existe um *check box* selecionado com o nome *Installed*. Isto indica que esta *view* está instalada para todas as *size classes*. Clique no botão "+" a esquerda e no diálogo "*Introduce Variation Based On:* " selecione *Width* e *Height* como *Regular*. Clique no botão *Add Variation* e um novo *check box* para esta *size class* será criada. Remova a seleção deste *check box*, com isto tal imagem será "desinstalada" da *size class Regular* x *Regular* (ou seja, não será exibida em um iPad).

Faremos o contrário para a imagem **Large Image**. Selecione-a, crie uma nova variação para *Regular* x *Regular*, mantenha esta instalada e desinstale a opção "*Installed*" sem nenhuma *size class*. Isto fará com que esta imagem esteja definida apenas para iPad em qualquer das orientações.

Vamos executar a aplicação (o código está preparado para tratar a inexistência de uma das imagens) e ver que os resultados são distintos para iPhone e iPad.

Passo 2 - Criar *constraints* específicas por *size class*

Thumbnail

Esta imagem, após o passo anterior, está definida apenas para iPhones (ou seja, para os *class sizes Compact* x *Compact*, *Compact* x *Regular* e *Regular* x *Compact*).

Vamos posicionar esta imagem com um espaçamento do canto superior esquerdo (20 pontos em relação ao *Top Layout Guide* e ao canto esquerdo da tela), para todas as 3 classes. Crie estas duas *constraints* e deixe-as habilitadas. Renomeie estas *constraints* para **Thumbnail Top** e **Thumbnail Leading** no *Document Outline* (Navegação das telas a esquerda do storyboard).

Queremos que esta imagem seja quadrada (para depois fazermos dela uma imagem circular). Crie uma constraint de *Aspect Ratio* 1:1 instalada para todas as orientações também. Renomeie a *con straint* para **Thumbnail Aspect**

Façamos com que a imagem ocupe 20% da largura ou altura (a menor das dimensões). Vamos criar uma constraint de *Proportional Width* e *Proportional Height* em relação a view pai com multiplicador 0.2 (Considerando Dimensão_Imagem = 0.2 Dimensão da Tela). Renomeie-as para **Thumbnail Width** e **Thumbanil Height**.

Agora, de forma similar ao que fizemos com as imagens, desabilite a constraint de *Proportional Height* para a classe *Compact* x *Regular* e deixe a de *Proportional Width* disponível apenas para ela. Para ver as opções de instalação e desinstalação dê dois cliques sobre a *constraint* no *Size Inspector* (o ícone da régua do lado direito da tela).

Ao executar a aplicação agora você verá o posicionamento da *view* de acordo com as restrições que demos para cada orientação.

Sumário

- **Thumbnail Top** (20/1) - Installed
- **Thumbnail Leading** (20/1) - Installed
- **Thumbnail Aspect** (20/1) - Installed
- **Thumbnail Width** (0/.2) - Installed (*unchecked*) / wC hR (*checked*)
- **Thumbnail Height** (0/.2) - Installed / wC hR (*unchecked*)

Título

O *label* de título será exibido para todas as orientações, logo não precisamos desisntalá-lo. Porém, seu posicionamento será diferente entre os dispositivos.

Vamos criar uma *constraint* de *Top* entre este campo e o **Thumbnail** que renomearemos para **Title Align Top**.

Vamos criar também uma *constraint* alinhando o *Trailing* da imagem com o *Leading* do *label* (mantendo uma constante de 20 pontos de espaçamento). Isto pode ser feito posicionando o label e criando uma constraint de *Horizontal Spacing* ou criando uma *constraint* e editando as propriedades *First Item* e *Second Item* nos detalhes da *constraint*. Vamos renomeá-la para **Title-Thumbnail Spacing**.

Vamos criar agora *constraints* entre o label e o *Top Layout Guiding* e a entre este e a parte esquerda da tela. Vamos renomeá-los para **Title Top** e **Title Leading**. Ambos com valor constante 20.

Agora vamos desabilitar as duas primeira constantes para a classe *Regular* x *Regular* e manter as duas últimas apenas para ela.

Ao executarmos agora a aplicação teremos o título posicionado corretamente tanto nos iPhones quanto nos iPads.

Sumário

- **Title Align Top** (0/1) - Installed (*checked*) / wR hR (*unchecked*)
- **Title-Thumbnail Spacing** (20/1) - Installed (*checked*) / wR hR (*unchecked*)
- **Title Top** - (20/1) - Installed (*unchecked*) / wR hR (*checked*)
- **Title Leading** - (20/1) - Installed (*unchecked*) / wR hR (*checked*)

Imagem Grande

Esta imagem só será exibida nos iPads. Selecione "*View as: iPad*" para poder selecionar a view e criar suas constraints.

Adicione constraints de centralização horizontal e vertical no container e as renomeie para **Image Center - Horizontal** e **Image Center - Vertical**.

Adicione mais uma constraints para que a imagem tenha 50% (0.5) da largura da tela. Renomeia a *constraint* para **Image Width**. Crie também uma *constraint* de Aspect Ratio 1:1 para manter a imagem quadrada. Renomeie esta para **Image Aspect**.

Sumário

- **Image Center - Horizontal** (0/1) - Installed
- **Image Center - Vertical** (0/1) - Installed
- **Image Width** - (0.5/1) - Installed
- **Image Aspect** - (0/1) - Installed

Ícone do tipo de cerveja

Vamos alinhar este ícone com o topo e chamar estas *constraint* de **Icon Top**. Vamos criar constraints de altura e largura com valor 44 e chamá-las **Icon Width** e **Icon Height**.

Vamos criar uma *constrait* para que este ícone tenha 20 pontos de distância do *Trailing* da superview e chamá-la **Icon Trailing**.

Por fim, criemos uma *constraint* de 20 pontos de espaçamento entre o ícone e o título e chamá-la **Title-Icon Spacing**.

Todas estas constraints estarão habilitadas em todas as classes.

Sumário

- **Icon Top** - (0/1) - Installed
- **Icon Width** - (44/1) - Installed
- **Icon Height** - (44/1) - Installed
- **Icon Trailing** - (20/1) - Installed
- **Title-Icon Spacing** - (20/1) - Installed

Texto

Este texto será apresentado em todas as orientações, como o título. Desta forma não precisamos desinstalá-lo. Porém, seu posiconamento será distinto entre iPhones e iPads.

Vamos alinhar o leading deste texto com as duas imagens, criando duas constraints. Vamos chamá-las de **Text Align Leading - Thumbnail** e **Text Align Leading - Image**.

Vamos alinhar o trailing com a imagem grande e chamar esta *constraint* de **Text Align Trailing**.

Vamos criar uma *constraint* de espaço entre este texto e a parte direita da tela, com valor 20 e chamá-la **Text Trailing**. Esta *constraint* deve ser desinstalada para a classe *Regular* x *Regular*.

Agora criemos uma *constraint* de espaço para o *Bottom Layout Guide* de 20. Chamaremos esta *constraint* de **Text Bottom**.

Por fim, criemos *constraints* de espaço vertical entre o texto e as duas imagens. Vamos chamá-las **Text-Thumbnail Vertical Space** e **text-Image Vertical Space**.

Note que como as *constraints* em relação aos componentes só existem se o componente estiver instalado não houve necessidade de desinstalar as *constraints* associadas às imagens.

Com isto temos a nossa tela construída.

Sumário

- **Text Align Leading - Thumbnail** - (0/1) - Installed
- **Text Align Leading - Image** - (0/1) - Installed
- **Text Align Trailing** - (0/1) - Installed(*checked*) wR hR (*unchecked*)
- **Text-Thumbnail Vertical Space** - (20/1) - Installed
- **Text-Image Vertical Space** - (20/1) - Installed
- **Text Bottom** - (20/1) - Installed

Tutorial 4 - Split View

Veja que esta app está organizada sobre uma **Split View**. Este componente se divide em duas partes: um *master view controller* e um *detail view controller*. Tal organização permite navegar entre itens no elemento *master* e ver detalhamento de suas informações na *view controller* de detalhes.

Vale notar também a *segue* entre a table view e a tela de detalhes, chamada de **showBeer**. Esta segue é utilizada para trazer as informações de uma cerveja selecionada na *view controller master* e apresentá-la. Veja que esta *segue* é do tipo **Show Detail (e.g. Replace)*.

Se executarmos esta aplicação em um iPad ou em um iPhone Plus em modo *landscape*, a view *master* será apresentada a esquerda e os detalhes a direita. Ao selecionar um item, a view da direita é substituida e populada.

No iPhone Plus em modo *portrait* ou em um iPhone em qualquer orientação, o par *master-detail* se comporta como uma *segue* do tipo *Push*, apresentando uma barra de navegação que permite a volta (que pode ser feita com um *swipe* a esquerda também).

O componente **Split View** é um exemplo de componente que se apresenta de forma distinta em diversas Size Classes, sendo um exemplo e elemento importante para interfaces adaptáveis.

Tutorial 5 - Popover

Vamos utilizar um outro componente adaptável: o *popover*, uma janela que ocupa parcialmente a tela.

Para isto, vamos criar uma *segue* entre a tela **BeerDetailViewController** e a **BeerTypesViewController** do tipo "*Present as Popover*" e nomeá-la "**beerTypes**".

No *Attributes Inspector*, onde podemos definir o identificador da *segue*, existem outras propriedades. Uma delas é **anchor**, o componente sobre a qual o diálogo do popover vai ser ancorado. No campo de texto deste campo existe um círculo (que se torma um + quando passamos o cursor sobre ele). Clique sobre ele e arraste para a imagem Icon da tela **BeerDetailViewController**. Esta será nossa âncora.

Vamos disparar esta *segue* quando clicarmos no ícone. Altere o método *viewDidLoad* para que ele trate o gesto de *tap* sobre a imagem:

```
override func viewDidLoad() {
        super.viewDidLoad()

        let tap = UITapGestureRecognizer(target: self, action: #selector(BeerDeta
ilViewController.beerTypes(_:)))
        beerIcon.addGestureRecognizer(tap)

}
```

Este código vai disparar o método *beerTypes* (que ainda não criamos) quando um *tap* for detectado sobre o ícone. Vamos adicionar este método também:

```
func beerTypes (_ tap:UITapGestureRecognizer) {
        performSegue(withIdentifier: "beerTypes", sender: self)
}
```

Teste a aplicação em um iPad e um iPhone. Veja que no iPad o componente abre como um pequeno diálogo sobre o ícone (que fecha ao clicarmos fora do componente), enquanto no iPhone ele abre como uma janela modal.

Podemos alterar este componente para que ele tenha o mesmo comportamento no iPhone e no iPad. Vamos fazer com que a classe **BeerDetailViewController** implemente o protocolo **UIPopoverPresentationControllerDelegate**, que customiza o comportamento do popover:

```
class BeerDetailViewController: UIViewController, UIPopoverPresentationController
Delegate {
```

E precisamos definir que este view controller será o *delegate* do popover que será aberto quando formos disparar a *segue*:

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

        //TODO: Set popover presentation controller delegate
        segue.destination.popoverPresentationController!.delegate = self

}
```

Agora podemos definir o comportamento do popover incluindo o seguinte método:

```
func adaptivePresentationStyle(for controller: UIPresentationController, traitCol
lection: UITraitCollection) -> UIModalPresentationStyle {
        return .none
}
```

Este método faz com que o popover não seja aberto como um *view controller* modal. Com isto conseguimos replicar o comportamento do *popover* no iPad em um iPhone!