# Livro do Educador

## Desenvolvimento de Aplicativos Móveis

### iOS

# Unidade 4

## Aula 1 – View Controllers

Um aspecto importante em iOS é oferecer uma navegação dinâmica e fluida ao usuário, para isso precisamos entender o fluxo entre nossas Views e a hierarquia relacionada entre suas subclasses.

### CARGA HORÁRIA

Conforme o plano de aula, esta aula terá duração de 1h30 e deverá ser conduzida de acordo com as orientações pedagógicas.

### OBJETIVO DA AULA

Ao final da aula, você deverá garantir que o aluno tenha subsídios para

* Compreender as funções das Views Controller;
* Compreender a relação hierárquica entre a View Controller e suas subviews;
* Compreender o ciclo de vida dos Views Controllers;
* Relacionar o fluxo de navegação com a hierarquia das Views;
* Compreender e praticar o uso do UINavigationController;
* Compreender e praticar o uso do UITabBarController;
* Fazer uso de Navigation Controllers.

### ORIENTAÇÕES PEDAGÓGICAS

Para atender os objetivos de aprendizagem, você deverá conduzir o processo de ensino considerando a organização didática apresentada a seguir:

* 40 minutos de aula expositiva;
* 10 minutos para tirar as dúvidas dos alunos;
* 40 minutos para desenvolver as atividades propostas para a turma e apresentação da TDP.

### TÓPICOS DE ESTUDO

Todos os tópicos a seguir, conforme livro do aluno, devem ser trabalhados de forma dinâmica, criativa, com embasamento teórico e prático voltado ao mercado de trabalho.

* Entendendo a hierarquia das Views;
* Compreendendo View Controller;
* UINavigationController;
* UITabBarController.

### PONTOS IMPORTANTES

Fazer uso dos fluxograma irá ajudar bastante o aluno a compreender as relações entre as Views e os métodos usados em seu ciclo de vida.

### Entendendo a hierarquia das Views

Educador, aqui vamos iniciar um passo necessário para criação de apps mais dinâmicos e complexos. O aluno deve aprender como funcionam as Views e as Views Controllers para que possa ter domínio e certa autonomia no desenvolvimento de apps iOS. Como já visto, o iOS presa pela usabilidade, ou seja, preza para que o usuário tenha a melhor experiência de uso em seus aplicativos, com isso se faz necessário ter bastante compreensão das relações hierárquicas entre as Views.

Para explicar vamos iniciar dizendo que:

A View é representada pela classe UIView (UI-Interface do Usuário), ela é representada por um retângulo.

O aluno pode querer afirmar que existem Views de todas as formas geométricas, como por exemplo o botão Play, de forma triangular. Na realidade não é assim. Explique ao aluno que no exemplo do botão Play, assim como em todos os botões e ícones, a View tem forma retangular e nela há uma imagem, com um Triangulo e fundo de mesma cor que a View da camada de trás, ou é uma imagem sem fundo, dando assim a impressão de que a View, ou o botão, tenha a área apenas do triangulo, quando na verdade continua tendo área retangular.

A UIView é a classe mais alta, ou primitiva, das Views e possui outras classes inferiores, como UIButton, UIImageView, UIControl. Estas subclasses podem ser colocadas nas Views, formando assim uma estrutura de camada de Views e é isso que vamos ver a partir de agora.

#### 1.1.1. Visão Geral

A UIView também fazem parte do grupo UIKit, que é o conjunto principal de classes que fazem interação com o usuário do app.

Educador, explique as principais responsabilidades da View. Algumas delas é a manipulação de layout, onde cada View define seu próprio comportamento de redimensionamento em relação à superview de acordo com a forma de redimensionamento que ela foi baseada, como explicado na aula passada. Também uma responsabilidade importante é a manipulação de eventos onde a View pode responder a certo evento, como toques, e não menos importante, as VIews apresentam elementos gráficos de desenhos e animações.

Tudo isso é possível pela capacidade que uma View tem de se incorporar com outras, criando hierarquias visuais sofisticadas.

Explique também as relações entre as Views incorporadas e as incorporadoras, ou seja, superviews e subviews. Um tipo de View que possui subviews mas não possui nenhuma View acima dela é a View Controller que será explicada em seguida.

### 1.2. Compreendendo a View Controller

Educador, explicando o que é uma View Controller, basicamente é uma View da classe UIViewController, que é uma superview que gerencia um conjunto de views que compõem uma parte da interface do usuário no app. Ou seja, ele administra diversas atividades de um grupo de Views, como as interações entre elas, carrega-las e eliminá-las e outras coisas.

#### 1.2.1. O ciclo de vida de uma View Controller

O ciclo de vida dos Views Controllers é gerido por métodos que são chamados em momentos adequados, como em transições e mudanças de estado delas. Estes métodos da View Controller são sempre herdados por uma subclasse e permite que adicionemos o comportamento desejável para cada um desses métodos. Esta estrutura está organizada no padrão Template Method como vimos em Android.

Educador, explique ao aluno que para configurar certos comportamentos de views ou eliminá-los, é necessário utilizar o método adequado para cada configuração de view a ser feita. Mostre a Figura 1 e explique o ciclo de vida de uma View Controller.

Alguns exemplos de métodos podem ser chamados pela UIViewController, o **viewDidLoad()**, cria e carrega o conteúdo principal da View Controller; **viewWillAppear()**, destina-se a operações quando a View irá tornar-se visível (análogo a este há o método **viewWillDisapear()**); o **viewDidAppear()**, destinado a qualquer operação que ocorra logo depois da View se tornar visível (análogo a este há o método **viewDidDisapear()**).

Com isso, o aluno consegue compreender que as views do app são controladas por Views controladoras, superiores a elas, herdando dessas, métodos que podem gerenciar o ciclo de vida de suas subviews e definir operações de acordo com cada etapa deste ciclo.

### 1.3. UINavigationController

Agora vamos explicar um View controller específico. O UINavigationController.

Explique ao aluno, que esta é uma View que gerencia a navegação entre conteúdos. É possível utilizá-la como ela está ou podemos acrescentar subclasses a ela para oferecer-lhe mais funcionalidades.

Educador, dê como exemplo o menu de configurações do iPhone, mostre que como na Figura, o fluxo de navegação entre as views se compara à organização hierárquica das views. Conforme selecionamos as configurações em novas telas, mais inferiores são as views em na hierarquia.

#### 1.3.1. Definindo a navegação do app

Abrir o material de apoio **Unidade\_4\_-\_Aula\_1\_-\_Exemplo\_1.zip** para que o aluno possa aprender com adicionar a navegação entre duas telas.

* Abrir o **Main.storyboard** e para criar uma cena **Navegar**, arrastar uma UIViewController da paleta de componentes até o storyboard, inserir uma UIImageView na cena até então vazia e escolher a imagem **1. Navegar** na paleta de propriedades. Ajustar as restrições da Imagem View conforme o passo 5 e explicar como é possível fazer com que a Navigation Bar se esconda.
* Selecionar o ícone amarelo no topo desta cena e embarcar esta View Controller em um Navigation Controller acessando **Editor > Embed In > Navigation Controller.** Para personalizar a Navigation Controller, selecioná-la e marcar a opção **Initial View Controller**. Assim fazemos com que a Navigation seja a primeira tela que irá aparecer em nosso app.
* Selecionar a Navigation Bar no painel de navegação e mudar a cor da propriedade **Tint** para #FF2D55 (../../../../Desktop/Captura%20de%20Tela%202016-11-22%20às%2022.33). Em seguida, dar duplo clique na cena Navegar, na Navigation Bar e escrever **Navegar**.
* Para adicionar um botão para exibir a cena Música, adicionar um botão, como feito na Unidade 3, sem título, da forma apresentada no passo 14, mas ele deve ser de cor transparente. Adicione as restrições ao botão como no passo 15.
* Para fazer a ligação do botão à cena Música, faça como na Unidade 3, segurando a tecla **Control** e arrastando o botão para a cena Música. Na caixa aberta, escolher a opção Show. Agora execute o app e faça os testes.

### 1.4. UITabBarController

Outra View Controller é a UITabBarController é um controlador de navegação de abas ou guias, àquelas que aparecem na parte inferior, ou às vezes superior, dos apps de iOS e é possível usá-lo como está ou fazer nossas próprias personalizações.

Educador, explique como um Tab Bar Controller está associado à uma View Controller para cada aba, e como isso faz com que seja possível que as Tab Bar Controller apresentem tipos diferentes de informações ou informações idênticas utilizando estilos diferentes sem que as Views Controller de cada aba estejam associadas entre si. Utilize a Figura como exemplo e vamos a prática.

#### 1.4.1. Criando uma navegação em abas

Abrir o arquivo de partida **Unidade\_4\_-\_Aula\_1\_-\_Exemplo\_2.zip** e observar como está o Main.storyboard.

Adicionar uma UITabBarController no topo do Storyboard e remover as duas Views Controllers que vieram atreladas a ela.

Fazer uma conexão da Tab Bar Controller para cada Navigation Controller, escolhendo a opção **view controller** para cada uma das conexões e configurar a Tab Bar Controller como sendo a view controller inicial em **Is Initial View Controller.**

Para adicionar ícones e alterar a cor das tabs, entrar em **MainViewController.swift** e no método **viewDidLoad()** adicionar o código como visto no passo 1, dando as devidas explicações. A partir disso, selecionar a TabBarController e mudar a classe dela para **MainViewController** no painel de propriedade. Agora temos uma Tab Bar com imagens e cores personalizadas.

### 1.5. Resumo

O aluno aprendeu o que são **Views Controllers** e entendeu sobre a sua **hierarquia** e **fluxo de navegação**. Conheceu os **métodos** que regem o **ciclo de vida** das Views Controllers. Aprendeu a usar na prática a **UINavigationController** e a **UITabBarController** com a particularidade de cada uma.

## Aula 2

### 2. Listas com Table Views

Para que possamos apresentar dados na tela de nosso app com certa organização, como por exemplo em forma de lista, fazemos uso de tabelas e as Tables Views são as Views certas para este fim.

### CARGA HORÁRIA

Conforme o plano de aula, esta aula terá duração de 1h30 e deverá ser conduzida de acordo com as orientações pedagógicas.

### OBJETIVO DA AULA

Ao final da aula, você deverá garantir que o aluno tenha subsídios para

* Implmentar listas em seus apps;
* Localizar as Table Views dentro da hierarquia das Views Controllers;
* Fazer uso de métodos para definir o número de linhas, o tamanho, tipo de texto e o conteúdo delas assim como organizá-las em seções.

### ORIENTAÇÕES PEDAGÓGICAS

Para atender os objetivos de aprendizagem, você deverá conduzir o processo de ensino considerando a organização didática apresentada a seguir:

* 40 minutos de aula expositiva;
* 10 minutos para tirar as dúvidas dos alunos;
* 40 minutos para desenvolver as atividades propostas para a turma e apresentação da TDP.

### TÓPICOS DE ESTUDO

Todos os tópicos a seguir, conforme livro do aluno, devem ser trabalhados de forma dinâmica, criativa, com embasamento teórico e prático voltado ao mercado de trabalho.

* Table Views Contents;
* Table View;
* Implementando uma Table View.

### PONTOS IMPORTANTES

A parte prática e teórica desta aula é relativamente curta. Entregamos um ponto de partida do app com consideráveis mudanças, aproveite e tente explicar a forma como os arrays foram implementados ou também aproveite para alterar as strings com nomes e estilos de música da prefrência dos alunos.

### 2.1. Table View Controllers

Educador, a Table View Controler, representada pela classe UITableViewController é perfeita para se dar uma introdução para exibição de listas. Ela é responsável por apresentar na tela dados dinâmicos que podem variar ou simplesmente organizar o conteúdo do layout.

#### 2.1.1. Anatomia de uma Table View Controller

Educador, a Table View Controller foi feita de componentes configurados para facilitar o uso, e estes componentes se organizam da seguinte forma:

Use a Figura 1 podemos ver que a **UITableViewController** é uma classe que se estende da superclasse **UIViewController**, que é a classe mais alta na hierarquia das Views, como visto na unidade passada. A Table View Controller é usada para gerenciar as **Table Views** (listas) que por sua vez contém e gerencia as **Table Views Cell** (células das listas). Também é a Table View que implementa os **delegates** e **data sources** que explicaremos mais à frente.

### 2.2. Table View

É um componente da UITableViewController que se gerencia as listas do app em iOS. E para inserir dados em uma Table View, precisamos de uma Table View Cell, que é uma linha (row) da lista, mas antes é necessário escolher um layout para esta, que já vem com alguns layouts pré-definidos no UIKit. São layouts simples, para aplicações menos exigentes.

#### 2.2.1. Table View Cell

Educador, explicando melhor o que seria uma Table View Cell, é uma View pela qual podemos exibir alguma coisa na Table View. Nela é possível escolher um estilo de célula padrão, ou personalizado, podendo ser prototipado direto storyboard que aprenderemos na prática.

Também explique que para definir as configurações das células em Swift, é necessário implementar protocolos.

### 2.3. Implementando uma Table View Controller

Para implementar uma Table View Controller, abra o arquivo base **Unidade\_4\_-\_Aula\_2\_-\_Exemplo\_1.zip** no material de apoio e ver como está nossa tela **Navegar**.

* Permitir que o aluno formate as Labels e o estilo das células, tendo como base o passo 1.
* Selecionar a **Table View** e mudra a propriedade **Style** para **Grouped**, assim a lista ficará dividida em seções. Selecionar a **TableViewCell** e no campo Identifier e iserir o valor **UITableViewCellIdentifier**.
* No painel de navegação, abrir o arquivo **BrowseTableViewController**. Notar o array criado com os dados a serem exibidos na tabela. Explicar ao aluno a estrutura do array, que está separado em Section e Music.
* No método **numberOfSections(in:)**, implementar o código como no passo 8, para oferecer o número de elementos do array **items**.
* No método **tableView(numberOfRowInSection:)**, implementar o código como no passo 9, para definir o tanto de linhas que cada sessão terá de acordo com a quantidade de música que tem dentro de cada seção.
* No método **tableView(heigthForRowAt:)**, implementar o código como no passo 10, para que a altura de cada célula seja de 65 e também implementar o código como no passo 11, para definir o texto de cada seção.
* No método **tableView(cellForRowAt:)**, implementar o código como no passo 12. Com ele é possível obter a célula através de nosso identificador e obter para ela as músicas de cada célula do array de itens associados a respectiva seção.
* Por último, no método **tableView(didSelectRowAt:)**, adicionar o código como no passo 13, é o método que é chamado quando tocamos em alguma célula e chama a cena **Música**.

### 2.4. Resumo

Na aula, o aluno aprendeu na prática como implementar um Table View, assim como sua teoria, seu lugar na hierarquia das Views Controller e os cinco métodos necessários para que a Table View tenha um funcionamento correto.

## Aula 3

### 3. Animações básicas

Para que o usuário tenha uma experiência de uso confortante por prolongadas horas, nada melhor que animações de objetos durante a navegação, partindo desse pressuposto, iniciaremos a aula de hoje.

### CARGA HORÁRIA

Conforme o plano de aula, esta aula terá duração de 1h30 e deverá ser conduzida de acordo com as orientações pedagógicas.

### OBJETIVO DA AULA

Ao final da aula, você deverá garantir que o aluno tenha subsídios para

* Entender a importância das animações nas navegações no app;
* Identificar os frameworks que encontramos componentes para tratar de animações simples ou mais complexas;
* Utilizar métodos que gerem o ciclo de vida de animações.

### ORIENTAÇÕES PEDAGÓGICAS

Para atender os objetivos de aprendizagem, você deverá conduzir o processo de ensino considerando a organização didática apresentada a seguir:

* 40 minutos de aula expositiva;
* 10 minutos para tirar as dúvidas dos alunos;
* 40 minutos para desenvolver as atividades propostas para a turma e apresentação da TDP.

### TÓPICOS DE ESTUDO

Todos os tópicos a seguir, conforme livro do aluno, devem ser trabalhados de forma dinâmica, criativa, com embasamento teórico e prático voltado ao mercado de trabalho.

* Por que utilizar animações?;
* O que pode ser animado?;
* Como animar as Views.

### PONTOS IMPORTANTES

Nesta aula é importante que o aluno obtenha um sentido de obrigatoriedade em acrescentar animações em seu trabalho e conhecer toda estrutura de classe e ciclo de vida das animações.

### 3.1. Por que utilizar animações?

Mais uma vez ressaltamos que o usuário de iOS busca sempre uma navegação mais fluída, e as animações durante as transições tornam esta experiência de navegação muito mais prazerosa. Para qualquer mudança nas Views, ou entre as Views utiliza-se uma animação.

Explique que o uso de animação nas transições não é obrigatório mas é essencial para a interação do homem com o app, isto é comprovado por estudos. Caso o desenvolvedor não opte por usar animações em mudanças de seu app, o usuário sentirá certa estranheza pois, nada em nosso mundo que sofre mudança, sofre instantaneamente, e sim, passa por uma transição. Sendo assim, usando animação, o usuário utilizará o app de forma mais natural e fluida, e passará mais tempo utilizando o mesmo.

### 3.2. O que pode ser animado

Educador, explique um pouco sobre o UIKit e como ele surgiu da linguagem Obejective-C. Enquanto o UIKit é responsável pela interface gráfica, o Core Animation é o framework que possui tratamentos específicos e de mais baixo nível para manipulação, mas ambos suportam animações e o UIKit felizmente possui um conjunto básico de animações e será este que iremos nos deparar na maioria das vezes. Diversas propriedades das Views podem ser animadas, cite algumas delas e de suas funcionalidades ao aluno.

### 3.3. Como animar as Views

Podemos animar views de maneira simples com métodos da UIView, mas para animações mais complexas, o iOS 10 nos oferece a UIViewAnimating e UIViewPropertyAnimator. Vamos focar um pouco na UIViewAnimating.

#### 3.3.1. UIViewAnimating

Este é um protocolo do iOS 10 que define métodos a implementar por parte dos animadores. Esses métodos controlam a opção de iniciar, pausar e cancelar animações. Os Animadores são definidos pela classe UIViewPropertyAnimator, e começm no estado inativo São iniciados e pausados pelo método **startAnimation()** e **pauseAnimation()** respectivamente e cancelados com o método **stopAnimation()** conservando o estado que a animação deixou o objeto.

Educador explique sobre o ciclo de vida das animações segunda a Figura e também apresente e explique algumas propriedades delas.

### 3.4. Resumo

Nesta aula o aluno aprendeu um pouco sobre as animações, descobriu o UIKit e o Core Animation como frameworks que possuem componentes que tratam das animações e se adentrou um pouco mais entre os componentes de animação do UIKit, tendo o animador da classe UIViewPropertyAnimator e alguns métodos que gerem seu ciclo de vida.