# Livro do Educador

## Desenvolvimento de Aplicativos Móveis

### iOS

# Unidade 3

## Aula 1 – Storyboards

Educador, aqui veremos algo parecido com Storyboard em Java, com algumas diferenças. O aluno iniciará um primeiro projeto no Xcode e realizar prototipagem e com isso o aluno verá que a manipulação de layous no Xcode é muito mais simples que tudo que ele já viu parecido.

### 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

* Localizar e reconhecer as características do Storyboard;
* Criar um novo projeto;
* Inserir e utilizar imagens em um projeto;
* Fazer ligações entre telas de um app.

### 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.

* O que é Storyboard?
* Criando um novo projeto;
* Conhecendo a Storyboard;
* Prototipando com o Storyboard;

### PONTOS IMPORTANTES

Este será praticamente o primeiro contato prático do aluno com o Xcode. Naturalmente ele terá grande dificuldade em navegar pela tela e identificar certos elementos solicitado. Educador, preste toda ajuda possível e utilize o material para auxiliá-los.

### O que é um Storyboard

Educador, vamos começar relembrando de Storyboard em Java, que se assemelhava a uma história em quadrinhos onde se expunha um problema que o software irá resolver e como uma ferramenta de prototipagem. Diga o aluno, que em iOS, Storyboards se assemelha muito a isso mas possui suas diferenças.

Explique ao aluno o que é um Storyboard, dizendo que é uma representação visual da interface do usuário, como se fosse uma linha de fluxo que mostra a sequencia de telas e transição entre elas.

É interessante também fazer uma comparação com o Canvas do Android, mostrando que o Storyboard também é capaz de inserir mais de uma ela num app e definir op fluxo entre elas ligando pontos.

Por fim, de uma breve explicação sobre o Main.storyboard e o LaunchScreen.storyboard, que são criados juntamente com a criação de um novo projeto.

### 1.2. Criando um novo projeto

Nesta parte você, Educador, irá auxiliar o aluno a criar seu primeiro projeto no Xcode.

Basicamente o que deve ser feito é:

* Na tela de abertura do Xcode, clicar na opção **Create a new Xcode Project** e em seguida escolher um **template** para o novo projeto. Templates são modelos de app já pré-iniciados, que fazem com que economizemos tempo com elementos padrões que já são inseridos no início do projeto. Escolher a opção **Single View Aplication**, como mostrado no passo 3;
* Na tela de configuração de projetos temos configurações como Nome do Projeto, da Equipe, da Organização. Temos também o identificador de Organização e de Pacote, como vimos no Android Studio, e por fim, temos a escolha da linguagem de programação e os dispositivos destinados. Configurar de acordo com o passo 4.
* Para finalizar, escolher o local onde será salvo o projeto.

### 1.3. Conhecendo a Storyboard

Educador, agora vamos conhecer o **Main.storyboard**. Aqui, ajude o aluno a localizá-lo no painel de navegação. Explique, que é uma cena e a seta que indica o ponto de entrada da cena.

### 1.4. Prototipando com o Storyboard

Educador, a partir de agora iremos dar explicações a respeito da prototipagem utilizando um método prático. Iremos criar um aplicativo de músicas e a cada novo recurso utilizado, procure explicar ao aluno como funciona e dê exemplos de outras utilizações para eles.

#### 1.4.1. A pasta Assets

Educador, explique ao aluno que a pasta **Assets** é onde colocaremos todas as imagens do nosso projeto, bastando apenas arrastá-las para dentro da pasta. Ela pode ser encontrada no painel de navegação como **Assets.xcassets** .

* Para o nosso app musical, devemos abrir a pasta do material do curso que contém as imagens necessárias e arrastá-las para o Xcode. É simples assim.

#### 1.4.2. Usando as imagens para criar os protótipos

Agora, para usar as imagens já adicionadas ao projeto, instrua o aluno a proceder da seguinte forma:

* Entrar em **Main.storyboard** e alterar a visualização para iPhone 7. O local que se encontra esta opção pode ser visto no passo 1. Arrastar e soltar um **Image View**, que se encontra na paleta de componentes, dentro da cena e redimensioná-lo para que complete a cena por completo.
* Selecionar o **Image View** e na propriedade **Image** (no painel) inserir o nome “1. Navegar”. Este é o nome de uma das imagens que inserimos na pasta Assets e pode agora ser visualizada na cena do Storyboard.
* Trocar o simulador para iPhone 7 como mostrado nos passos 7 e 8 e executá-lo. Ele ficará parecido com o mostrado no passo 10.
* Depois, arrastar e soltar uma **View Controller** no Storyboard e inserir uma imagem como feito anteriormente. Fazer isso para cada uma das imagens restantes. Ficará como no passo 14.

#### 1.4.3. UIImageView

Educador, explique sobre a view que exibe imagens no nosso app e a classe a qual ela corresponde, a **UIImageView**, e explique as suas hierarquias. Explique que para uma imagem ser exibida é necessário fornecer uma **UIImagem** apenas especificando o nome do arquivo de imagem.

#### 1.4.4. UIViewController

Também explique o **UIViewController**, como sendo uma tela de nosso app, semelhante às **Activities** do Android.

#### 1.4.5. Continuando os protótipos

Educador, agora é necessário ligar as cenas e para isso, primeiro devemos inserir botões.

Instrua o aluno a fazer isso da seguinte forma:

* Abrir a **Main.storyboard** localizar o botão na paleta de componentes e arrastá-lo à cena. Ver as imagens no passo 1, adicionar e redimensionar botões conforme os quadrados verdes, tantos quanto forem necessários.
* Entrar nas propriedades de cada botão e remover seus nomes, de modo que todos os botões fiquem transparentes. É possível também mudar a cor na propriedade **Background**.

#### 1.4.6. UIButton

Educador, com os botões colocados nos locais certos, explique como ele funciona. Ele se assemelha ao Button do Android e é basicamente um controle de entrada que recebe interações de diversos tipos de toque do usuário.

#### 1.4.7. UISegue

Agora, as cenas serão ligadas entre si e serão trocadas, com animação de transição, a partir de toques simples nos botões.

Para isso, seguir os seguintes passos:

Veja na imagem do passo 6 um mapa de ligações de cada botão à outra tela. Para realizar cada ligação, segure a tecla **Control** de seu teclado e segure e arraste o botão até a cena de destino, ou seja, cena a qual será redirecionada ao ser tocado tal botão. Ao soltar, clicar na opção **Present Modally**, que estará disponível, para que pareça que uma tela passa sobre a outra na troca de cenas.

É possível ver a segue, agora desabilitar a animação, selecionando a Segue e desmarcando a opção **Animates**. Repetir o processo para cada ligação com base na imagem no passo 6.

Agora as ligações estão concluídas.

### 1.5. Resumo

O aluno conheceu o **Storyboard** e viu que é possível prototipar apenas **arrastando e soltando componentes** para o Storyboard, sem escrever código algum. Conheceu três novos componentes: a **UIImageView**, a **UIViewController** e a **UISegue** e as utilizou para prototipar nossa primeira aplicação, o **MusicProject**.

## Aula 2

### 2. Views e Auto Layout

Agora que o aluno já sabe utilizar o Xcode, vamos aprender a utilizar, dimensionar e restringir algumas views para que possamos ter uma noção sobre o Auto Layout e mais alguns alternativos à ele, necessários para o decorrer do curso.

### 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

* Reconhecer e diferenciar mudanças externas e mudanças internas referente ao layout de um app.
* Reconhecer três tipos de formas de restrições e ajustes de layout que podem ser utilizados em iOS;
* Fazer uso do auto layout para criação da interface de um app.

### 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.

* Mudanças externas;
* Mudanças internas;
* Disposição Auto Layout Versus baseada em Frame;
* Aplicando o Auto Layout.

### PONTOS IMPORTANTES

Esta aula terá um grande conteúdo prático onde o foco será o uso, redimensionamento e restrições de Views. A aula é um pouco longa e há muitos passos a serem seguidos, é necessário que o aluno mantenha a atenção e que você o auxilie a concluir o objetivo da prática.

É importante lembrar também, que durante a prática estaremos inserindo nomes e imagens manualmente no aplicativo de música. Não é esta a forma que este tipo de aplicativo é feito, faremos dessa maneira apenas para ilustrar o funcionamento do auto layout. Um app comum reconhece o nome e a imagem do arquivo de música e os mostram na interface, isso o aluno aprenderá a fazer no decorrer do curso

### 2.1. Mudanças externas

Aqui iniciaremos a falar sobre o Auto Layout. Ensine ao aluno que é possível que todas as Views se auto ajustem em uma View Controller de acordo com as restrições (constraints) atribuídas a elas.

Defina mudanças externas como mudanças que ocorrem decorrente de algum evento fora do app que alteram o tamanho ou forma da superview. Cite alguns exemplos de mudança externa e esclareça dúvidas dos alunos. Nestas mudanças também devem ser observados os diferentes tamanhos de telas entre os modelos de iPhone e iPad.

### 2.2. Mudanças internas

Educador, defina as mudanças internas como aquelas que ocorrem quando uma view que possui restrições e que afeta outra view é alterada. Dê alguns exemplos de quando elas ocorrem. Explique também o que ocorre quando o aplicativo exibe texto ou imagem dinâmica, necessitando de ajuste do layout antigo e como o auto layout resolveria isto.

Explique o que é internacionalização e sua exigência de um auto ajuste. E por fim, dê o exemplo das fontes dinâmicas, que podem alterar de tamanho e como a interface deve se comportar com o auto layout.

### 2.3. Disposição Auto Layout Versus baseada em Frame

Educador, explique ao aluno a possibilidade de implementar uma interface de usuário como também usar o auto layout. Com a Figura 30 mostre como funciona o sistema de coordenada do iOS. Na Figura 30, temos uma superview, que é o retângulo maior e mais claro, com mais 2 views menores e mais escuras. Para definir a posição e tamanho das views, basta três propriedades, a origem, em relação a superview (20,20), a altura (80) em relação a origem e o comprimento (120) em relação a origem, sendo que a altura e o comprimento no ponto da origem são 0 e aumentam conforme se afastam.

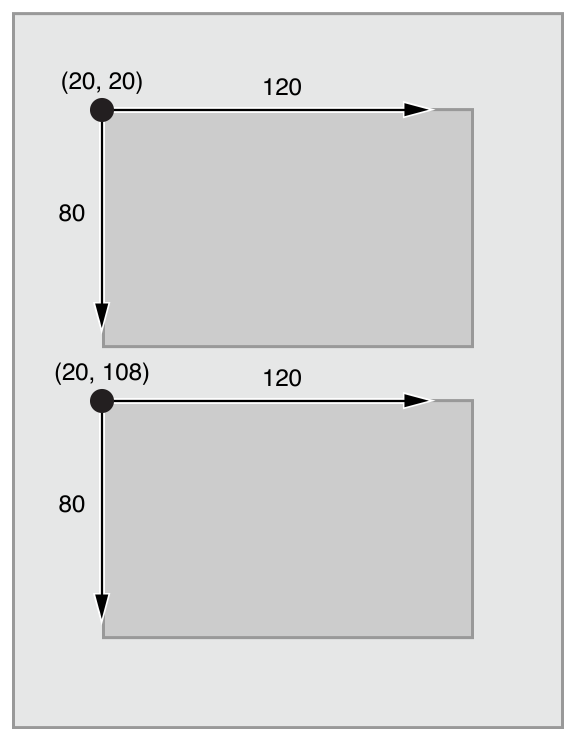


Figura 30 - Sistema de coordenadas do iOS

#### 2.3.1. Layout baseado em Frame

Educador, explique ao aluno como eram calculados a posição e tamanho de cada view e a problemática quando ocorre alguma mudança.

Vale ressaltar que programando os frames da view nos confere máxima flexibilidade e poder, mas é muito mais susceptível ao erro e o trabalho é exaustivo.

#### 2.3.2. Layout baseado em máscaras de redimensionamento automático

Dado a definição anterior, mostre ao aluno que a as autoresizing masks são capazes de aliviar um pouco deste trabalho. Elas são utilizadas apenas para layouts que se adaptam a mudanças externas e oferecem suporte apenas para layouts simples, o que nos faz ainda ter de realizar, via código, modificações para nossas máscaras, caso optemos por interfaces mais complexas.

#### 2.3.3. Auto Layout

Assim temos o auto layout para aliviar todas as nossas dores de cabeça. Educador, explique ao aluno que com o Auto Layout, não precisamos nos preocupar com a estrutura das views, somente nos preocuparemos com como elas se relacionam entre si.

O auto layout respondem dinamicamente às mudanças internas e externas e recalcula o tamanho e posição de cada view de acordo com essas mudanças.

Para dominar o uso do auto layout deve-se entender a lógica por tras dos layouts baseados em restrição e aprender a API.

Explique que podemos definir as restrições via código e via Interface Builder, mas neste curso, o faremos este último.

### 2.4. Aplicando o Auto Layout

Agora entraremos na aplicação prática do assunto. Este procedimento é bem longo, é bem suceptivel que o aluno se perca nos passos deste, por isso o auxilie e tire suas dúvidas.

Em síntese, os passos são estes:

* Abrir o app **MusicProject** em **Unidade\_3\_-\_Aula\_1\_-\_Exemplo\_1.zip**.e verificar a falha de layout. Em seguida, teclar **Control + **para acessar o modo paisagem e verificar outra falha.
* Abrir o storyboard Main e adicionar uma UIImageView e selecioná-la. Abrir a aba **Size Inspector**  e modificar as dimensões conforme o passo 1.
* Abrir a aba **Add New Constraints** no interface builder e selecionar as restrições como no passo 3, e explique os detalhes da Figura 35.

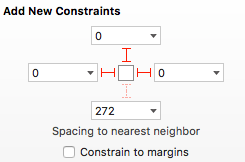


Figura 35 - Restrições de espaçamento para a Image View

* Adicionar uma UIView no interface builder e trocar suas dimensões como no passo 4. Em seguida, adicionar as restrições como no passo 5 e ativá-las clicando em **Add 4 Contraints**. Para esta mesma view, adicionar restrição de altura em **Add New Constraints**, marcando a caixa Height, conforme o passo 7 e também mudar a cor do fundo indo na propriedade **Background** alterando para **EBEBEB**. Verificar o resultado em simuladores diferentes, agora as Views se adaptam.
* Se o aluno não atingir o resultado esperado, auxilie-o neste procedimento.
* Para criar um controlador do player, adicionar uma UIImageView dentro da View inferior e atribuir a imagem **ic\_pause.png** para ela. Adicionar restrições conforme o passos 14 e 15, selecionando a Image View e indo até **Add New Constraints.**
* Explique os significados das linhas vermelhas, amarelas e azuis. Ao atualizar os frames em **Update Frames** é possível corrigir as linhas amarelas.
* Adicionar uma restrição de posição X em **Add New Alignment Constraints** e então **Horizontally in Container**. Atualizar os frames.
* Adicionar uma Image View do lado direito do ícone de pause e atribuir a imagem **ic\_foward.png** à ela. Alinhar ao ícone de pause, segurando a tecla **Control** e clicando e arrastando na imagem até o botão pause. Selecionar a opção **Center Vertically**, que estará disponível. Realizar o mesmo para o botão **preview** à esquerda do botão de pause e visualizar o resultado.
* Para inserir textos, inserir duas UILabels como subviews da view inferior. Selecionando a primeira UILabel, escrever o nome de uma música no painel de propriedades, alterar a fonte para tamanho 21 e adicionar uma restrição em **Add New Constraints**, baseadas na Figura 45 do passo 27. Definir mais restrições em **Add New Alignment Constraints** conforme o passo 28. Atualizar e observar o resultado.
* Selecionar a segunda UILabel e colocar o nome do artista em Propriedades e alterar a fonte para 17. Definir as restrições da label em **Add New Constraints**, conforme o passo 31 e em **Add New Alignment Constraints,** conforme o passo 32.
* Atualizar o frame e verificar o resultado final.

### 2.5. Resumo

Nesta aula o aluno aprendeu três formas de restringir o layout para ajustá-lo a **mudanças**, sendo elas **externas** e **internas**. Tais formas foram utilizando **layout baseado em frame**, que tem maior potencial mas é extremamente trabalhoso de se implementar; o layout **baseado em mascaras de redimensionamento**, que reduz um pouco de trabalho, mas se restringe apenas à ajustes a mudanças externas e a interfaces simples; e por último, o **auto layout**, onde vimos na prática a facilidade de definir as restrições e como o layout se ajusta em diferentes simuladores e orientações de tela sem ao menos escrever uma linha de código.

## Aula 3

### 3. Controles e eventos de entrada

Como é óbvio, precisamos atribuir aos nossos botões um reconhecimento de entrada do usuário e definir ações para cada entrada realizada. Para isso, aprenderemos um pouco mais sobre Controles de entrada e eventos de maneira fácil, que só o Xcode e a linguagem Swift pode nos oferecer.

### 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

* Criar métodos de entrada para botões;
* Criar eventos ou ações para certos tipos de entradas;
* Aprenderá a fazer conexões entre botões e seus métodos de entrada e ações;
* Criar timer progressivo e regressivo;
* Criar Sliders.

### 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.

* Controles de Entrada;
* Eventos de Entrada;
* Adicionando ações aos nossos botões;

### PONTOS IMPORTANTES

Esta será a primeira aula que o aluno terá contato com o código na linguagem Swift e de fato implementar algo, então dúvidas surgirão, erros de digitação, e alunos desanimados. Estamos mais do que acostumados com isso, então daremos a devida atenção a eles.

### 3.1. Controles de entrada

Educador, compare e relembre sobre controladores de entrada em Android e deixe claro que este são, basicamente, elementos visuais que estão prontos para receber algum tipo de interação do usuário, como toque, texto ou voz.

#### 3.1.1. UIControl

Educador, agora vamos apresentar e definer alguns tipos de controladores de entrada em iOS.

Começando pelo UIControl, ele é uma subclasse da UIView e pode manipular diversos eventos estipulados pelas constantes definidas pela UIControlEvents. Alguns eventos são: encostar na tela, deslizar dentro dos limites do Control, deslizar para dentro ou para fora dos limites do Control, encostar e tirar o dedo da tela (que será mais utilizado entre nós) e muitos outros eventos.

Explique também como é possível atribuir esta classe para UIViews.

#### 3.1.2. UIButton

UIButton são botões simples, ele recebe toque do usuário e com isso executa alguma atividade no app. Explique que os botões já possuem componentes pré-definidos como título e imagem e explique também sua relação hierárquica com a UIControl.

#### 3.1.3. UITextField e UITextView

O UITextField é onde o usuário pode inserir textos e o UITextView é onde se insere textos de múltiplas linhas, editaveis. Explique que estes controladores possuem protocolos e suas possibilidades dado a isto.

#### 3.1.4. Outros

Também temos uma lista de vários outros controladores de entrada. Utilize os exemplos do livro do aluno e explique a função e funcionamento de cada um deles.

### 3.2. Eventos de entrada

Há outros eventos de entrada além daqueles que vimos em UIControlEvents. A seguir veremos alguns deles.

#### 3.2.1. Reconhecimento de gestos

Educador, explique ao aluno que devemos implementar, via código um método de reconhecimento de gestos, tal método é **addGestureRecognizer()**  e pode ser atribuído à alguma view que não é extensão da UIControl.

Explique sobre a classe e as particularidades do código que foi utilizada como exemplo:

|  |  |
| --- | --- |
| **Nomenclatura** | **Definição** |
| let recognizer =  UITapGestureRecognizer(target: self, action: #selector(viewDidTouched))  view.addGestureRecognizer(recognizer) | Classe mais utilizada para reconhecimento de gestos. |
| let recognizer =  UITapGestureRecognizer(target: self, action: #selector(viewDidTouched)) | Seleciona qualquer método que esteja dentro do contexto do objeto passado via argumento (target). |
| let recognizer =  UITapGestureRecognizer(target: self, action: #selector(viewDidTouched)) | Argumento que indica o local do método a ser utilizado pelo objeto. Dentro da própria classe. |
| let recognizer =  UITapGestureRecognizer(target: self, action: #selector(viewDidTouched))  view.addGestureRecognizer(recognizer) | Método a ser executado quando a view for tocada. |
| let recognizer =  UITapGestureRecognizer(target: self, action: #selector(viewDidTouched))  view.addGestureRecognizer(recognizer) | Método que atribui a view o reconhecimento de gestos. |

#### 3.2.2. 3D Touch

Está disponível apenas nas versões iPhone 6s ou superiores, é um recurso que mostra opções e realiza ações de acordo com a pressão de nosso toque na tela.

#### 3.2.3. Reconhecimento de movimento

Detecta movimentos através do giroscópio e acelerômetro. Explique ao aluno o funcionamento do acelerômetro e giroscópio.

Agora explique três situações em que precisamos acessar informações destes hardwares e como proceder.

### 3.3. Adicionando ações aos nossos botões

Agora vamos adicionar botões e dar função a eles.

* Abrir o projeto em **Unidade\_3\_-\_Aula\_3\_-\_Exemplo\_1.zip** e observar a mudança do arquivo ViewController.swift. Abrir o Main.storyboard e observar as novas imagens, um UISlider e duas labels de tempo que foram adicionados, verifique também os estados do botão Play.
* Para mostrar estes estados, selecionar o botão Play e na propriedade **State Config** e escolher a opção **Selected**. Na propriedade **Image**  colocar o nome **ic\_pause**, para que o ícone seja trocado quando o botão está selecionado.
* Para referenciar os componentes no código, abrir o **Assistant Editor** e organizar de modo que o storyboard esteja do lado esquerdo e o MusicViewController.swift esteja do lado direito, conforme os passos 1 e 2. Segurar a tecla **Control** e clicar sobre o botão Play, e arrastar a linha para dentro da classe (exeto dentro dos métodos), em seguida surgirá uma caixa de diálogo com algumas opções. Explique cada uma delas, conforme o passo 5 e 6. Na opção **Name** inserir o nome **playButton** e clicar em **Connect** e verificar a linha de código que aparecerá na sua classe. Repetir o processo para o UISlider e os dois UILabels, de modo que o código fique como no passo 10.

Educador, primeiro explique ao aluno o que ocorrerá com o ícone do botão Play, com as Labels de tempo e com o Slider, quando o usuário apertar o botão Play e o que ocorrerá quando apertar novamente (pause). Depois explique como é definido os tempos de duração da música, o tempo restante e o total tocado. Explique também como o timer é inicializado e disparado.

Escreva e explique os métodos **resumeTimer()**, **pauseTimer()**, **updateTimer()** e **updateLabels()**.

|  |  |
| --- | --- |
| **Nomenclatura** | **Definição** |
| func resumeTimer() {  timer = Timer.scheduledTimer(timeInterval: 1,  target: self,  selector: #selector(updateTime),  userInfo: nil,  repeats: true)  } | **Método resume.** |
| func resumeTimer() {  timer = Timer.scheduledTimer(timeInterval: 1,  target: self,  selector: #selector(updateTime),  userInfo: nil,  repeats: true)  } | Nome da variável. |
| func resumeTimer() {  timer = Timer.scheduledTimer(timeInterval: 1,  target: self,  selector: #selector(updateTime),  userInfo: nil,  repeats: true)  } | Valor que o timer receberá. |
| func resumeTimer() {  timer = Timer.scheduledTimer(timeInterval: 1,  target: self,  selector: #selector(updateTime),  userInfo: nil,  repeats: true)  } | Classe que utiliza um método (selector:) especificado em seu parâmetro (target:) que repete (repeats:) a cada intervalo (timeInterval:) de 1s. |
| func pauseTimer() {  timer.invalidate()  } | **Método pause.** |
| func pauseTimer() {  timer.invalidate()  } | Método que invalida o timer (interrompe as repetições) |

|  |  |
| --- | --- |
| **Nomenclatura** | **Definição** |
| func updateTime() {    } | **Método update time** |
| func updateTime() {    if Int(remaingTime) == 0 {  timer.invalidate()    totalPlayed = 0.0  remaingTime = musicDuration    }else{  remaingTime -= 1  totalPlayed += 1  }  updateLabels()  } | Condicional que invalida o timer quando o tempo restante for zero, atribui o tempo total tocado como zero e atribui o tempo restante como tempo total de duração da música, ou seja, o tempo rodará do inicio novamente. |
| func updateTime() {  if Int(remaingTime) == 0 {  timer.invalidate()  totalPlayed = 0.0  remaingTime = musicDuration  }else{  remaingTime -= 1  totalPlayed += 1  }  updateLabels()  } | Se o tempo restante não for zero diminuir do tempo restante e aumentar 1 do tempo total tocado. |
| func updateTime() {  if Int(remaingTime) == 0 {  timer.invalidate()  totalPlayed = 0.0  remaingTime = musicDuration  }else{  remaingTime -= 1  totalPlayed += 1  }  updateLabels()  } | Chama o método update Labels. |
| func updateLabels() {  let formatter = DateFormatter()  formatter.dateFormat = "mm:ss"  let remainingTimeString = formatter.string(from: Date(timeIntervalSinceReferenceDate: remaingTime))  let totalPlayedString = formatter.string(from: Date(timeIntervalSinceReferenceDate: totalPlayed))  } | **Método update labels** |
| let formatter = DateFormatter()  formatter.dateFormat = "mm:ss" | Cria uma constante formatador que recebe o método formatador de data e define o formato de data como mm:ss |
| let remainingTimeString = formatter.string(from: Date(timeIntervalSinceReferenceDate: remaingTime))  let totalPlayedString = formatter.string(from: Date(timeIntervalSinceReferenceDate: totalPlayed))  } | Converte os valores de tempo restante e o tempo total tocado, recebidos no método updateTime no formato definido anteriormente (Date) e em seguida converte o tempo para String para ser exibida. |

Para que os timers funcionem,

* No método updateTime() acrescentar o código como no passo 1 para que o botão play troque de ícone quando o tempo restante da música chegar a zero. Inserir os comandos como no passo 2, dentro do método updateLabels(), para que as Labels sejam atualizadas com o texto que for atribuídos a elas.
* Ainda neste método, adicionar o código como no passo 3 para que o Slider seja atualizado, mudando o valor máximo do Slider para 180, na propriedade **Maximum**.
* Referenciar o botão Play soltando-o no final da classe, escolher o tipo de conexão como **Action** e escolher o nome como **playButtonTouchUpInside**. Como Type, escolher **UIButton** e no campo Event, escolher **TouchUpInside**. Clicar em **Connect**. Surgirá o método como no passo 10.
* Dentro deste novo método, adicionar novo código, seguindo o passo 11.
* Este método fará que, caso o botão Play seja tocado pelo usuário, e o estado do botão seja **selecionado**, subentende-se que a música está tocando e a música deve ser pausada, chamando o método **pauseTImer()**. Se o estado não for **selecionado**, com este toque, se inicia a música ou se continua onde ela parou e ainda se troca o estado do botão.
* É importante lembrar que ainda não temos nenhuma música envolvida em nosso app, o que fazemos apenas é inicializar e pausar o nosso temporizador.

Faça o teste, o slider não irá obedecer. Vamos dar ação a ele:

* Segurar a tecla **Control** e arrastar a linha para a classe no painel de implementação, surgirá a caixa de Connections. Dar ação ao slider mudando o *Event* para **Value Changed**, e adicionar as instruções no método criado, como nos passos 3 e 4. Executar e explicar o código implementado.

### 3.4. Resumo

Desta vez, o aluno aprendeu a criar funções para o botão. No caso para um toque no botão Play, o aluno atribuiu a função de iniciar ou parar os timers (criados por ele mesmo) de tempos tocado e restante da música, assim como aprendeu como funcionam os métodos por traz disso. Aprendeu também a dar funcionalidade para o Slider adicionado.