Unidade 1

Aula 1

Vimos no curso de Android que a grande massa dos smartphones estão rodando o SO da Google. Mas por mais que o Android tenha esta vantagem mundial, outro sistema operacional mobile se destaca: o iOS. Aplicado em smartphones ponta de linha (e dos mais caros atualmente), o iOS ainda tem uma parte considerável do mercado, principalmente do norte americano. Apesar de possuir bem menos usuários do que o Android, desenvolver aplicativos para iOS acaba sendo mais rentável do que aplicativos iOS. Isto por causa dos diferentes perfis de consumidores. Enquanto os usuários de tendem a serem mais conservadores e optam por softwares gratuitos, os usuários de iOS tendem a ter um poder aquisitivo maior e estão dentro do grupo de usuários que pagam por aplicativos de melhor qualidade. Então o iOS é uma peça importante ao mercado mobile.

# Introdução

Sendo um dos pioneiros no ramo de sistemas operacionais para smartphone e também sendo o SO que popularizou o uso de smartphones, o iOS tem uma história breve, porém promissora.

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. Esse 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 e na hora certa, pois não há momento melhor que este para começar a desenvolver para iOS.

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 introduz ao desenvolvimento de *apps*, mas também oferece teoria e exercícios práticos suficientes para torná-lo um bom desenvolvedor *mobile*, este curso é o que você procura.

O curso de iOS explicará 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. 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 alguns requisitos.

Possibilidade de máquina virtual com SO apropriado ao Xcode.

1. **Um computador Mac:** um Mac é 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 opta por 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. Lamentavelmente, 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 seu valor. No ano de 2016, por exemplo, o custo era de 99 dólares. Sem essa 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 da marca.
3. **Download do Xcode:** uma vez que você se registrou como desenvolvedor da Apple, poderá baixar o Xcode pela AppStore. O Xcode é a IDE para desenvolvimento iOS, sobre a qual você aprenderá mais nas próximas aulas.
4. **Conhecimento em POO:** a linguagem *Swift*, adotada pela Apple para desenvolvimento dos seus *apps* e SOs (Sistemas Operacionais), é totalmente orientada a objetos, possuindo recursos modernos em relação a outras linguagens encontradas hoje em dia. Ter conhecimento em POO ajudará bastante neste curso. O módulo de POO que você realizou neste curso é mais do que suficiente para você se dar bem neste módulo de iOS.
5. **Foco e força de vontade:** no início poderá ser difícil fixar os conceitos. Você será levado a pensar diferente e será mostrada a filosofia por trás do *design* *mobile*. É 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* quando o mercado era dominado pelos telefones celulares simples.



Figura - iPhone 2G, o primeiro iPhone lançado

*(Fonte: http://www.blogtechsoeasy.com/wp-content/uploads/2014/09/iphone\_verge\_super\_wide.jpg)*

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, você vai 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 é dizer que o iPhone foi o pioneiro nesta tecnologia.

Como vimos no curso de Android foi o primeiro dispositivo a ser considerado *smartphone* foi o **IBM Simon**, que chegou às lojas em 1994. Ele possuía uma tela resistiva monocromática de 4,5 polegadas e seu sistema operacional era baseado em DOS (aquele mesmo que deu origem ao **Windows**), mas em vez de exibir uma linha de comando, mostrava uma interface dinâmica, baseada em atalhos.



Figura - Imagens do IBM Simon

*(Fonte: http://i1-news.softpedia-static.com/images/news2/Did-You-Know-The-First-Modern-Smartphone-was-the-IBM-Simon-470537-5.jpg)*

### A poderosa Apple Inc

Antes de saber sobre iOS, você verá 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. A 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 e Apple Pay Brasil, são esses 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. Agora que você já sabe um pouco da criadora do iPhone, você vai 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 à 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 à 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 e 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 à internet via Wi-Fi e uma câmera de 2 *megapixels*, que naquela época era uma resolução inovadora.



Figura - Imagens do iPhone 3G

*(Fonte: http://i2.tudocdn.net/img/max\_width1000/id40784\_1.jpg)*

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* (termo do inglês, significa dispositivo em português), 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 e 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*, têm-se dois ou mais aplicativos sendo executados ao mesmo tempo e o usuário pode alternar entre eles, sem precisar inicializá-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 - Fotos do iPhone 4

*(Fonte: http://i2.tudocdn.net/img/width660/height660/id93684\_1.jpg)*

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 caía. 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 o da 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 essa versão veio o iOS 6.



Figura - Fotos do iPhone 5

*(Fonte: http://i2.tudocdn.net/img/max\_width1000/id58119\_1.jpg)*

Anunciado em setembro de 2012, o iPhone 5 foi um sucesso de vendas. No dia do seu lançamento, foram vendidas 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 dourada, 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 às do iPhone 5, mas seu acabamento era em plástico. Qual era o acabamento do outro? Não foi citado no texto.

**DICA**: O fato de o 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 - Fotos dos modelos do iPhone 5C

*(Fonte: http://www.imore.com/sites/imore.com/files/styles/large/public/field/image/2014/03/topic\_iphone\_5c.png?itok=z2U8jqc9*)

**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 numa época em que a internet móvel estava se iniciando. Com o passar do tempo, as pessoas ficaram cada vez mais integradas ao 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 - Fotos do iPhone 6 (à esquerda) e do iPhone 6 Plus (à direita)

*(Fonte: http://i2.tudocdn.net/img/width542/height797/id92164\_1.jpg)*

O iPhone 6 e o iPhone 6 Plus traziam **telas de 4,7 e 5,5 polegadas respectivamente.** Com os iPhones 6, 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* desses novos modelos não mudou em relação aos modelos anteriores, pois as versões **S** possuem apenas melhorias no *hardware* interno e não mudanças visuais. Embora melhorias de hardware e o **iOS 9** tenham sido inseridos, 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 lançou o iPhone SE.



Figura - Foto dos diversos modelos do iPhone SE

(Fonte: http://store.storeimages.cdn-apple.com/4973/as-images.apple.com/is/image/AppleInc/aos/published/images/i/ph/iphonese/select/iphonese-select-2016?wid=437&hei=498&fmt=png-alpha&qlt=95&.v=9xwkJ2)

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 iPhones 7 e 7 Plus possuem grandes novidades em relação a tudo que já foi visto até aqui.



Figura - Foto do iPhone 7

(Fonte: https://www.telstra.com.au/content/dam/tcom/personal/mobile-phones/product-catalogue/iphone7/iphone-7-mattblack-angle.png)



Figura - Foto do iPhone 7 Plus

(Fonte: https://www.starhub.com/content/dam/catalog/starhub-store-apple-iphone-7-plus-black-back.png)

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



Figura - Cores do iPhone 7

(Fonte: http://www.bigbox.ee/26820-thickbox\_default/nutitelefon-apple-iphone-7-32gb-black.jpg)



Figura - Cores do iPhone 7 Plus

(Fonte: <http://thumbs.picclick.com/00/s/MzU4WDU4MA==/z/HsYAAOSw8oFX0W7t/$/NEW-SEALED-iPhone-7-32GB-Black-SEPT-DELIVERY-_1.jpg>)

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 há 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 a 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. Isso 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 dela, e esse número é bem menor do que a gama que o Android atinge. Atualmente o iOS está presente nos iPhones, iPod Touch e iPads. Existem derivados do iOS que operam nos sistemas embarcados watchOS e tvOS, mas não são mais considerados iOS.

Analisando por esse 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 têm qualidade superior e mais garantida do que os do Android.

Do ponto de vista do desenvolvedor, é mais vantajoso que se especialize na plataforma iOS do que na Android. Seria ótimo aprender as duas, pois são raras as vezes que se encontram 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, ela é mais valorizada em relação à mão de obra para 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 essa discussão, e uma coisa é certa, ela nunca terá uma resposta.

Tanto Android quanto iOS têm 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 e essa 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, você vai entender e proteger a sete chaves esse princípio, mas primeiro é preciso 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 isso você deve conhecer as maiores expectativas que os usuários têm sobre um *app* iOS.

São três temas primários que diferenciam um aplicativo iOS dos 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, serão pontuadas outras dicas e sugestões de *design* que a Apple define.

## Resumo

Nesta aula você viu um pouco das vantagens do desenvolvimento *mobile*. Foi abordado que o desenvolvimento *web* vem perdendo espaço para os aplicativos *mobile*, então é vantajoso que se tire proveito dessa onda para aprender e lançar aplicativos novos. Você também conheceu um pouco sobre a história dos *smartphones*, como o IBM Simon, e a história revolucionária do iPhone.

Além disso, foi visto que o iOS não está sozinho no mercado. O gigante Android domina 84% dos *smartphones*, mas o iOS ainda possui sua reputação.

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

## TDP

## Exercícios

## TDP

Aula 2

Como primeiros passos para iniciar a jornada no desenvolvimento de iOS, é preciso preparar seu ambiente de desenvolvimento. Nesta aula você vai se registrar no *site* da Apple e instalar o ambiente de desenvolvimento Xcode direto da loja de aplicativos AppStore. Após ter preparado seu ambiente de desenvolvimento, você finalmente criará seu primeiro programa na linguagem Swift.

# Ambiente de desenvolvimento

Diferente do desenvolvimento Android, em iOS há 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 se tem ao desenvolver para iOS é que para publicar seus *apps* é preciso pagar uma licença no *Apple Developer Program*. Você verá esses 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 necessária para utilizar os serviços da Apple, inclusive para utilizar seu próprio sistema operacional. No curso você vai utilizá-lo para fazer *download* do Xcode (a IDE utilizada para criar seus *apps* iOS). Para publicar seus *apps* será preciso 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 desse valor no momento que estiver fazendo sua conta.

Como você está aprendendo, não será criada a conta neste momento, você pode criar quando começar a divulgar seus *apps*. No início você vai utilizar apenas a licença de teste. Se você não tem um ID Apple ainda, crie 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-mail* que você cadastrou.
3. Agora que você já tem o seu ID criado, configure sua Apple Store com essa nova conta. Sem ela não será possível baixar os aplicativos para Mac. Abra a **App Store.**

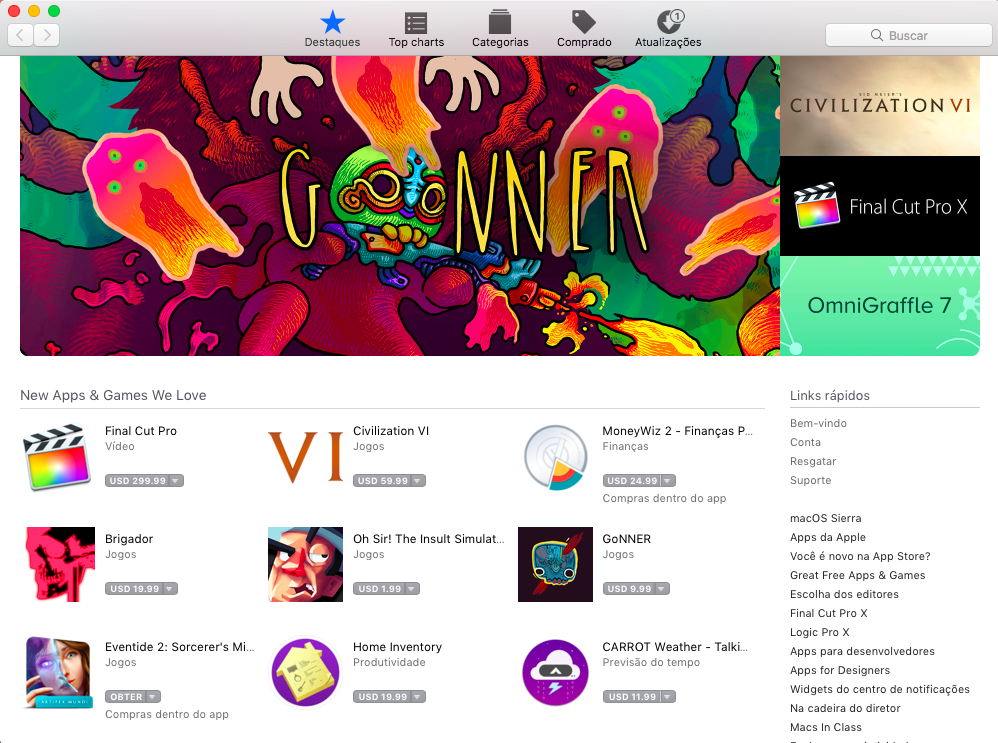


Figura - App Store

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



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

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



Figura - Tela de início de sessão da App Store

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

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

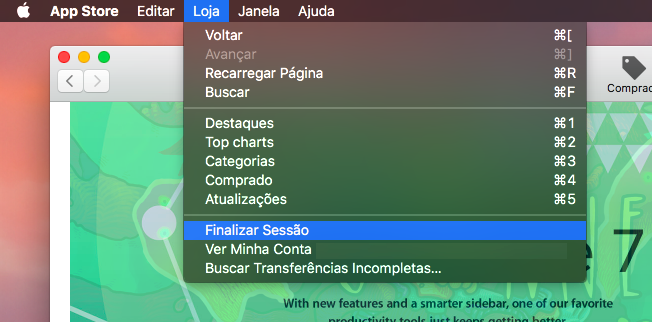


Figura - Opções mostradas quando o usuário já está com uma sessão iniciada

## Xcode

Tendo seu AppleID criado, a segunda coisa que você precisa é do Xcode.

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

Em comparação ao Android, o Xcode equivale ao Android Studio. O diferencial do Xcode é que ele é desenvolvido pela própria Apple e é compatível apenas com Macs, isso garante que a IDE funcionará 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 - Imagem institucional do Xcode

*(Fonte: https://developer.apple.com/xcode/)*

Infelizmente, o Xcode não vem instalado por padrão, então será preciso 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 há uma série de novas funções, como suporte para desenvolvimento no iOS 10 e outras coisas que serão vistas no decorrer do curso. Baixe então o Xcode 8.0.

Você pode executar os passos a seguir em sua casa caso queira instalar o Xcode em seu Mac pessoal. Como esse 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>>



Figura - Buscando Xcode na App Store

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

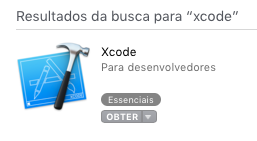


Figura - Clique em Obter



Figura - Clique em Instalar App

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

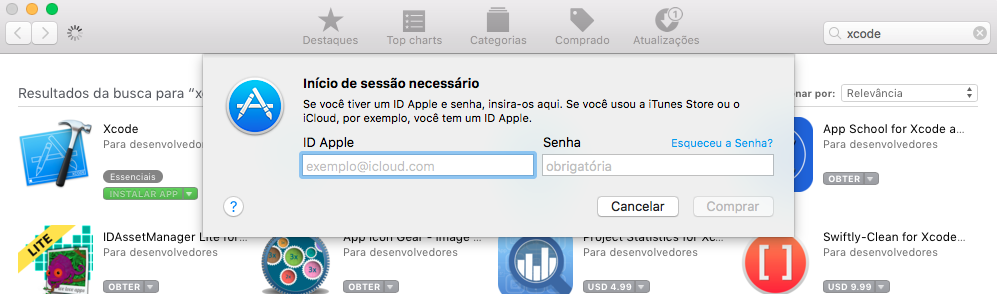


Figura - Autenticação na App Store

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* será baixado e instalado.

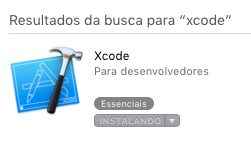


Figura - Instalação do Xcode iniciada

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



Figura - Launchpad



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

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



Figura - Xcode instalado e pronto para ser usado

Pronto. Seu Xcode está instalado. Para conhecer sua interface, veja os tópicos abaixo. Você ainda não vai criar um projeto, então apenas acompanhe este conteúdo.

## Conhecendo o Xcode

Como falado anteriormente, o Xcode é bastante integrado com o Mac, então seu trabalho flui naturalmente, sem travamentos ou surpresas. Existem algumas funcionalidades e painéis que se destacam. Veja-os a seguir.

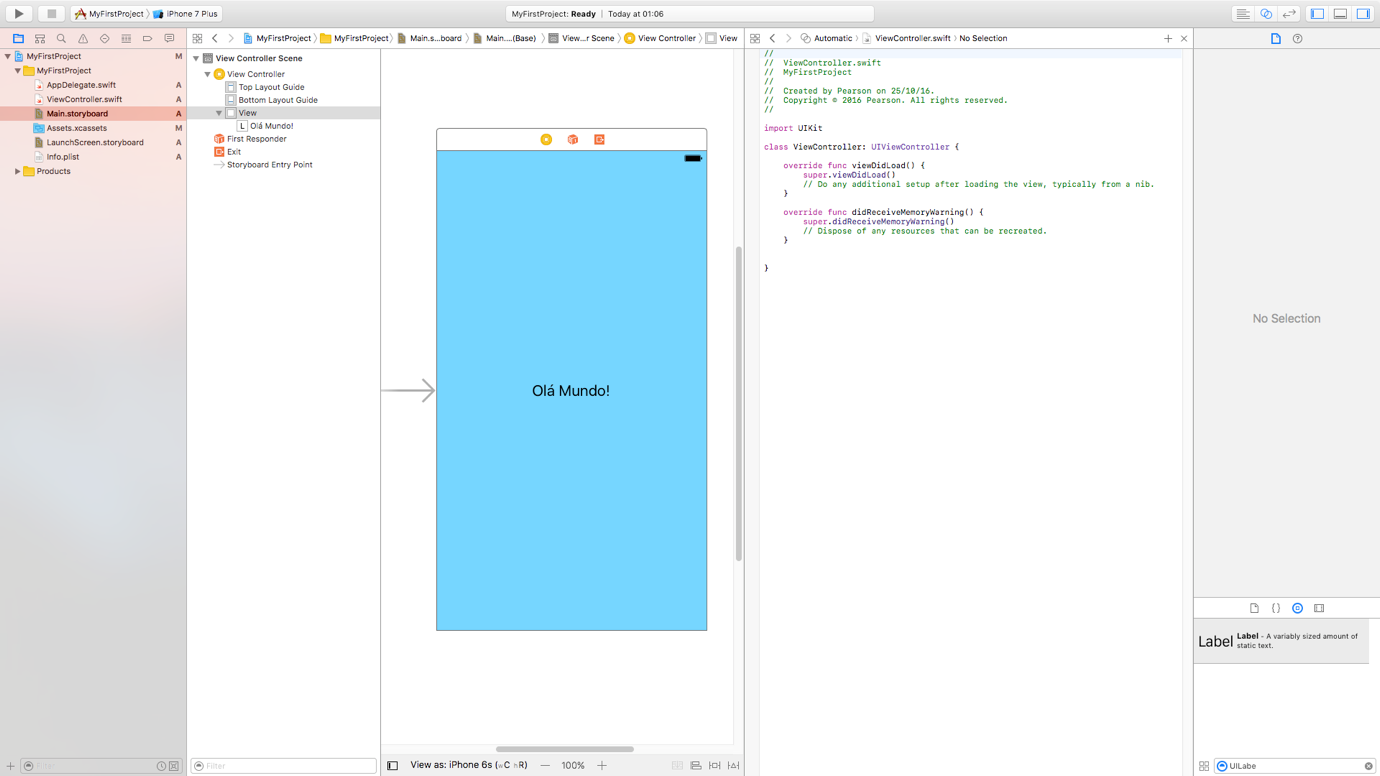


Figura - Tela do Xcode

### Navegação do projeto

No painel de navegação do projeto, você tem acesso à 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, você vai criar esses grupos e poderá ver a diferença. Os grupos são mais uma forma de organizar seu projeto sem interferir na organização do diretório dele. Já as pastas em azul refletem os diretórios reais do seu projeto. Se arrastar arquivos para o painel de navegação, você pode copiá-los ou referenciá-los ao projeto.

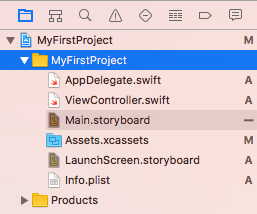


Figura – Seção do Xcode de navegação do projeto

### Editor

O editor é o local onde é colocado o seu 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 digita;
* ***pré-build*,** que detecta erros e inconsistências de compilação e sintaxe antes mesmo de o desenvolvedor executar o código (fazer uma *build*) etc.

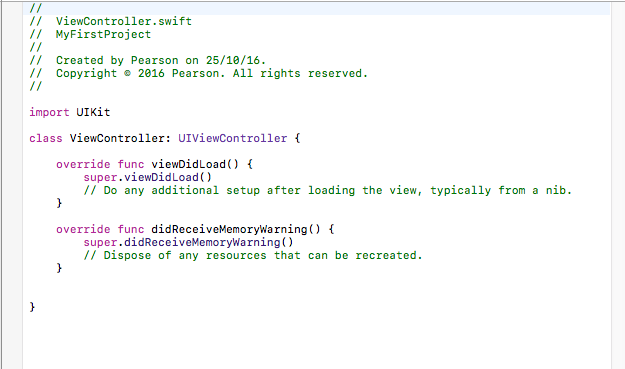


Figura - Seção do Xcode de edição do código

### *Assistant Editor*

Com este assistente, é possível dividir o editor em dois. Com essa divisão você pode apresentar o código de duas classes diferentes, por exemplo. Além disso, é possível mostrar arquivos relacionados entre si, como um *layout* e sua respectiva classe. Pode-se também fazer controle de versionamento, exibindo as diferenças de código de uma versão para outra.

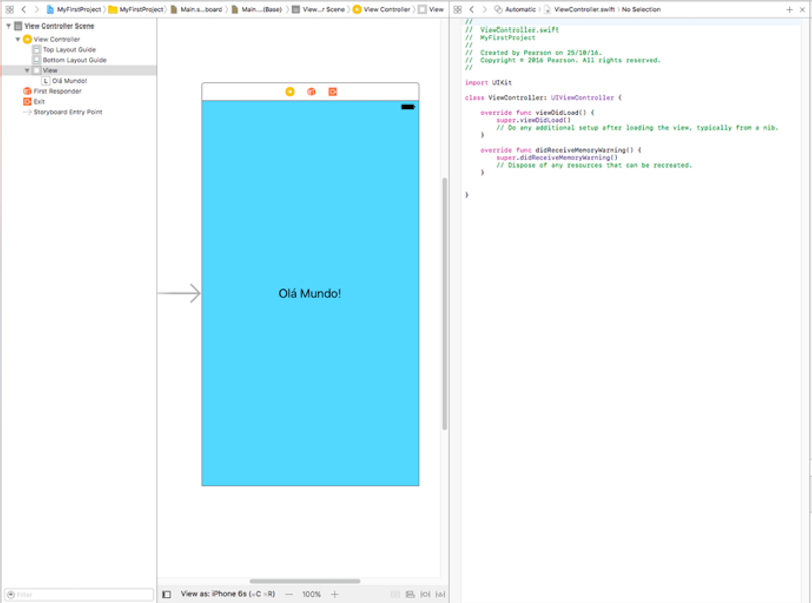


Figura - Assistant Editor exibindo um Interface Builder e uma classe Swift lado a lado

### *Jump Bar*

A Jump Bar é localizada no topo do seu Editor e, clicando nela, você pode rapidamente modificar o que será visto no *Assistant Editor*. Por exemplo, quando se está editando um código no *Assitant Editor* principal, é possível selecionar por meio 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 serão vistos mais adiante) relacionado a ela.



Figura - 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,* você pode construir uma UI rica, sem ter que escrever uma linha de código, apenas com recursos *Drag & Drop* (arrastar e soltar). Diferente do Android, não é preciso escrever seus *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* 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 - 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 o que é essencial ao controle de versão.

### Teste

O teste é uma fase do processo de desenvolvimento de *software* que garante a qualidade e a confiabilidade do produto. A Apple dá muita importância para essa 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 mostrará uma lista de arquivos relacionada à sua busca, então você pode escolher o arquivo e pressionar a tecla Return para abrir o arquivo rapidamente.

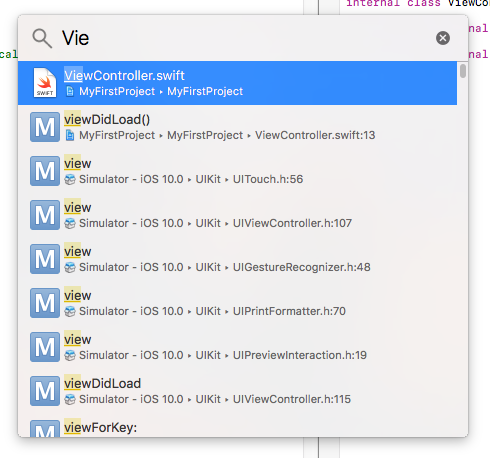


Figura - Tela do Open Quickly

### *Debug* e console

No painel de *Debug*, você pode assistir às variáveis do escopo em *debug*. No console, são apresentados os logs do aplicativo, como descrições de exceções e *prints*. No console é possível também executar comandos para *debug*.

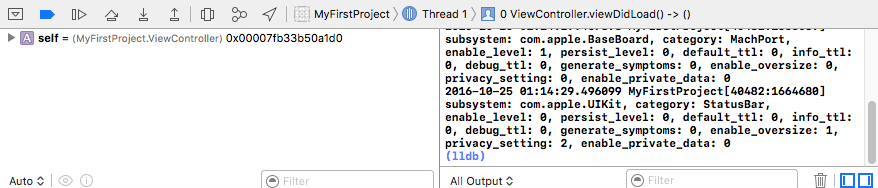


Figura – Seção do Debugger (na esquerda) e do console (na direita)

### Propriedades

O painel de propriedades mostra as características do arquivo selecionado. Nele é possível 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.

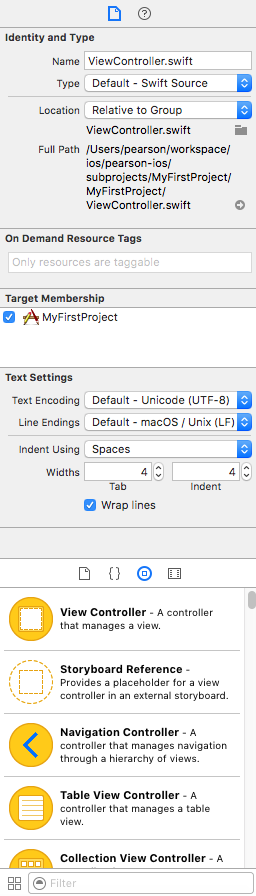


Figura - Seção de Propriedades do arquivo/elemento

### Paleta

Na paleta é possível encontrar os **componentes gráficos** para o seu *Interface Builder*, como o UIButton, UILabel e UIView, que serão abordados nas unidades seguintes. Também é possível encontrar as imagens do projeto, as chamadas ***Assets***. Por fim, na paleta você pode encontrar e definir ***snippets***, que são trechos de código prontos que são utilizados para agilizar a sua 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 por meio da barra de menus do Xcode. Estas e outras coisas serão explicadas com mais detalhes no decorrer do curso conforme a necessidade.

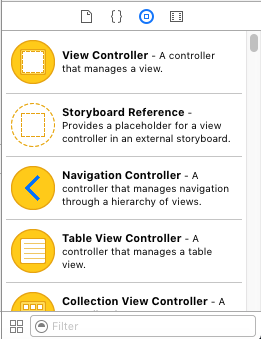


Figura - Seção da paleta de elementos

## Swift 3

A linguagem atual utilizada para desenvolver *apps* iOS é a Swift e neste curso você vai explorar a versão 3 dessa linguagem. As versões anteriores ao Xcode 8 não são compatíveis com a Swift 3, então para criar *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, você vai explorar com mais detalhes a linguagem Swift, mas para matar a curiosidade, veja algumas diferenças comparadas ao Java.

* **Hello World**

Para definir um Hello World em Java e Swift, respectivamente, você precisa 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 em String em Java e Swift, respectivamente, você precisaria 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, você precisaria de:

**Java:**

public class MyClass {

public String color = "Vermelho";

}

**Swift:**

let color = "Vermelho"

Esses são alguns exemplos das diferenças, mas note que em Swift não foi preciso criar uma classe para definir o método, o atributo ou para chamar a função **print()**, isso 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 visto no curso de Android, há diferenças 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 neles em tempo de execução. Você verá que o Xcode apontará erros no seu código, mas isso é mérito do Xcode, e não da linguagem Swift.

Nas aulas seguintes será explorado mais a Swift.

## *Cocoapods* x *Carthage*

Como as limitações para desenvolvimento de *apps* iOS são grandes (é necessário um Mac para programar e pagar uma taxa anual de $99,00, por exemplo), a comunidade de desenvolvedores é bem menor que a comunidade Android, mas isso não quer dizer que ela não é fortemente ativa.

Como boa prática, em iOS também devem-se 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 nesse sentido e disponibiliza bibliotecas de ótima qualidade e variedade.

Para instalar as bibliotecas de terceiros, a forma padrão é 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 suas dependências. Então é preciso instalar manualmente uma ferramenta que resolva essa questão. As duas melhores são: ***Cocoapods*** e ***Carthage***. Veja 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 o mais antigo, é o mais utilizado no mercado, possuindo algumas vantagens e características:

* Cocoapods está entre nós desde que os *apps* iOS eram criados utilizando a linguagem Objective-C. Então ele dá suporte a bibliotecas escritas com essa 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.
* Executando o comando “*pod install*”, todos os *pods* (nome dado às 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 executar comandos do Xcode via **linha de comando**, e a demora para instalar ou atualizar os *pods*.

A demora nas instalações dos *pods* se dá devido ao fato de que o *CocoaPods* foi desenvolvido inicialmente para gerenciar dependências escritas em Objective-C, e como se está trabalhando com Swift, o *CocoaPods* leva um tempo maior para otimizar os componentes para essa linguagem. Mas esse não é o fato mais agravante. O *CocoaPods* exige que todos os componentes que ele agrega sejam 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 se trata de diferença de segundos de atraso em relação ao *Carthage*.

Como você pode ver, os malefícios do *CocoaPods* geram benefícios, então será ele o utilizado no curso.

### *Carthage*

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

### Instalando o *Cocoapods*

Instalar o *Cocoapods* é uma tarefa fácil. Você vai fazer isso 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. Recomenda-se utilizar essa 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 ser finalizada.

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

1. Futuramente será mostrado como criar um novo projeto e você poderá voltar aqui para executar esses 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

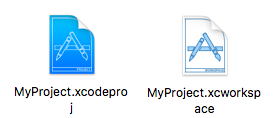
1. Agora é preciso inicializar o *Cocoapods* no projeto. Execute o comando:

pod init

1. Na prática, o **init** apenas criou um ***Podfile*** para você. No decorrer do curso você vai utilizá-lo. Agora execute o seguinte comando:

pod install

O comando ***install*** vai 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 e o arquivo de extensão .xcworkspace torna-se o arquivo principal do projeto (que antes disso 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 será falado mais um pouco do *Cocoapods*.

## *Playground*

No Android Studio, não há um ambiente controlado para fazer seus experimentos, como testar algoritmos ou realizar testes rápidos em uma biblioteca. Se você quisesse tal feito, teria que executar um aplicativo no *smartphone* ou no simulador e sabe-se que isso demora.

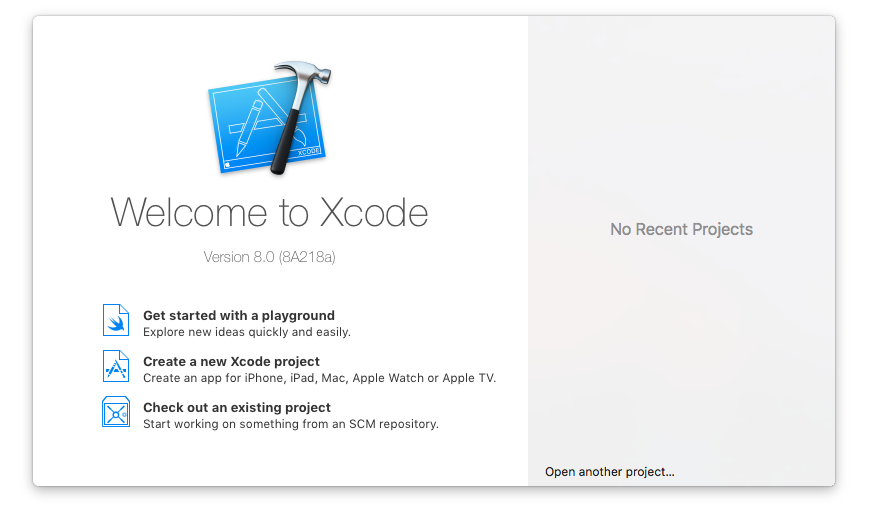
Quando o intuito é testar um trecho de código que não diz respeito a uma interface gráfica, o Xcode oferece o *Playground*.

Como o próprio nome sugere, o *Playground* é o local utilizado para brincar de programar. Esse é o local mais aconselhado para aprender a programar em Swift, pois são obtidos resultados rápidos e não necessariamente estruturados.

### Criando um *Playground*

Agora você vai criar um *Playground,* pois ele será bastante utilizado na Unidade 2. Para isso execute os seguintes passos.

1. Abra o Xcode e a seguinte tela aparecerá:



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 foi dado o nome de **Swift\_Parte-1.**

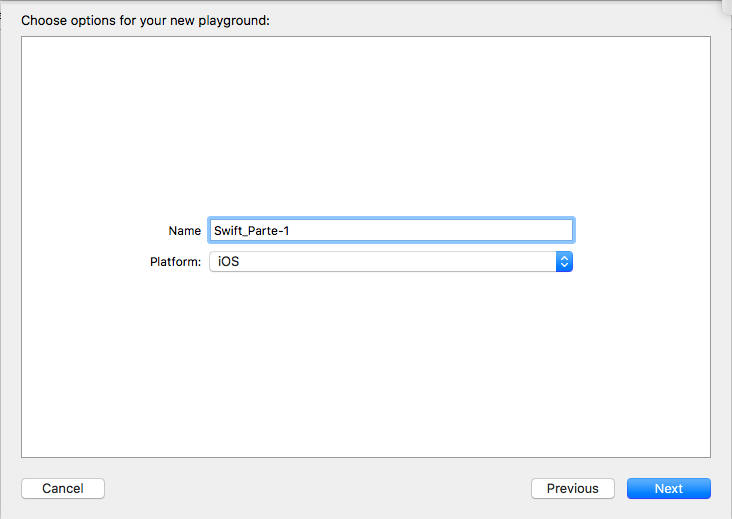


Figura - Nomeando o Playground

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

Feito! Você já tem o primeiro arquivo em Swift.

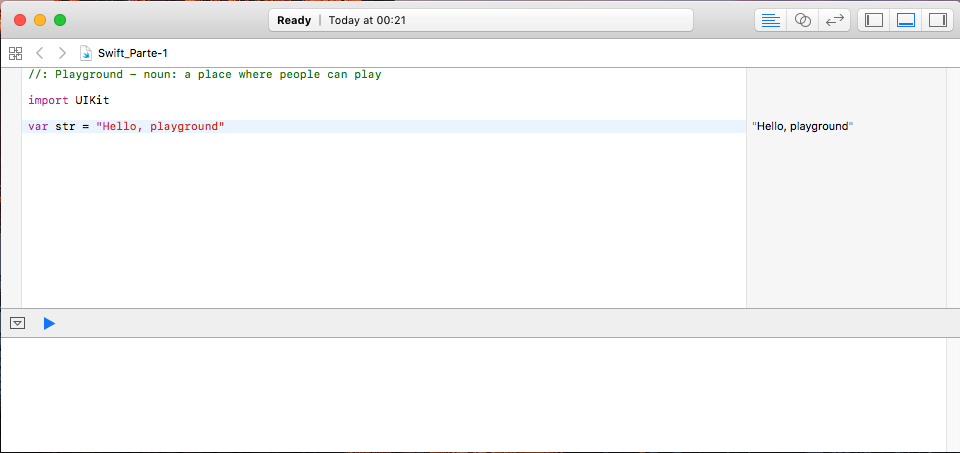


Figura - Novo Playground criado

### Painéis do Playground

**Editor**

O editor do Playground é a sua área de trabalho. É o local onde você vai codificar.

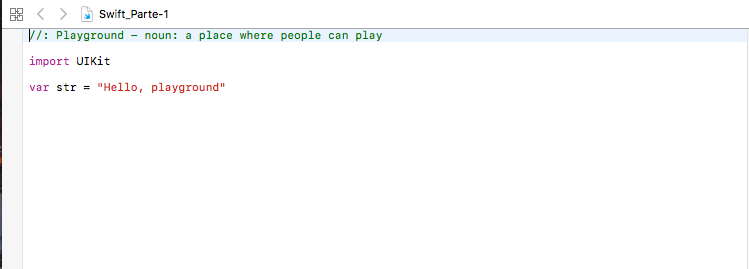


Figura - Editor do Playground

**Console**

Resultados de *prints* e exceções são exibidos no console.



Figura - Cosole do Playground

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



Figura - 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 das operações de cada linha em específico.

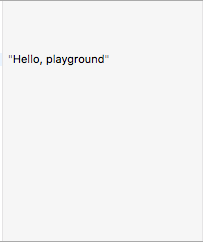


Figura - Seção de Preview do Playground

## Hello World

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

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

print("Hello world")

Você vai ver que a String aparecerá no console. Você fugiu da maldição a tempo. Agora você está mais do que preparado para explorar o Swift. Afinal, você percebeu que não há o **ponto e vírgula**?Você vai então estudar esta linguagem modernana próxima aula.

## Resumo

Nesta aula você aprendeu como configurar seu ambiente de desenvolvimento. Você pode notar que é bem mais simples do que instalar um ambiente Java, por exemplo. Foram introduzidos também alguns componentes principais do Xcode e do seu *Playground*. Com tudo isso, você teve a oportunidade de escrever seu primeiro programa, o Hello World, que nunca foi tão curto.

## Exercícios

1. Quais os requisitos para um desenvolvedor iOS?

2. Qual a empresa foi criadora do iPhone? Quem foi o criador do iPhone?

3. Em qual iPhone o sistema operacional foi batizado como iOS? Quais novidades que esse iPhone trazia?

4. Apresentavam telas maiores, de 4,7 e 5,5 polegadas respectivamente.

5. Qual a razão do barateamento do iPhone 5c?

6. Quais foram as melhorias no iPhone 6s e iPhone 6s Plus em relação aos anteriores?

7. Quais as grandes vantagens do iPhone 7 e iPhone 7 Plus em relação aos anteriores?

8. Qual a diferença entre o sistema operacional Android e o iOS?

9. Cite vantagens e desvantagens do sistema operacional Android e iOS.

10. Quais os temas primários que diferenciam um aplicativo iOS de outras plataformas?

## TDP