Unidade 1

Aula 1

O mercado mobile é de longe o que mais cresce no ramo de tecnologia. As pessoas estão cada vez mais conectadas e apegadas aos seus smartphones. A era dos computadores pessoais está dando a vez para a era mobile. O conceito de computador pessoal ou PC, veio da década de 80, quando computadores gigantescos passaram a ser cada vez mais portáteis e acessíveis pelas famílias, porém este conceito ganha cada vez mais vigor com o tempo.

# Introdução

Os computadores estão cada vez menores e portáteis: após a década de 90, eles passaram a caber no bolso com a chegada dos primeiros smartphones.

Em 2007 foi lançado o primeiro smartphone equipado com o sistema iOS, o iPhone, e desde então o mercado mobile tem se voltado diretamente para ele, competindo apenas com o gigante Android. Hoje o sistema operacional Android tem predominância de mercado, instalado em aproximadamente 84% dos smartphones contra apenas 11% de smartphones equipados com iOS. Este número pode ser intimidador, mas na prática é bem mais otimista para o iOS. Você entenderá o motivo no decorrer do curso.

Com o crescimento do mercado mobile, a evasão dos usuários de sistemas web e desktop tem aumentado, então você está no lugar certo na hora certa, pois não há momento melhor para começar a desenvolver para iOS como este.

Desenvolver para smartphone é similar a desenvolver para desktop, porém mais divertido. Se existem infinitas possibilidades de aplicações desktop e web, então existem o dobro para mobile. A vantagem principal é a própria mobilidade. Com aplicativos mobile, podemos criar aplicações com bluetooth, GPS, Wi-fi, câmera, sensores de movimento, sensores de luminosidade, microfone, sons e etc. Coisas estas que não são tão comuns em notebooks e desktops. É realmente uma infinidade de aplicações. Além disso, com uma boa ideia , quem sabe você não se torna um bilionário?

Bem vindo ao curso de iOS 10!

## Para quem é este livro

Este livro foi escrito para aqueles que desejam ser ótimos desenvolvedores de aplicativos usando a plataforma iOS 10. Se você está procurando um material que não só o introduz ao desenvolvimento de apps, mas também oferece teoria e exercícios práticos suficientes para lhe tornar um bom desenvolvedor mobile, este curso é o que você procurava.

O curso de iOS, irá explicar ideias e conceitos gerais sobre desenvolvimento de aplicativos. Além disso, você poderá criar aplicativos incríveis usando a linguagem Swift, que é a linguagem utilizada para criação de apps em iOS 10 (verificar versão). Com este curso você será um desenvolvedor de aplicativos iOS eficaz e preparado para o mercado de trabalho.

## Requisitos

Para começar a sua jornada como desenvolvedor iOS, você vai precisar de algumas coisas antes de iniciar

Possibilidade de máquina virtual com SO apropriado ao Xcode

1. **Um computador Mac:** Um Mac é um requisito fundamental para o desenvolvimento de aplicativos em iOS. Além disso ele deve conter o sistema operacional OS X ou superior. Como o iOS é uma plataforma fechada da Apple, obviamente ela optaria em limitar seu uso a seus produtos, então você não conseguirá criar apps fora de um Mac. Se você é usuário de Windows, infelizmente não conseguirá produzir apps com iOS. Infelizmente no Brasil, os produtos da Apple são um pouco caros.
2. **Se cadastrar como desenvolvedor:** Para publicar e executar seus aplicativos em um iPhone (iPad, iPod Touch, ou qualquer outro aparelho da Apple) você precisará de uma licença anual, e será preciso verificar o valor, por exemplo no ano de 2016 o valor era de 99 dólares. Sem esta licença você poderá se registrar gratuitamente no site da Apple, mas poderá executar seus aplicativos apenas nos emuladores ou em versão teste (que possui poucos dias de validade) nos dispositivos Apple.
3. **Download do Xcode:** Uma vez que você se registrou como desenvolvedor da Apple, você poderá baixar o Xcode pela AppStore. O Xcode é a IDE para desenvolvimento iOS e você aprenderá mais sobre ela nas próximas aulas.
4. **Conhecimento em POO:** A linguagem Swift, adotada pela Apple para desenvolvimento dos seus SOs (não seria apps? SOs seria sistemas operacionais?) é totalmente orientada a objetos, possuindo recursos modernos em relação a outras linguagens encontradas hoje em dia. Ter conhecimento em POO irá te ajudar bastante neste curso. O módulo de POO deste curso, que você realizou, é mais do que o suficiente para você se dar bem neste módulo de iOS.
5. **Foco e Força de vontade:** No início poderá ser difícil fixar os conceitos. Nós iremos fazer você pensar diferente e mostraremos a filosofia por traz do design mobile. Bastante conteúdo para um curso, mas ele será extremamente prazeroso e gratificante.

## História do iOS

O iPhone é um smartphone da empresa **Apple** que popularizou o mundo dos telefones celulares inteligentes com a tecnologia *touchscreen*, onde o mercado era dominado pelos telefones celulares simples.



Figura 1 - iPhone 2G, o primeiro iPhone lançado

Os celulares simples realizavam tarefas como fazer ligações, enviar mensagens SMS e salvar contatos. Com o passar do tempo foram agregando cada vez mais funcionalidades, como gravar áudio e vídeo, ouvir música pelo rádio FM e até jogar alguns games. Antes de falar da história do **iOS**, sistema operacional do iPhone, vamos conhecer como era o mercado mobile antes do lançamento do primeiro iPhone.

### O primeiro smartphone touchscreen

Muitos se enganam quando pensam que todos os smartphones são touchscreen (tela sensível ao toque). Erro maior ainda dizer que o iPhone foi o pioneiro nesta tecnologia.

O termo smartphone, adaptando para o português, significa “telefone inteligente” e é um telefone móvel que combina recursos de [computador pessoal](https://pt.wikipedia.org/wiki/Computador_pessoal), com funcionalidades avançadas que podem ser estendidas por meio de programas executados por seu sistema operacional (OS), chamados de aplicativos ou **apps**. É uma junção de um PC (desktop) com um telefone móvel.

Dependendo da sua idade, você não deve nem se lembrar da época em que os celulares eram controlados por teclados numéricos. Eles nem sempre foram com tela *touch* desde o seu início.

O primeiro celular a ganhar este tipo de tecnologia é o **IBM Simon**, que chegou as lojas em 1994. Ele possuía uma tela resistiva, monocromática de 4,5 polegadas, seu sistema operacional era baseado em DOS (aquele mesmo que deu origem ao **Windows**), mas ao invés de exibir uma linha de comando, mostrava uma interface dinâmica, baseada em atalhos.



Figura 2 - Imagens do IBM Simon

Apesar do termo “*smartphone”* ter surgido somente em 1995, o IBM Simon foi considerado o primeiro celular deste segmento. Então coincidentemente ou não, ele foi o primeiro *smartphone* e o primeiro telefone móvel com tela *touch,* graças à junção das funcionalidades de PC e telefone comuns.

### A poderosa Apple Inc

Antes de falarmos do iOS, vamos falar um pouco da sua criadora, a Apple.

A multinacional foi fundada em 1974 por **Steve Wozniak** e **Steve Jobs** e agrada pela inovação e simplicidade. Apple é uma das maiores empresas que atuam no mercado de tecnologia, com destaque para smartphones, ultrabooks e tablets. Entre os produtos da empresa destacam-se os famosos **iPhone**, **iPad**, **iPod,** **MacBook, Apple TV e Apple Watch**. A Apple também atua no ramo de softwares, com o navegador Safari, o iTunes, o quicktime e os sistemas operacionais e de pagamentos Mac OS, OS X e o iOS, Apple Pay Brasil, são alguns exemplos. Além, é claro, de acessórios como fones de ouvido e capinhas de celular.

### iPhone e iOS

Como mencionado no tópico 1.3.2, o iPhone foi criado pela Apple e agora que já sabemos um pouco da criadora do iPhone, vamos conhecer seu histórico.

**2007 - iPhone 2G**: O lendário **Steve Jobs**, então CEO da Apple, lança mais uma de suas inovações. Nomeado **iPhone 2G** (Figura 1), foi colocado a venda em 29 de junho de 2007, sendo alvo de desejo de muitas pessoas. O aparelho contava com tela de 3,5 polegadas e era revolucionário, pois podia se conectar a internet, porém sua conectividade era com as redes GPRS (2G) e EDGE (apelidada de 2,5 G). O aparelho foi fabricado em versões de 4, 8 e 16 GB de armazenamento e não chegou a ser vendido no Brasil.

Seu sistema operacional não tinha nome próprio, ou seja, ainda não era chamado iOS e era visto apenas como uma adaptação do OS X (sistema operacional nativo dos computadores Apple).

No dia do seu lançamento os consumidores lotaram as lojas da Apple e mais de 200 mil smartphones foram vendidos em um único dia. Em outubro do mesmo ano ele atingiu 1 milhão de vendas.

Inicialmente o iPhone 2G rodava apenas aplicativos nativos, ou seja, você não conseguia instalar aplicativos, nem desenvolvê-los. Somente em outubro de 2007 a Apple liberou o desenvolvimento de aplicativos de terceiros.

**2008 - iPhone 3G:** Lançado em 9 de julho de 2008, o iPhone 3G trazia novidades como GPS, a Apple Store (loja que disponibiliza aplicativos de terceiros), jogos, acesso a internet via Wi-Fi e uma câmera de 2 megapixels, que naquela época era uma resolução inovadora.



Figura 3 - Imagens do iPhone 3G

Seu sistema operacional foi melhorado para a versão 2.0 e foi batizado como iOS. Como o sucesso continuou espantoso, as vendas cresceram e o aparelho começou a ser vendido oficialmente no Brasil.

**2009 - iPhone 3GS**: A versão 3.0 do iOS foi lançada em julho de 2009, com a chegada do iPhone 3GS. A versão 3GS possuía o mesmo design (aparência) do aparelho anterior, porém recebeu melhorias em seu hardware e software. Agora ele contava com uma câmera de 3 megapixels, bússola, comandos de voz, gravação e edição de vídeos no próprio device (do inglês, dispositivo), além de apresentar um hardware mais potente que executava os aplicativos com mais eficiência.

**2010 - iPhone 4:** Agora com seu design totalmente reformulado e a versão 4 do iOS, a grande novidade do aparelho estava no processador A4, com frequência de 1 GHz, com 512 MB de memória RAM, que permitia o uso de recursos multitarefa.

DICA: Em computação, recursos multitarefa referem-se à capacidade do hardware de processar várias tarefas simultaneamente. Em outras palavras, quando aplicado aos smartphones, temos dois ou mais aplicativos sendo executados ao mesmo tempo, e o usuário pode alternar entre eles, sem precisar inicializa-los.

Lançado juntamente com o iPhone 4, em junho de 2010, o iOS 4 contava com novidades, como criação de pastas para organização de apps, roteamento Wi-fi, corretor ortográfico e o FaceTime, um aplicativo para chamadas de vídeo.



Figura 4 - Fotos do iPhone 4

Uma falha no iOS 4 causou um escândalo na comunidade Apple. Quando os usuários seguravam o iPhone 4 cobrindo determinada parte do aparelho, o sinal da operadora simplesmente caia. Uma atualização logo foi feita em julho de 2010 para solucionar o problema.

**2011 - iPhone 4S:** Como sempre, recebendo melhorias de câmera, processador e qualidade de imagem e vídeo, o iPhone 4S contava com o iOS 5 e uma grande novidade, a Siri. Seu design é idêntico ao iPhone 4 (Figura 4). Lançado em outubro de 2011, junto com o aparelho, o sistema operacional contava com a assistente virtual Siri, uma central de notificações, sincronização do iTunes via Wi-fi, o surgimento da iCloud entre outras novidades.

**2012 - iPhone 5:** O iPhone 5 foi lançado com diversas novidades. Apesar de seu design ser bem parecido com a versão anterior, o novo iPhone possuía uma tela de 4 polegadas, meia polegada a mais do que seus antecessores. Além disso, o smartphone passou a suportar conectividade em 4G, memória RAM de 1GB e processador de dois núcleos. Além de possuir uma tela maior, o aparelho ficou mais leve, pesando apenas 112 gramas. Com esta versão veio o iOS 6.



Figura 5 - Fotos do iPhone 5

Anunciado em setembro de 2012, o iPhone 5 foi um sucesso de vendas. No dia do seu lançamento foram vendida s mais de 5 milhões de unidades somente nos EUA.

**2013 - iPhone 5S e iPhone 5C:** Lançados em setembro de 2013, o iPhone finalmente começou a se ramificar em categorias. Além de cores novas, foi lançada uma versão de baixo custo.

O iPhone 5S disponibilizava versões nas cores dourado, prata e cinza espacial, enquanto que o iPhone 5C apresentava as cores branca, azul, verde, amarela e rosa, sendo vendido como uma opção de menor custo do que as demais versões do IPhone.

Como linha principal, o iPhone 5S vinha equipado com dispositivo de leitura biométrica, ou seja, o usuário poderia se autenticar via impressão digital. O aparelho dispunha também de um processador de 64-bit e o iOS 7.

Com a proposta de atingir as classes sociais mais baixas, a Apple lançou o IPhone 5C, suas configurações eram idênticas ao iPhone 5, mas seu acabamento era em plástico. Qual era o acabamento do outro? Não foi citado no texto.

**DICA**: O fato do iPhone 5C ter seu exterior predominantemente em plástico foi o principal fator para o barateamento de seu preço. O hardware deste modelo tem algumas diferenças em relação ao iPhone 5, mas não refletiu significativamente no seu preço. Uma confusão que é gerada é que os usuários dizem que o iPhone 5C é a versão de baixo custo do iPhone 5S, quando na verdade ele é a versão barateada do iPhone 5.

Com o lançamento do iPhone 5C, o iPhone 5 foi descontinuado devido a diversos problemas (apesar do estouro de vendas, muitos problemas eram encontrados pelos usuários).



Figura 6 - Fotos dos modelos do iPhone 5C

**2014 - iPhone 6 e iPhone 6 Plus:** Durante os até então 7 anos do iPhone surgiram diversas pequenas novidades, mas nada tão grande quanto o lançamento do primeiro iPhone (lançado em 2007). A Apple passou a ser acusada por seus usuários de ter parado no tempo. Além das reclamações de processamento e memória RAM, o principal desgosto dos usuários era o tamanho da tela.

Steve Jobs concebeu o iPhone em uma época em que a internet móvel estava se iniciando, e com o passar do tempo as pessoas ficaram cada vez mais integradas no ambiente web, e a navegação pela internet passou a ser massacrante em smartphones com tela pequena. Já era época de inovar. Tim Cook, que havia assumido o posto de CEO da Apple após a morte prematura de Jobs (em 2011), resolveu lançar em setembro de 2014 a linha dos **iPhones 6**.



Figura 7 - Fotos do iPhone 6 (na esquerda) e iPhone 6 Plus (na direita)

O iPhone 6 e o iPhone 6 Plus traziam **telas de 4,7 e 5,5 polegadas respectivamente?**. Com os iPhones foi lançado o **iOS 8**, com diversas novidades.

**2015 - iPhone 6s e iPhone 6s Plus:** em 9 de setembro de 2015 foi anunciado o lançamento da linha “S” do iPhone 6 e 6 Plus. Seguindo o padrão das versões “S”, o design destes novos modelos não mudou em relação aos modelos anteriores, pois as versões **S** possuem apenas melhorias em hardware interno, e não mudanças visuais. Melhorias de hardware e o **iOS 9** foram inseridos, mas a grande novidade foi a tecnologia **3D Touch**, que usa o reconhecimento de pressão na tela para permitir que diferentes ações possam ser realizadas dependendo do nível de pressão do toque do usuário.

A câmera também melhorou para não ficar atrás de seus concorrentes, o iPhone 6s e 6s Plus vieram equipados com uma câmera de 12 megapixels, que chega a gravar vídeos com resolução 4K.

**2016 - iPhone SE:** Pensando em atender a todos os clientes Apple, visto que alguns gostavam da tela maior e outros preferiam a tela menor do aparelho, em março de 2016 a Apple lança o iPhone SE.



Legenda?

Seu visual (incluindo o tamanho de tela de 4 polegadas) é o mesmo de um iPhone 5S, mas ele possui as mesmas configurações de um iPhone 6s em termos de processamento. O iPhone SE é equipado de fábrica com o iOS 9 e não possui o recurso 3D Touch.

**2016 - iPhone 7 e iPhone 7 Plus:** Lançado em 11 de novembro de 2016, os iPhone 7 e 7 Plus possuem grandes novidades em relação a tudo que já foi visto até aqui.



Figura 8 - Foto do iPhone 7



Figura 9 - Foto do iPhone 7 Plus

Agora o armazenamento inicial dos iPhones é de 32GB, tendo a opção de 128GB e 256GB. Novas cores também estão disponíveis. Além do dourado, silver e rosê, temos o preto fosco e o preto piano.

Figura 10 - Cores do iPhone 7

****

Figura 11 - Cores do iPhone 7 Plus

Ambos possuem os mesmos tamanhos de tela (4,7 para iPhone 7 e 5,5 para iPhone 7 Plus). Seu processador é o A10, que garante maior desempenho e melhorias em jogos.

Desta vez os aparelhos estão classificados como IP67 segundo a norma IEC 60529, que garante resistência a água, respingos e poeira.

As câmeras continuam com 12 MP, porém no iPhone 7 Plus temos uma câmera extra com lentes grande-angular, teleobjetiva e zoom ótico de até duas vezes.

### Concorrentes (Google)

Como foi dito na introdução, a plataforma que domina o mercado é o Android, estando presente em 84% dos smartphones contra 11% do iOS.

O sistema operacional Android, desenvolvido pela Google, é de código aberto, ou seja, qualquer um pode criar sua versão e lançar um smartphone com sua própria versão Android. Isto facilitou realmente a propagação da plataforma, mas como qualquer um pode ter sua versão, podemos esperar aparelhos muito bons e até aparelhos de péssima qualidade, pois a Google não tem total controle sobre os fabricantes que fazem uso da plataforma. Resumindo, os 84% dos smatphones estão divididos entre os fabricantes e possuem qualidades diferentes.

Já o iOS, produzido pela Apple, opera somente nos sistemas operacionais da mesma, e este número é bem menor do que a gama que o Android atinge. Atualmente o iOS está presente nos iPhones, iPod Touch, iPads. Existem derivados do iOS que operam nos sistemas embarcados watchOS e tvOS, mas não são mais considerados iOS

Analisando por este ponto de vista, a Apple é detentora de todos os lucros que o iOS pode dar, além disso, por possuir um conjunto de aparelhos pequeno, é mais fácil para o desenvolvedor otimizar seu aplicativo, sendo assim, os aplicativos tem qualidade superior e mais garantida do que o Android.

Do ponto de vista do desenvolvedor é mais vantajoso que se especialize na plataforma iOS do que na Android. Seria ótimo se aprendêssemos as duas, pois são raras as vezes que encontramos aplicativos exclusivos de uma das duas plataformas. Como a mão de obra para desenvolvimento iOS é mais escassa, devido ao alto investimento que o desenvolvedor deve realizar para isto, ela é mais valorizada em relação a Android, que é totalmente de graça. Já que você passou pelo curso de desenvolvimento Android, ao término deste (iOS) você será um desenvolvedor mobile completo.

Quem é melhor? Android ou iOS? Você vai presenciar muito esta discussão, e uma coisa é certa, ela nunca terá uma resposta.

Tanto Android quanto iOS tem seus defeitos e qualidades, enquanto o iOS é ótimo em performance, aparência e segurança, o Android vence na flexibilidade, facilidade e rapidez de desenvolvimento, pois seu código é aberto. Por outro lado, produtos com iOS possuem um sistema extremamente fechado, e o desenvolvedor sente mais dificuldade para realizar certas coisas que seriam triviais no Android (por exemplo, tarefas rodando em background são quase proibidas no iOS, enquanto que no Android ela é natural). Já o Android, por ser muito libertino, acaba dando muita liberdade ao desenvolvedor, esta liberdade pode ser perigosa, podendo abrir portas para vírus, vazamento de informações, erros de software, erros de sistema, queda de desempenho e outros. Existem infinitos pontos que poderíamos destacar, mas não se preocupe, Android e iOS ainda lutarão por muito tempo.

## Usabilidade e experiência de uso

Um dos princípios da Apple é o “usuário em primeiro lugar”. Apesar de sermos os especialistas, devemos pensar como usuário. O app não pode ser apenas bonito, precisa ter boa usabilidade, ser intuitivo e de fácil uso. No decorrer do livro iremos entender e proteger a sete chaves este princípio conosco, mas vamos entender o básico.

### Princípios de Design

Como um designer de apps você tem a oportunidade de entregar produtos extraordinários que ganharão destaque na Apple Store. Para isto você deve conhecer as maiores expectativas que os usuários têm sobre um app iOS.

São três temas primários que diferenciam um aplicativo iOS das outras plataformas:

* **Clareza:** Os textos são legíveis em qualquer tamanho de tela, os ícones são lúcidos e precisos, adornos extras são sutis e as funcionalidades devem ser orientadas ao design (a maioria das funcionalidades de um app já tem um padrão de design preestabelecido).
* **Deferência:** A interface, ou visual do aplicativo, não compete com seu conteúdo. Eles devem ser complementares entre si.
* **Profundidade:** Animações e camadas trabalham em conjunto para criar uma experiência de usuário mais envolvente.

No decorrer do curso iremos pontuar outras dicas e sugestões de design que a Apple define.

## Resumo

Nesta aula vimos um pouco das vantagens do desenvolvimento mobile. Vimos que o desenvolvimento web vem perdendo espaço para os aplicativos mobile, então é vantajoso que tiremos proveito desta onda para aprendermos e lançarmos aplicativos novos. Você também conheceu um pouco sobre a história dos smartphones, com o IBM Simon, e a história do revolucionário do iPhone.

Vimos que o iOS não está sozinho no mercado. A (ou O?) gigante Android domina 84% dos smartphones, mas o iOS ainda possui sua reputação.

Por fim, introduzimos o estudo sobre usabilidade e experiência de uso, que será discorrido no decorrer do curso.

## TDP

## Exercícios

## TDP

Aula 2

Como primeiros passos para iniciarmos a nossa jornada no desenvolvimento de iOS, precisamos preparar nosso ambiente de desenvolvimento. Nesta aula iremos nos registrar nos site da Apple, instalar o ambiente de desenvolvimento Xcode direto da loja de aplicativos AppStore. Após termos preparado nosso ambiente de desenvolvimento iremos finalmente criar nosso primeiro programa na linguagem Swift.

# Ambiente de desenvolvimento

Diferente do desenvolvimento Android, em iOS temos algumas restrições para desenvolvimento. A primeira e principal delas é que só é possível desenvolver para iOS usando um computador Mac. Outro ponto, que não é necessariamente uma limitação, mas é um passo a mais, é que todos precisam ter um registro de desenvolvedor no site da Apple. Por fim a limitação principal que temos ao desenvolver para iOS é que para publicarmos nossos apps precisamos pagar uma licença no Apple Developer Program. Veremos estes pontos no decorrer desta aula.

## ID Apple

Agora que você tem um Mac, é necessário criar uma conta na Apple. Se você já está com o mac OS instalado, provavelmente já tenha um ID Apple.

O ID Apple é uma conta que precisamos para utilizar os serviços da Apple, inclusive para utilizar seu próprio sistema operacional. No nosso curso iremos utilizá-lo para fazer download do Xcode (a IDE utilizada para criarmos nossos apps iOS). Para publicarmos nossos apps precisamos de uma conta especial (Apple’s iOS Developer Program) que pode ser usada pagando uma taxa média de $100,00 por ano, lembrando que você sempre deve verificar a atualização deste valor no momento que estiver fazendo sua conta.

Como estamos aprendendo, não vamos criar a conta neste momento, você pode criar quando começar a divulgar seus app’s, neste inicio vamos utilizar apenas a licença de teste. Se você não tem um ID Apple ainda, vamos criar um agora.

1. Para criar o ID Apple, entre no seguinte site e preencha o formulário: <https://appleid.apple.com/account#!&page=create>
2. Você deve seguir todos os passos e ao final, utilizar o código de verificação enviado ao e-e-mail que você cadastrou.
3. Agora já temos o nosso ID criado, vamos configurar nossa Apple Store com esta nova conta. Sem ela não conseguiremos baixar os aplicativos para Mac. Abra a **App Store.**

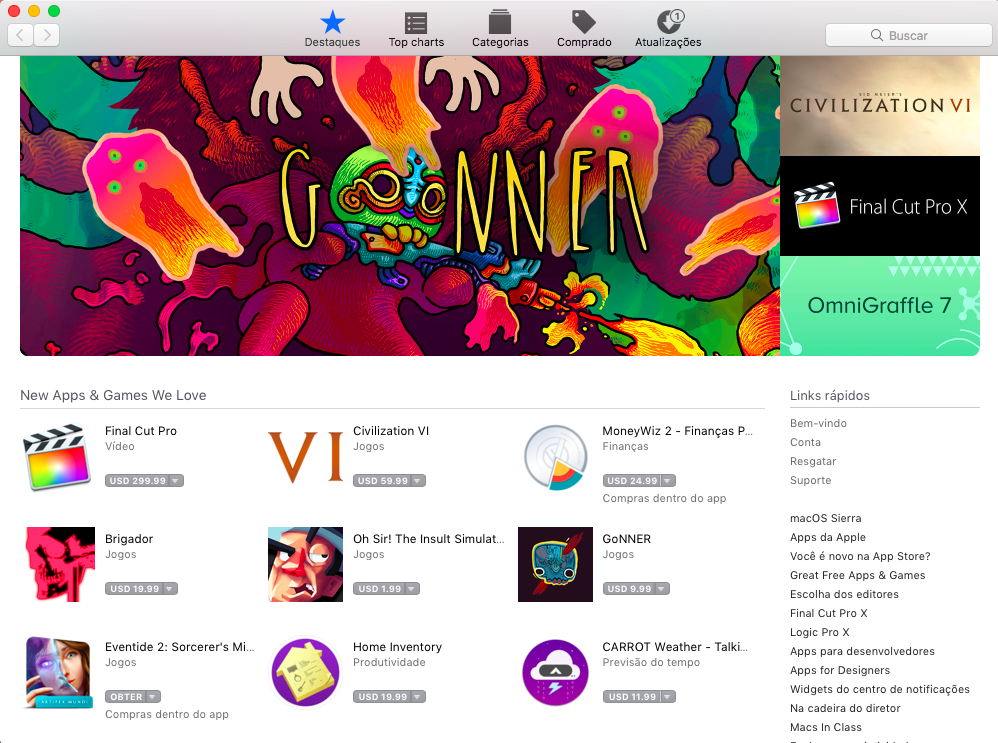


Figura 12 - App Store

Sempre será a tela acima que aparecerá quando se acessa a App Store? Caso não seja o aluno poderá ter dificuldades.

1. No menu, clique em **Loja > Iniciar Sessão...**



Figura 13 - Localização da opção "Iniciar Sessão..." da App Store

1. Entre com seu e-mail e senha cadastrados no ID Apple.



Figura 14 - Tela de início de sessão da App Store

1. Aceite os termos de uso e insira seu endereço (sim, este tipo de dado pessoal é obrigatório pela Apple).
2. Comece a comprar!

Se a opção **Iniciar Seção** não apareceu então aparentemente você já está com sua conta cadastrada na Apple Store, então não é necessário executar os passos acima.

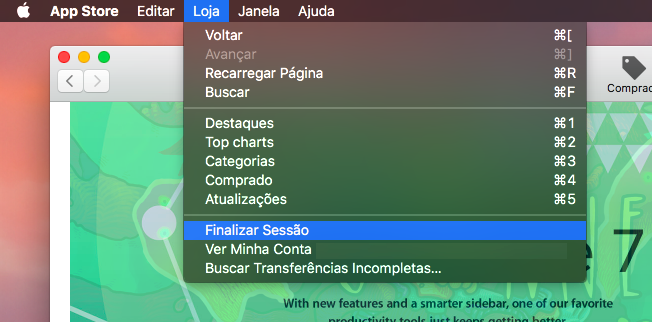


Figura 15 - Opções mostradas quando o usuário já está com uma sessão iniciada

## Xcode

Tendo nosso AppleID criado, a segunda coisa que precisamos é do Xcode.

O Xcode é a IDE (*Integrated Development Environment, ou Ambiente de Desenvolvimento Integrado*) oficial da Apple. O Xcode é o centro de desenvolvimento dos produtos Apple e com ela temos um incrível ambiente produtivo para a criação de apps para Mac, iPhone, iPad, Apple Watch e Apple TV.

Em comparação ao Android, o Xcode se equivale ao Android Studio. O diferencial do Xcode é que ele é desenvolvido pela própria Apple e é compatível apenas com Macs, isto garante que a IDE irá funcionar com vigor, sem travamentos até mesmo com Macs mais antigos e menos potentes, pois o app (Xcode é dado como um app para Mac) foi criado de forma otimizada para os Macs.



Figura 16 - Imagem institucional do Xcode

Infelizmente o Xcode não vem instalado por padrão, então teremos que baixá-lo da App Store. Em sua aula ele já estará instalado, mas para que saiba, o Xcode tem aproximadamente 4,5 GB, então reserve um espaço no seu HD e um tempo para esperar o download em sua casa.

O Xcode está em sua versão 8, com ela temos uma série de novas funções como suporte para desenvolvimento ao iOS 10 e outras coisas que veremos no decorrer do curso. Vamos então baixar o Xcode 8.0.

Você pode executar os passos a seguir em sua casa caso queira instalar o Xcode em seu Mac pessoal. Como este processo de instalação é um pouco demorado, sua escola já estará com tudo pronto, então pule para o tópico 2.3 desta aula. Para realizar a instalação do Xcode, em sua máquina, realize os seguintes passos:

1. Abra a **App Store** e procure por Xcode 8. Opcionalmente você pode tentar acessar utilizando esta URL no seu Safari ou browser de preferência: <macappstores://itunes.apple.com/br/app/xcode/id497799835?mt=12>

O link acima não está funcionando.Mesmo que se copie e cole no browser escolhido.



Figura 17 - Buscando Xcode na App Store

1. Nos resultados da busca, localize o Xcode, clique em **Obter** e em seguida clique em **Instalar App.**

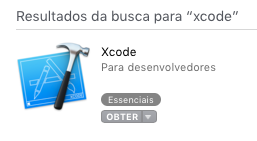


Figura 18 - Clique em Obter



Figura 19 - Clique em Instalar App

1. Na nova tela você terá que se autenticar, então insira seu ID Apple e senha.

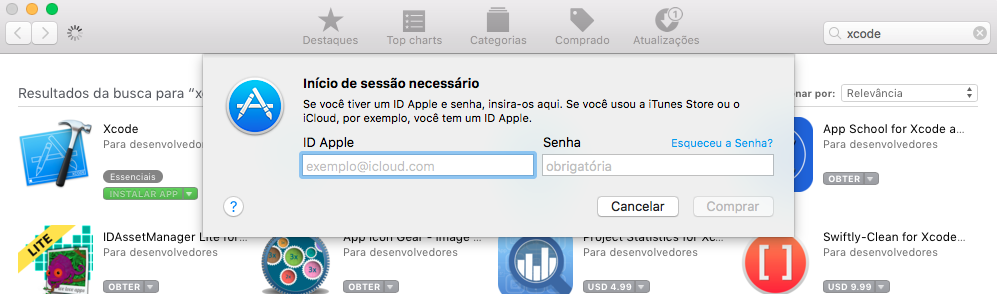


Figura 20 - Autenticação na App Store

1. Se você não baixou nenhum app anteriormente na App Store, você precisará confirmar seus dados de pagamento (lembrando que o Xcode é um app gratuito).
2. O app irá baixar e instalar.

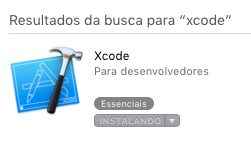


Figura 21 - Instalação do Xcode iniciada

1. Você pode acompanhar o andamento da instalação entrando no Launchpad (que fica na Dock do Mac) localizando o ícone do Xcode.



Figura 22 - Launchpad



Figura 23 - Andamento da instalação do Xcode (ícone do Xcode no Launchpad)

1. Aguarde até a instalação terminar e abra o Xcode.



Figura 24 - Xcode instalado e pronto para ser usado

Pronto. Temos nosso Xcode instalado. Para conhecermos sua interface, veja os tópicos abaixo. Ainda não vamos criar um projeto, então apenas acompanhe este conteúdo.

## Conhecendo o Xcode

Como falamos anteriormente o Xcode é bastante integrado com o Mac, então nosso trabalho flui naturalmente, sem travamentos ou surpresas. Existem algumas funcionalidades e painéis que se destacam. Vamos vê-los a seguir.

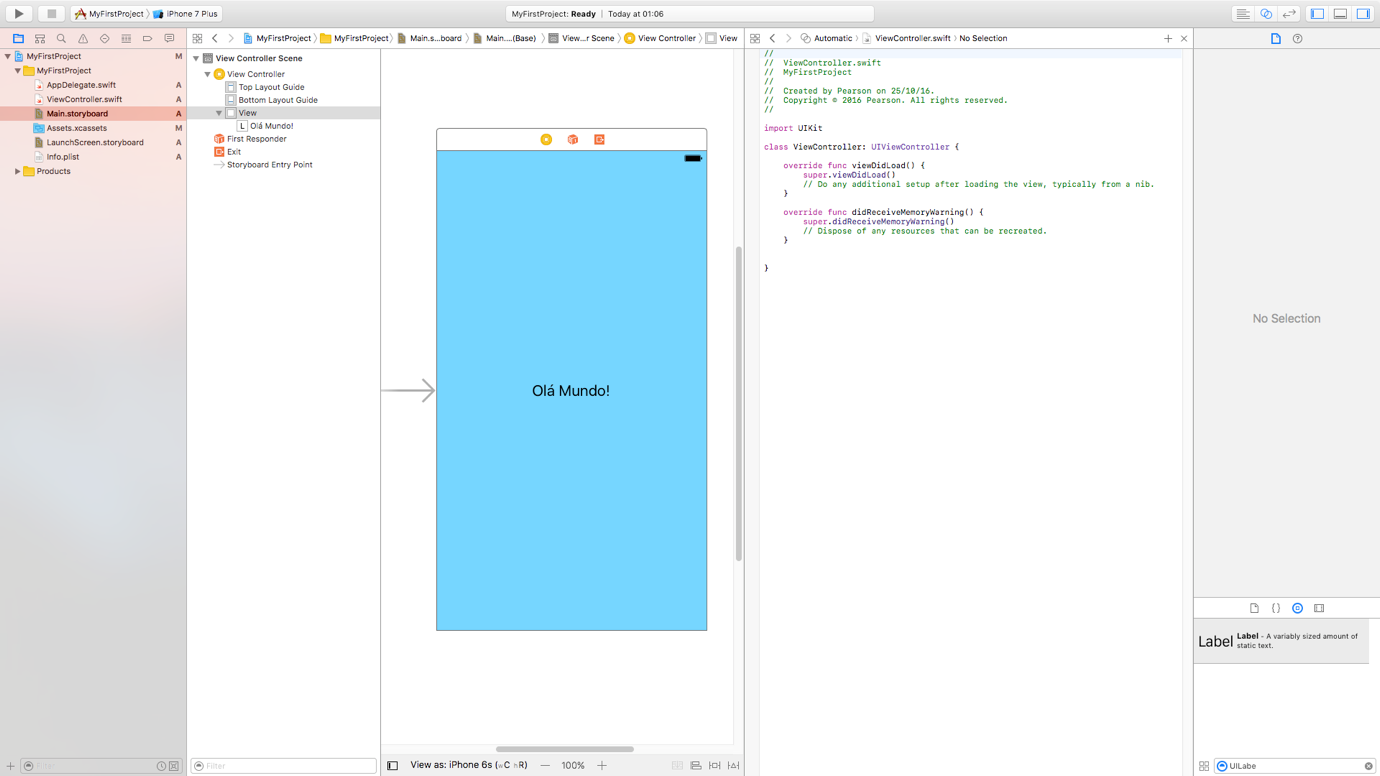


Figura 25 - Tela do Xcode

### Navegação do Projeto

No painel de navegação do projeto temos acesso a estrutura de arquivos e diretórios adicionados ao projeto. Ele está organizado em forma de árvore de diretórios, sendo que os diretórios de cor amarela são grupos e não são diretórios contidos no seu projeto. Nas aulas seguintes iremos criar estes grupos e você poderá ver a diferença. Os grupos são mais uma forma de organizar seu projeto sem interferir na organização do diretório do seu projeto. Já as pastas em azul refletem os diretórios reais do seu projeto. Se arrastarmos arquivos para o painel de navegação podemos copiá-los ou referenciá-los ao projeto.

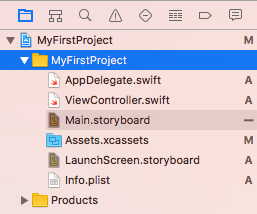


Figura 26 – Seção do Xcode de Navegação do Projeto

### Editor

O editor é o local onde colocamos nosso código. Nele existem várias funcionalidades que aumentam a produtividade do desenvolvedor, como:

* **auto-complete**, que completa o código que está sendo digitado automaticamente;
* **sugestões**, que sugere trechos de código ao desenvolvedor à medida que se digit;,
* **pré-build** que detecta erros e inconsistências de compilação e sintaxe antes mesmo de o desenvolvedor executar o código (fazer uma build), e etc.

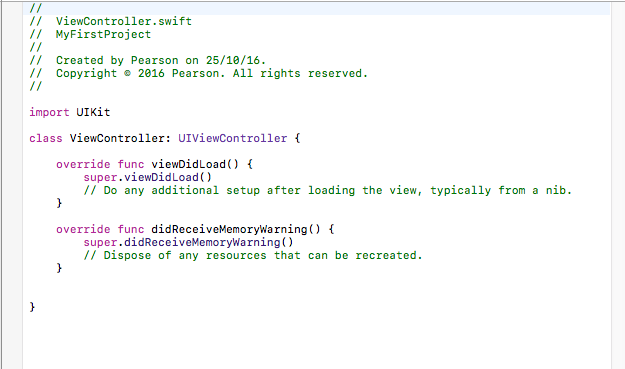


Figura 27 - Seção do Xcode de Edição do código

### Assistant Editor

Com este assistente podemos dividir o nosso editor em dois. Com esta divisão podemos apresentar o código de duas classes diferentes, por exemplo. Além disso podemos mostrar arquivos relacionados entre si, como por exemplo, um layout e sua respectiva classe. Podemos também fazer controle de versionamento, exibindo as diferenças de código de uma versão para outra.

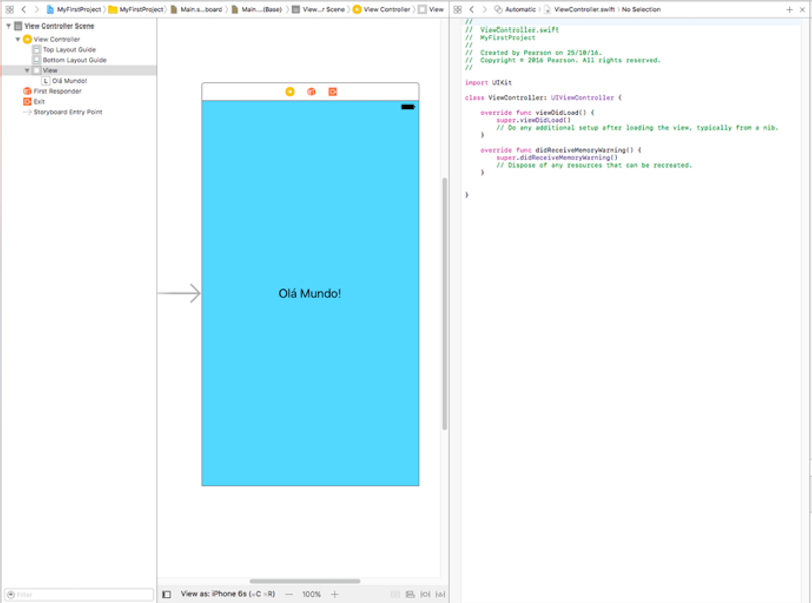


Figura 28 - Assistant Editor exibindo um Interface Builder e uma classe Swift lado-a-lado

### Jump Bar

A Jump Bar é localizada no topo do nosso Editor, e clicando nela, você pode rapidamente modificar o que será visto no Assistant Editor. Por exemplo, quando estamos editando um código no Assitant Editor principal, podemos selecionar através da Jump Bar o que será exibido no Assitant Editor secundário, como o counterpart header (cabeçalho homólogo) da classe sendo editada ou o Interface Builder (.xib ou .storyboard que veremos mais adiante) relacionado a ela.



Figura 29 - Jump Bar

### Interface Builder

Completamente integrado ao Xcode o Interface Builder torna a prototipagem e a construção de uma interface de usuário (UI) completa em uma tarefa simples. Com o Interface Builder podemos construir uma UI rica, sem ter que escrever uma linha de código, apenas com recursos Drag & Drop (arrastar e soltar). Diferente do Android, não precisamos escrever nossos layouts em linhas de código.

Prototipar em minutos, conectar graficamente sua interface com o código fonte no Editor, construir janelas, botões e controles deslizantes para criar uma interface de usuário funcional no Mac, iPhone ou iPad são facilidades que o Interface Builder nos proporciona. Com o Assistant Editor, você pode trabalhar no design gráfico lado-a-lado com o código fonte de implementação. Um simples arrastar do mouse de um controle de interface do usuário para o código de implementação cria uma conexão entre o código e a interface e pode até mesmo criar um código inicial para você.



Figura 30 - Interface Builder do Xcode

### Controle de Versão

Se você está familiarizado com algum software de versionamento de código como o Git ou SVN, vai gostar do módulo de controle de versão que o Xcode possui. Com ele você pode comparar duas versões, realizar commits, analisar o log de commits, empurrar e puxar código do repositório remoto, realizar blame do código e tudo que é essencial ao controle de versão.

### Teste

O Teste é uma fase do processo de desenvolvimento de software que garante a qualidade e confiabilidade do produto. A Apple dá muita importância para esta fase e por isso o Xcode possui as melhores ferramentas para testes dos seus apps iOS, macOS, watchOS e tvOS.

### Abra Rapidamente

Consiga acesso rápido aos arquivos do seu projeto com o comando **Open Quickly** (Command + Shift + O). O Xcode imediatamente irá mostrar uma lista de arquivos relacionada a sua busca, então você pode escolher o arquivo e pressionar a tecla Return para abrir o arquivo rapidamente.

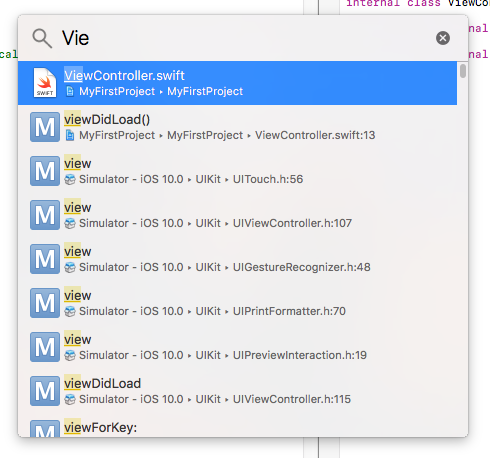


Figura 31 - Tela do Open Quickly

### Debug e Console

No painel de Debug podemos assistir as variáveis do escopo em debug. No console é apresentado os logs do aplicativo como descrições de exceções e prints. No console é possível também executar comandos para debug.

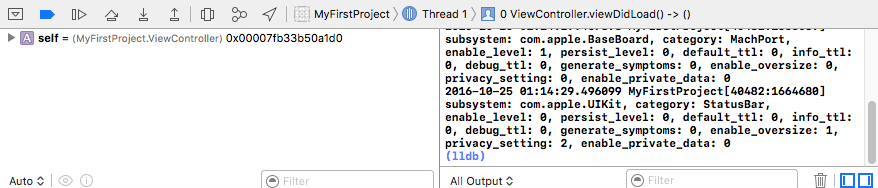


Figura 32 – Seção do Debugger (na esquerda) e do console (na direita)

### Propriedades

O painel de propriedades mostra as características do arquivo selecionado. Nele podemos configurar uma série de opções do arquivo como, local no diretório. Também é possível fazer configurações de classes e componentes selecionados no Interface Builder como, nome da classe, rótulo e tamanho de um botão, etc.

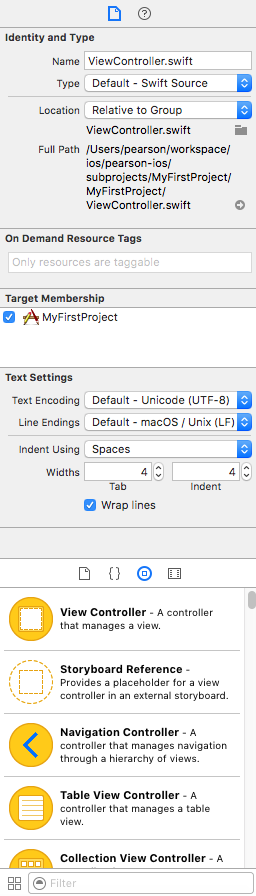


Figura 33 - Seção de Propriedades do arquivo/elemento

### Paleta

Na paleta podemos encontrar os **componentes gráficos** para nosso Interface Builder, como o UIButton, UILabel e UIView, que falaremos nas unidades seguintes. Também é possível encontrar as imagens do projeto, as chamadas **Assets**. Por fim, na paleta podemos encontrar e definir **snippets**, que são trechos de código prontos que utilizamos para agilizar nossa vida como o esqueleto da definição de uma classe ou de um laço **for.**

Estes são os componentes principais do Xcode. Alternativamente você pode encontrar estas opções através da barra de menus do Xcode. Estas e outras coisas serão explicadas com mais detalhes no decorrer do curso conforme a necessidade.

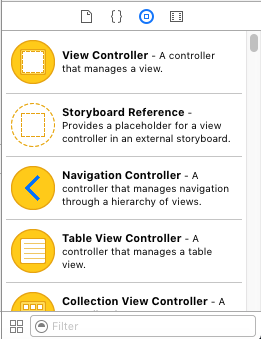


Figura 34 - Seção da Paleta de elementos

## Swift 3

A linguagem atual utilizada para desenvolver apps iOS é a Swift e neste curso iremos explorar a versão 3 desta linguagem. As versões anteriores ao Xcode 8 não são compatíveis com o Swift 3, então para criarmos apps com o que há de mais novo em recursos do iOS 10 e facilidades de desenvolvimento, a melhor escolha é o Xcode 8.

Nas aulas seguintes iremos explorar com mais detalhes a linguagem Swift, mas para matar a curiosidade, vejamos algumas diferenças comparadas com Java.

* **Hello World**

Para definirmos um Hello World em Java e Swift, respectivamente, precisamos de:

**Java:**  
  
 public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, world!");

}

}

**Swift:**  
  
 print("Hello World")

* **Definindo um método**

Para definir um método que converte Int para String em Java e Swift, respectivamente, precisaríamos de:

**Java:**

public class MyClass {

public String stringFromInt(Integer value) {

return value.toString();

}

}

**Swift:**

public func stringFromInt(value: Int) -> String {

return String(value)

}

* **Definindo um atributo**

Para definir um atributo do tipo String em Java e Swift, respectivamente, precisaríamos de:

**Java:**

public class MyClass {

public String color = "Vermelho";

}

**Swift:**

let color = "Vermelho"

Estes são alguns exemplos das diferenças, mas note que em Swift não foi preciso criar uma classe para definirmos o método, o atributo ou para chamar a função **print()**, isto porque a linguagem atua também como uma linguagem de Script, que não precisa ser compilada, apenas interpretada.

Na prática a Swift é compilável, mas neste caso é tão robusta que se comporta como uma linguagem interpretada.

DICA: Como vimos no curso de Android, na (não seria há?) diferenciação de Java e Javascript. A linguagem interpretada é aquela que não é compilada, ou seja, a sintaxe da linguagem não é verificada antes da execução do programa, sendo assim os erros de sintaxe são pegos apenas quando o programa esbarrar nele em tempo de execução. Você verá que o Xcode apontará erros no seu código, mas isto é mérito do Xcode, e não da linguagem Swift.

Nas aulas seguintes exploraremos mais o Swift.

## Cocoapods x Carthage

Como as limitações para desenvolvimento de apps iOS são grandes, a comunidade de desenvolvedores é bem menor que a comunidade Android, mas isto não quer dizer que ela não é fortemente ativa. (o sentido do parágrafo está confuso)

Como boa prática, em iOS também devemos utilizar bibliotecas de terceiros para economizar tempo e garantir um código mais confiável quando se trata de um componente mais complexo. A comunidade iOS é forte neste sentido e disponibiliza bibliotecas de ótima qualidade e variedade.

Para instalarmos as bibliotecas de terceiros, a forma padrão é exportar (importar?) **frameworks** para o projeto, mas como são estáticos, acabam correndo o risco de problemas de compatibilidade. Nos projetos do Xcode não existe algo similar ao **Gradle** do Android que gerencia nossas dependências. Então temos que instalar manualmente uma ferramenta que resolva esta questão. As duas melhores são: **Cocoapods** e **Carthage**. Vejamos a vantagem de cada uma delas.

### Cocoapods

O Cocoapods é um gerenciador de dependências criado na linguagem Ruby e é utilizado para gerenciar as bibliotecas de um projeto Xcode. Hoje, por ser a mais antiga, é a mais utilizada no mercado, possuindo algumas vantagens e características:

* Cocoapods está entre nós desde que os apps iOS eram criados utilizando a linguagem Objective-C. Então dá suporte a bibliotecas escritas com esta linguagem.
* Para criar um **package manager**, basta criar um arquivo **Podfile** na raiz do seu projeto e inserir alguns comandos como “pod ‘NomeDoFramework’”, e o framework será assinado (o aluno saberá o que assinado? O professor falará sobre isso?).
* Executando o comando “pod install” todos os pods (nome dado as bibliotecas instaladas via Cocoapods) são instalados ou atualizados.

Seus pontos negativos são a criação de um novo arquivo de projeto com a extensão **.xcworkspace,** que pode causar problemas na hora de executarmos comandos do Xcode via **linha de comando**. Outro ponto negativo é a demora para instalar ou atualizar os pods.

A demora nas instalações dos pods é dada devido ao fato que o CocoaPods foi desenvolvido inicialmente para gerenciar dependências escritas em Objective-C, e como estamos trabalhando com Swift, o CocoaPods leva um tempo maior para otimizar os componentes para esta linguagem. Mas este não é o fato mais agravante. O CocoaPods exige que todos os componentes que ele agrega são testados e validados, então ao instalar as dependências ele roda uma bateria de testes nela, garantindo que o componente funcionará corretamente. Mas não se preocupe, pois estamos falando diferença de segundos de atraso em relação ao Carthage.

Como você pode ver, os malefícios do CocoaPods geram benefícios, então será ele que utilizaremos no nosso curso.

Realmente é um assunto muito complicado e fora do foco para ser tratado nesse material. Acho que esse assunto não é importante para o contexto do assunto foco deste tópico. Se puder colocar alguns pontos para que o professor explane superficialmente o que ocasiona a demora pode ser uma saída.

### Carthage

O Carthage faz algo similar ao Cocoapods, a diferença é que ele baixa bibliotecas estáticas e você deve inseri-las manualmente ao projeto (baixando do repositório, clicando e arrastando para o projeto e configurando algumas propriedades no projeto para aceita-las). Pela praticidade e fama, usaremos a Cocoapods neste curso.

### Instalando o Cocoapods

Instalar o Cocoapods é uma tarefa fácil. Iremos fazer isto seguindo o exemplo no site [cocoapods.org](http://cocoapods.org/).

O Cocoapods é construído em Ruby e pode ser instalado com o Ruby disponível nativamente no seu macOS X. Nos recomendamos utilizar esta versão nativa do Ruby.

1. Abra seu terminal. A instalação do Cocoapods necessita de acesso **sudo**, ou seja, de permissão de administrador.
2. Com o terminal aberto digite o seguinte comando:

sudo gem install cocoapods

1. Forneça sua senha e aguarde a instalação finalizar.

O Cocoapods está instalado. Para utilizá-lo em um projeto existente, execute os seguintes comandos:

1. Futuramente mostraremos como criar um novo projeto e você poderá voltar aqui para executar estes passos. Então, com o terminal aberto, navegue para a pasta do projeto existente (o comando **cd** pode ser utilizado para navegar entre diretórios via linha de comando).

cd caminho/para/a/pasta/do/projeto

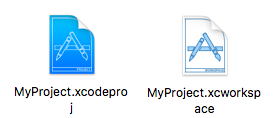
1. Agora precisamos inicializar o Cocoapods no projeto. Execute o comando:

pod init

1. Na prática o **init** apenas criou um **Podfile** para você. No decorrer do curso iremos utilizar este Podfile. Agora execute o seguinte comando:

pod install

O comando **install** irá configurar e preparar um projeto compatível com o CocoaPods. Veja que ele criou a pasta **Pods** e o arquivo de extensão **.xcworkspace.**

****

**A pasta Pods será onde todas as bibliotecas serão guardadas, enquanto que o arquivo de extensão .xcworkspace se torna o arquivo principal do projeto (que antes disto era o .xcodeproj). Então a partir do momento que você configura o CocoaPods no projeto (comando “pod install”) você deve utilizar somente o arquivo .xcworkspace.**

No decorrer do curso iremos falar mais um pouco do Cocoapods.

## Playground

No Android Studio, não temos um ambiente controlado para fazermos nossos experimentos como testar algoritmos ou realizar testes rápidos em uma biblioteca. Se quiséssemos tal feito, teríamos que executar um aplicativo no smartphone ou no simulador e sabemos que isto demora.

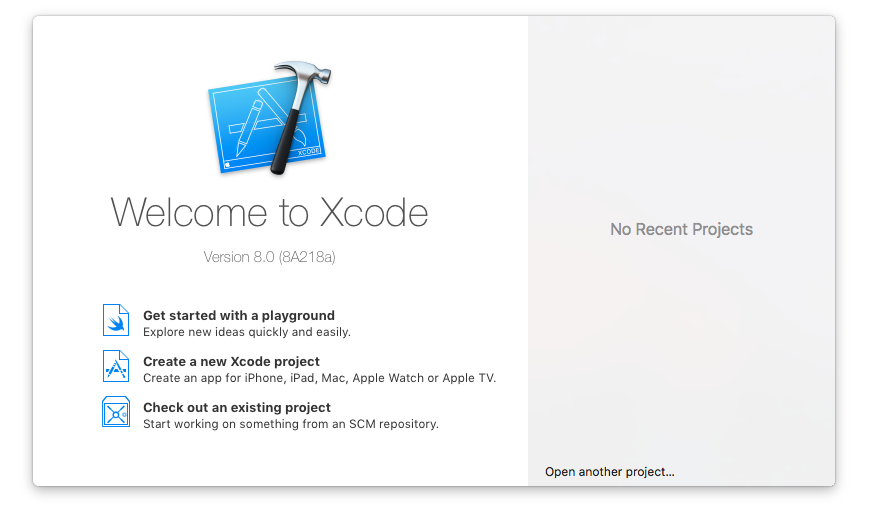
Quando o intuito é testar um trecho de código que não diz respeito a uma interface gráfica, o Xcode nos oferece o Playground.

Como o próprio nome sugere, o Playground é o local que utilizamos para brincar de programar. Este é o local mais aconselhado para aprendermos a programar em Swift, pois obtemos resultados rápidos e não necessariamente estruturados.

### Criando um Playground

Vamos logo criar um Playground pois ele será bastante utilizado na Unidade 2. Para isto execute os seguintes passos.

1. Abra o Xcode e a seguinte tela irá aparecer:



1. Clique na opção **Get started with a playground** (Inicie com um playground):



1. Alternativamente você pode criar um novo Playground utilizando o menu **File > New > Playground...** com o Xcode aberto.
2. Na nova tela que se abriu, escolha um nome para o seu Playground e clique em **Next**. Neste demos o nome **Swift\_Parte-1.**

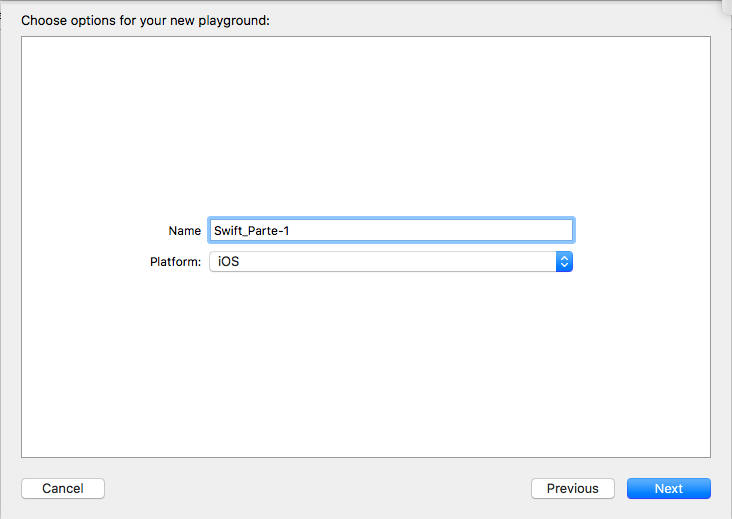


Figura 35 - Nomeando o Playground

1. Agora escolha o local onde você deseja salvar o arquivo playground.

Feito. Você já tem o primeiro arquivo em Swift.

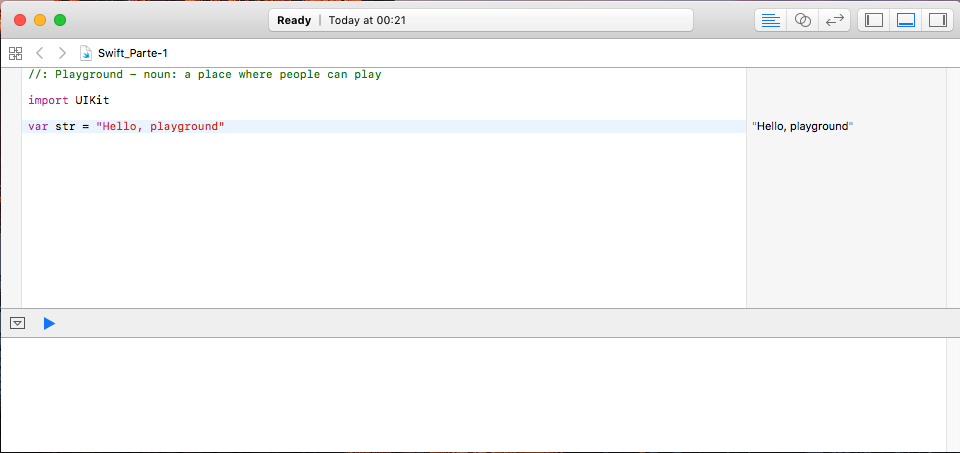


Figura 36 - Novo Playground criado

### Painéis do Playground

**Editor**

O editor do Playground é a nossa área de trabalho. É o local onde iremos codificar.

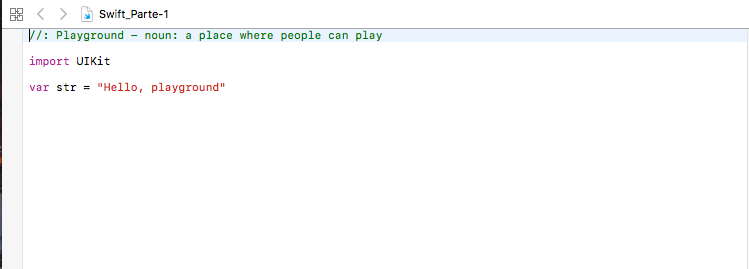


Figura 37 - Editor do Playground

**Console**

Resultados de prints e exceções são exibidos no console.



Figura 38 - Cosole do Playground

Logo acima do console temos dois botões: o **Ocultar/Mostrar Console** e o **Executar**, respectivamente.



Figura 39 - Botões Ocultar/Mostrar Console e o Executar, respectivamente

A cada nova instrução inserida o Xcode executa o código automaticamente e o resultado aparece na aba de **Preview**, mas você pode executar manualmente clicando no botão **Executar.**

**Preview**

Como o Xcode executa o código a cada modificação do arquivo, o painel Preview se mantém atualizado com os valores das variáveis e operações de cada linha em específico.

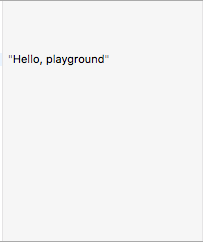


Figura 40 - Seção de Preview do Playground

## Hello World

Lembra da maldição do **Hello World?** Pois bem, parece que aqui não foi levada tão a sério. Vamos remover imediatamente a String “Hello, playground” e substituir pelo nosso “Hello World!”.

1. Remova todo o código do playground que acabamos de criar.
2. Agora insira a seguinte instrução:

print("Hello world")

Você vai ver que a String aparecerá no console. Ufa! Fugimos da maldição a tempo. Agora estamos mais do que preparados para explorar o Swift. Afinal você percebeu que não temos o **ponto e vírgula**?Vamos então estudar esta linguagem modernana próxima aula.

## Resumo

Nesta aula aprendemos como configurar nosso ambiente de desenvolvimento. Você pode notar que é bem mais simples do que instalar um ambiente Java, por exemplo. Introduzimos também alguns componentes principais do Xcode e do seu Playground. Com tudo isto tivemos a oportunidade de escrever nosso primeiro programa, o Hello World, que nunca foi tão curto.