CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Desenvolvimento de Jogos Digitais

Bruno Guidolim

COMPARAÇÃO ENTRE MEIOS DE DESENVOLVIMENTO DE JOGOS PARA DISPOSITIVOS IOS

Americana, SP

2015

CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Desenvolvimento de Jogos Digitais

Bruno Guidolim

COMPARAÇÃO ENTRE MEIOS DE DESENVOLVIMENTO DE JOGOS PARA DISPOSITIVOS IOS

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Desenvolvimento de Jogos Digitais, sob a orientação do Prof. Kleber Andrade.

Área de concentração: Jogos Digitais

Americana, SP

2015

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

Guidolim, Bruno
G972c Comparação entre meios de desenvolvimento de jogos para dispositivos IOS. / Bruno Guidolim. – Americana: 2015.
33f.

Monografia (Graduação em Tecnologia em Jogos Digitais). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Me. Kléber de Oliveira Andrade

1. Jogos eletrônicos 2. Dispositivos móveis – aplicativos I. Andrade, Kléber de Oliveira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.6
681.519

Bruno Guidolim

COMPARAÇÃO ENTRE MEIOS DE DESENVOLVIMENTO DE JOGOS PARA DISPOSITIVOS IOS

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Desenvolvimento de Jogos Digitais pelo CEETEPS/Faculdade de Tecnologia – FATEC / Americana.
Área de concentração: Jogos Digitais

Americana, 30 de Junho de 2015.

Banca Examinadora:

Kleber de Oliveira Andrade (Presidente)
Professor Mestre
FATEC / Americana

José Mario Frasson Scafi (Membro)
Professor Mestre
FATEC / Americana

Marcos Francisco Pereira da Silva (Membro)
Professor Especialista
FATEC / Americana

DEDICATÓRIA

À Jussara, Carlos, Clarissa,
Família, apoio, amor, carinho, companheirismo e amizade.
À Larissa,
Minha companheira, meu porto seguro.
Ao professor Kleber,
Meu orientador pela dedicação e apoio.

RESUMO

O presente trabalho tem por objetivo demonstrar meios de desenvolvimento de jogos para a plataforma iOS a partir de duas ferramentas conhecidas no mercado de desenvolvimento de aplicativos para *smartphones* atualmente. Para efeito deste trabalho, as ferramentas escolhidas para comparação são: Xcode da Apple e Titanium Mobile da Appcelerator. Foi avaliado desde linguagem, tempo de desenvolvimento, custo e dificuldades durante o processo de desenvolvimento. Por fim, é apresentada uma conclusão e comparação dos resultados e motivos de escolha entre uma ou outra ferramenta de desenvolvimento considerando um estudo de caso. Tendo em vista a demanda por aplicativos para dispositivos móveis e a crescente disponibilidade de *smartphones* em nossa sociedade, este trabalho auxiliará o leitor na tomada de decisão entre qual ferramenta de desenvolvimento escolher de acordo com o produto que deseja desenvolver, seja um jogo ou um aplicativo qualquer.

Palavras Chave: iOS; Titanium; Xcode;

ABSTRACT

The present work aims to demonstrate ways of developing games for the iOS platform from two well-known tools in developing applications for smartphone market. For purposes of this paper, two tools are chosen for comparison: Apple's Xcode and Titanium Mobile from Appcelerator. It was appreciated from language, development time, cost and difficulties. Finally, a conclusion and comparison of results and reasons for choice between one or another development tool, considering a case study. Given the demand for mobile applications and the increasing availability of smartphones in our society, this paper will assist the reader in making the decision between which development tool to choose according to the product they want to develop, for a game or any application.

Keywords: *iOS; Titanium; Xcode*;

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CEO	Chief Executive Officer
CSS	Cascade Style Sheet
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
SDK	Software Development Kit
GPS	Global Positioning System

1 INTRODUÇÃO

Os computadores têm cedido cada vez mais espaço para os chamados *smartphones*, telefones celulares com funcionalidades avançadas, nas quais tais funcionalidades podem ser estendidas através do desenvolvimento de aplicativos.

No cenário global, em 2011, foram vendidos mais *smartphones* do que computadores pessoais (CANALYS RESEARCH, 2012).

Atualmente, existem diversas plataformas para dispositivos móveis disponíveis, e cada uma traz consigo um sistema operacional e sua própria maneira de lidar com recursos de *hardware*, como câmera, GPS[1], acelerômetro, etc (GARDNER e GRIGSBY, 2012).

Devido a incompatibilidade entre os sistemas, surge como uma necessidade e alternativa, o desenvolvimento de aplicações híbridas, que consiste em ter uma única base de código que, de maneira satisfatória, consegue comportar-se de forma similar em diversas plataformas (PAANANEN, 2011).

O presente trabalho tem por objetivo demonstrar meios de desenvolvimento de jogos para a plataforma iOS a partir de duas ferramentas conhecidas no mercado de desenvolvimento de aplicativos para dispositivos móveis atualmente. Para efeito deste trabalho, as ferramentas escolhidas para comparação são: Xcode da Apple e Titanium Mobile da Appcelerator.

O Xcode é a forma padrão de desenvolvimento para o iOS, disponibilizada pela própria Apple e o único meio de compilar aplicativos. É uma ferramenta gratuita de desenvolvimento, cuja linguagem utilizada é a Objective-C, porém obriga o desenvolvedor a utilizar o sistema operacional da empresa, o OS X (APPLE INC. (A), 2014).

O Titanium Mobile surgiu com o intuito de facilitar o desenvolvimento de aplicativos para programadores que já conhecem linguagens web, como HTML, CSS e JavaScript. Considerando especificamente a linguagem JavaScript, verifica-se que a mesma é amplamente difundida na Internet e de fácil acesso e aprendizagem. Além disso, permite a reutilização de código para geração de aplicativos em outras plataformas, como o Android da Google (APPCELERATOR INC., 2008).

---

[1] *Global Positioning System* – Sistema de posicionamento global

Esta ferramenta, por meio de seu próprio SDK[2], converte todos os objetos criados via JavaScript em objetos reconhecidos pelo Xcode, gerando assim componentes nativos ao aplicativo final. Sua utilização não isenta o desenvolvedor de ter o Xcode instalado em sua máquina para gerar tais aplicativos (POLLENTINE, 2011).

Tendo em vista as ferramentas em questão, o presente trabalho busca analisar a melhor forma de desenvolvimento de acordo com cada aplicativo, levando em conta seu escopo, requisitos funcionais, prazo de desenvolvimento e performance desejada. Para refletir sobre os resultados obtidos, é utilizado como fonte de pesquisas artigos, *blogs* e documentações oficiais de cada empresa.

O trabalho em questão está dividido em três capítulos que têm por função trazer aspectos relevantes para o entendimento e a discussão da temática. Desta forma, o primeiro capítulo, traz a origem e contextualização da plataforma de desenvolvimento, o iOS, bem como suas ferramentas Xcode e Titanium Mobile. Em um segundo momento, é comparado um mesmo jogo desenvolvido nas duas ferramentas, a fim de verificar aspectos considerados importantes como tempo de desenvolvimento, aprendizagem, desempenho e usabilidade. No terceiro momento é feito a retomada e, conjuntamente, considerações finais acerca do objetivo e tema escolhido.

[2] *Software Development Kit* - Kit de Desenvolvimento de Aplicativos

2 PLATAFORMA E FERRAMENTAS DE DESENVOLVIMENTO

Um sistema operacional, para que se torne atrativo ao público em geral, seja corporativo ou doméstico, precisa dispor da possibilidade de instalação de novos aplicativos, sendo assim, expandindo suas funcionalidades. Para que isso ocorra de forma organizada e controlada, as empresas desenvolvedoras desses sistemas operacionais disponibilizam seu próprio SDK de desenvolvimento. A partir disso, outras empresas optam por desenvolver ferramentas de desenvolvimento que trabalham de forma intermediária, permitindo que o desenvolvedor possa escolher a ferramenta que mais lhe agrada, inclusive a linguagem utilizada.

2.1 O iOS

Em 9 de Janeiro de 2007 aconteceu, a que se tornou, a maior apresentação de todos os tempos da Apple, anunciada pelo seu atual CEO e grande visionário da nossa época, Steve Jobs, aquilo que revolucionaria todo o mercado de dispositivos móveis (VOX MEDIA, INC., 2013).

Rodando em um núcleo parecido com o do OS X, baseado em UNIX[3], o iOS era originalmente chamado de “iPhone OS”, e surgiu juntamente com seu principal *hardware*, o iPhone (VOX MEDIA, INC., 2013).

O iPhone emergiu em uma época em que o mercado de dispositivos móveis eram comparados detalhe por detalhe devido a similaridade dos sistemas estabelecidos na época: Windows Mobile, Palm OS, Symbian e BlackBerry. Além disso, surgiu com uma discrepante falta de recursos, tais como: conexão 3G, multitarefas, suporte a aplicativos de terceiros, função de copiar e colar, anexar arquivos arbitrários em e-mails, entre muitos outros recursos que os concorrentes já ofereciam (VOX MEDIA, INC., 2013).

Apesar da falta destes e mais recursos, não impediram o iPhone de se tornar uma grande novidade e sucesso de vendas, pois ele foi o primeiro *smartphone* a ter toda sua usabilidade através de uma tela capacitiva e sensível ao toque e

[3] Sistema operacional portátil, multitarefa e multiusuário criado por Ken Thompson, Dennis Ritchie, Douglas McIlroy e Peter Weiner (THE OPEN GROUP, [s.d.]).

multitoques, mudando todos os paradigmas da época sobre como utilizar um celular. Todos os botões físicos foram removidos, permanecendo apenas um botão principal, controle de volume e um botão bloqueio/desligamento (VOX MEDIA, INC., 2013).

O dispositivo inteiro foi planejado para causar a melhor impressão e usabilidade ao consumidor, tais como gestos, simulação de inércia, acelerômetro, opção de zoom com o movimento de pinça com os dedos e mudança de tela através de sua posição horizontal e vertical (VOX MEDIA, INC., 2013).

A possibilidade de instalação, e consequentemente de desenvolvimento, de aplicativos surgiu em 2008, com a versão 2.0 do iPhone OS, juntamente com a loja de aplicativos, veio o SDK para o *iPhone OS*, permitindo assim que qualquer pessoa com conhecimentos de computação e programação pudessem desenvolver seu próprios aplicativos e disponibilizá-los na loja de aplicativos da Apple de forma gratuita ou paga (VOX MEDIA, INC., 2013).

Além de outras funcionalidades que a versão 2.0 trouxe para o usuário, a maior de todas foi a possibilidade de instalar aplicativos de terceiros, pois por meio da criatividade e competência, surgiram aplicativos que muitas vezes se mostraram melhor do que aplicativos que já vinham pré-instalados no iPhone OS e o que ajudou, e muito, a popularizar e estabilizar o iPhone, no mercado de dispositivos móveis (VOX MEDIA, INC., 2013).

Desde então, o iPhone OS continuou sendo atualizado, trazendo funcionalidades básicas que até então não existiam, como a possibilidade de copiar e colar, e melhorias significantes no uso de recursos, aumentando o tempo de duração da bateria (VOX MEDIA, INC., 2013).

O iPhone OS passou a ser chamado de iOS apenas na versão 4.0, visto que o sistema operacional não rodava somente no iPhone, e sim em demais dispositivos como o iPod Touch, e desde a versão 3.2, no novo tablet da Apple, o iPad, adaptado para ser executado de forma útil e agradável a nova tela do iPad (VOX MEDIA, INC., 2013).

Até sua versão 6, o iOS mantinha um padrão de *design* que consistia em trazer imagens e ícone parecidos com os itens reais do cotidiano do usuário, tais como o ícone e o aplicativo em si de câmera, ter uma lente e ser parecido com uma câmera de verdade, e o aplicativo de notas manter uma fonte e linhas, assim como nos blocos de anotação. Mas com a chegada a versão 7.0, visto que o usuário, após 6 anos de iOS, já havia se adaptado aos *smartphones* sensíveis ao toque e gestos

padrões, toda sua parte visual foi remodelada, ficando mais simples e seguindo a tendência de ícones e interfaces mais limpa que até hoje predomina. Ou seja, em botões que antes ocupavam mais espaço para realmente parecer um botão e incluir junto um texto de ação, agora passou a ser uma simples imagem ou apenas o texto de ação, facilitando inclusive questões de espaçamento e organização do espaço dentro de um aplicativo (VOX MEDIA, INC., 2013).

No que diz respeito às ferramentas Xcode e Titanium Mobile elas surgem com o intuito de ser o fator intermediário entre programador e sistema operacional, assim para efeito de entendimento, os subcapítulos trazem mais informações a respeito delas.

2.2 O Xcode

O Xcode é a principal ferramenta de desenvolvimento para as plataformas iOS e OS X, que possibilita gerenciar todo o fluxo de um aplicativo, ou seja, desenvolvimento, testes, otimização e submissão para a loja de aplicativos (APPLE INC. (A), 2014).

Sendo a principal forma de comunicação direta com o sistema operacional, qualquer outra forma de desenvolvimento legal de aplicativos precisa passar pelo Xcode, que também possui facilidades de compilação via linha de comando, no qual as demais ferramentas utilizam para fazer tal integração (APPLE INC. (A), 2014).

Sua interface é baseada em uma única janela, facilitando a visualização ampla dos desenvolvedores para todas as áreas que nela contém, podendo ser personalizada de acordo com a preferência de cada profissional.

O Xcode, assim como a maioria das ferramentas de desenvolvimento, possui um assistente de código, que diferencia palavras privadas, variáveis e chamadas de métodos, além de auxiliar na localização de nomes de propriedades para que não haja erros. Esse assistente também já executa algumas pequenas validações antes mesmo de compilar, verificando tipos de variáveis e sugerindo alterações (APPLE INC. (A), 2014).

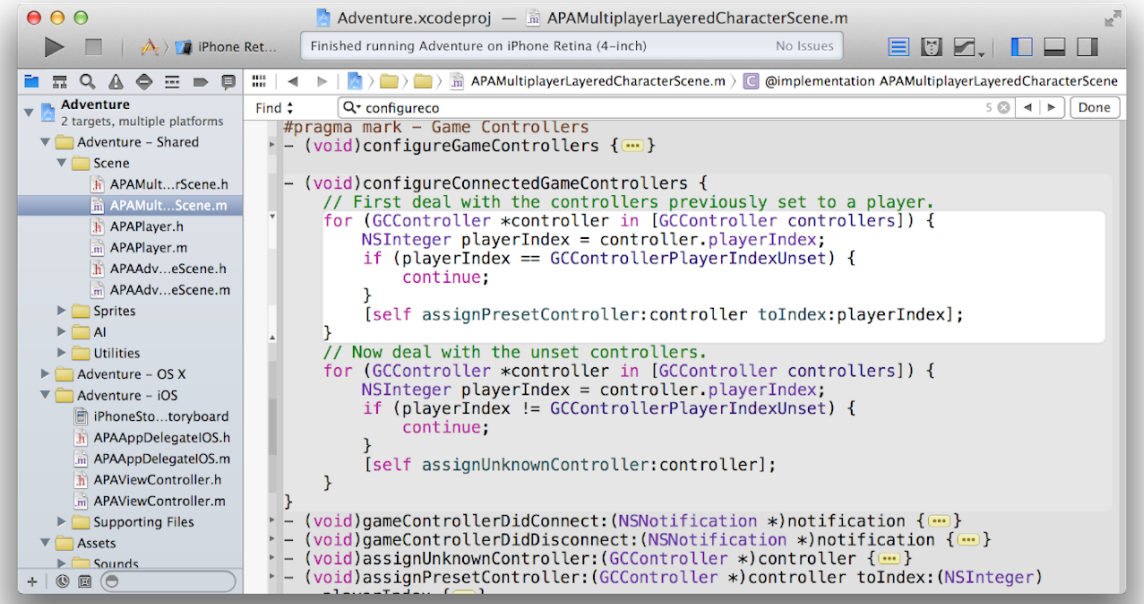

Figura 1 - Visão geral do Xcode (APPLE INC. (A), 2014)

Além do editor de texto e uma seção chamada de *explorer* (Figura 1), no qual é possível localizar todos os arquivos do projeto, padrão em qualquer IDE[4], o Xcode possui uma ferramenta que antes era separado, mas atualmente se encontra integrada, chamada *Interface Builder*, que permite o desenvolvedor agilizar a interface gráfica do aplicativo, sem precisar fazer toda sua configuração via código.

Por meio do *Interface Builder*, é possível conectar objetos gráfico ao código-fonte e definir regras de posicionamento para o caso de tamanho de telas diferentes ou para o caso do aplicativo suportar as duas orientações, vertical e horizontal, conforme apresentado na Figura 2.

[4] *Integrated Development Environment* – Ambiente de desenvolvimento integrado

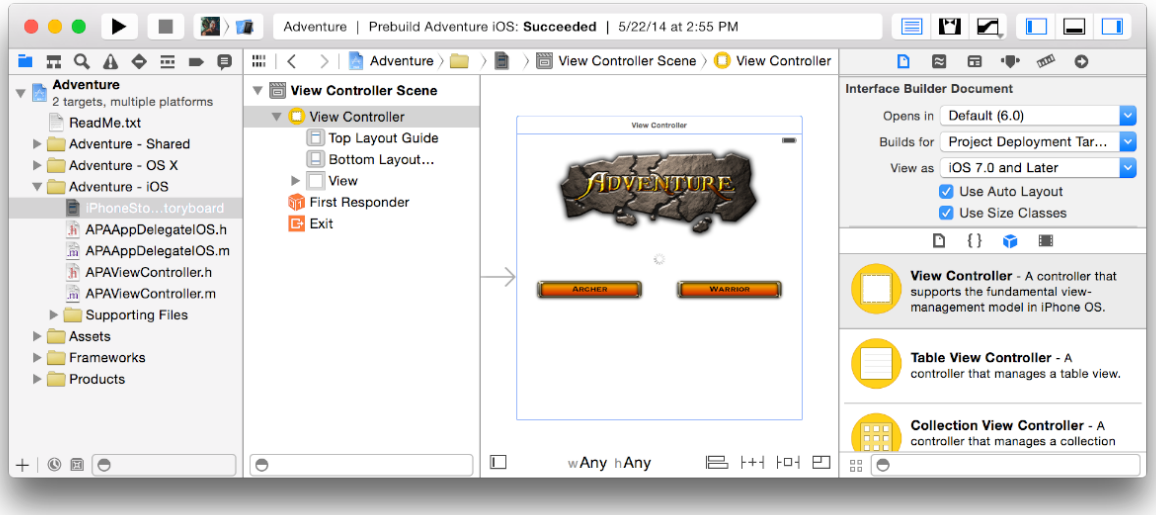

Figura 2 - Interface Builder (APPLE INC. (A), 2014)

2.3 Titanium Mobile

Titanium Mobile é uma ferramenta de código-fonte aberto desenvolvida pela Appcelerator, para o desenvolvimento híbrido de aplicativos, ou seja, é possível desenvolver versões de um aplicativo para plataformas distintas usando a mesma base de código.

Sua proposta é o desenvolvimento nativo através de um *framework*[5] que fica responsável de converter todo o código escrito em JavaScript em objetos nativos de cada plataforma, sem utilizar de técnicas de desenvolvimento web para conseguir tal feito, porém com a necessidade de aprender como o código deve ser estruturado para que seja possível tal conversão (APPCELERATOR INC., 2008).

Inicialmente, o Titanium era utilizado apenas para o desenvolvimento de aplicativos para sistemas operacionais de computadores comuns e foi introduzido ao mercado no final do ano de 2008 (PAANANEN, 2011).

A linguagem de programação JavaScript não foi escolhida por acaso. É uma linguagem altamente difundida na internet e com uma curva de aprendizado relativamente baixa por ser estruturada e não-orientada a objetos (W3SCHOOLS, 1999-2014).

Para que um aplicativo seja gerado pelo Titanium, é necessário que o SDK da plataforma escolhida esteja instalado no computador do desenvolvedor, uma vez

[5] Conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação (FAYAD e SCHMIDT, 1997).

que o Titanium se utiliza de toda a estrutura fornecida pelas empresas responsáveis pelo sistema operacional para compilar seu próprio código (SILVA, 2012).

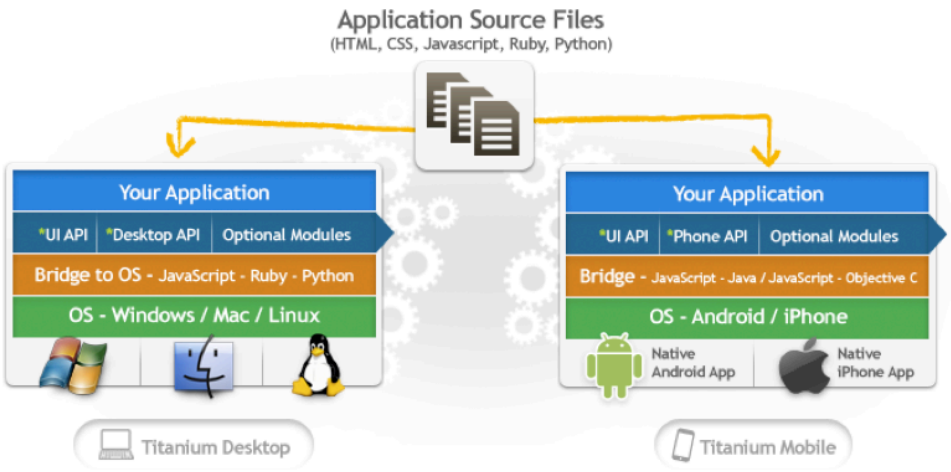

Figura 3 - Arquitetura ilustrada do Titanium (APPCELERATOR INC., 2008)

Conforme apresentada na Figura 3, a arquitetura do Titanium funciona da seguinte forma: o desenvolvedor escreve o código em JavaScript, essas chamadas ao *framework* do Titanium, converte internamente para objetos nativos da plataforma que se está utilizando, obtendo assim, como resultado final, um aplicativo nativo.

2.3.1 Titanium Studio

O Titanium Studio foi lançado no segundo trimestre de 2011 com o intuito de facilitar o processo de desenvolvimento. Antes de sua existência, o Titanium disponibilizava apenas de uma interface de listagem dos projetos com o único objetivo de compilá-los. Toda programação era feita através de arquivos de texto separados, usando a sintaxe JavaScript, no qual o desenvolvedor tinha livre escolha de qual editor utilizar (PEREZ, 2011).

Toda a *IDE* foi baseada em outra já existente chamada Aptana Studio, em sua versão 3.0, e a principal funcionalidade incorporada foi integrar o Titanium SDK ao editor de código em conjunto com um depurador de código (Figura 4), algo que antes era impossível de se fazer de maneira adequada, e mais opções de execução e empacotamento de aplicativos (MUSCHENETZ, 2011).

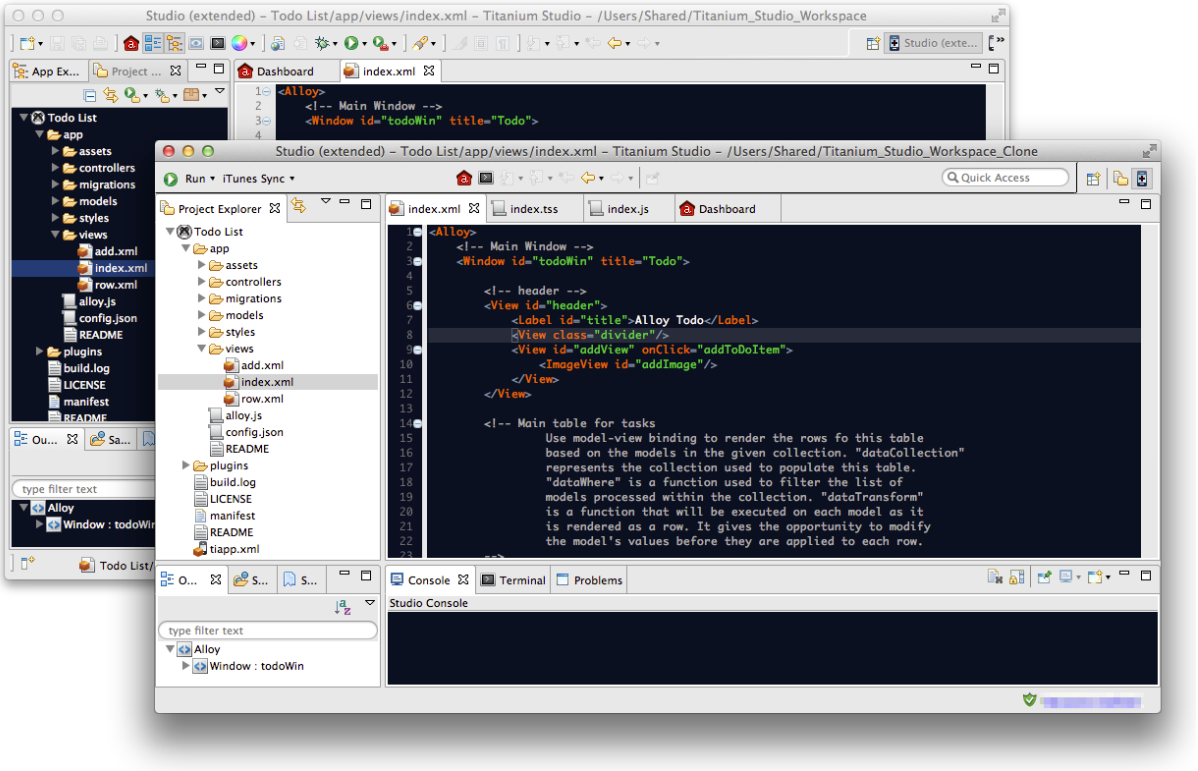

Figura 4 - Titanium Studio[6]

[6] http://goo.gl/5xMcl6 (Último acesso em 05 de Julho de 2015)

3 *GAME DESIGN DOCUMENT*: FLAPPY BIRD

A reflexão elaborada até então no presente trabalho, teve o propósito de abordar aspectos de desenvolvimento de um aplicativo simples na área de jogos, para efeito de entendimento, foi necessário discorrer sobre as duas ferramentas anteriormente citadas, Xcode e Titanium Mobile, uma vez que estas apresentam particularidades que devem ser levadas em consideração antes de iniciar o desenvolvimento.

O jogo escolhido foi o famoso *Flappy Bird*, que até o final de Janeiro de 2014 foi considerado o jogo gratuito mais baixado na loja de aplicativos da Apple (LOCAL WORLD LIMITED, 2014).

Para o desenvolvimento, primeiramente, foi realizado a atualização do *software* Xcode e Titanium Mobile e, a partir disto, foi criado o aplicativo nas duas plataformas de programação do iOS.

O sistema operacional utilizado foi o OS X 10.10 em um MacBook Pro e o iOS utilizado foi a versão 8.0.

3.1 História

O jogo não possui uma história com início, meio e fim. Por se tratar de um *endless game*[7], vamos considerar que o nosso personagem, o pássaro *Flappy,* simplesmente decide viajar para o leste por razões que desconhecemos.

O ambiente em que o personagem viaja é um dia ensolarado em campos verdes com uma cidade de fundo, porém existem canos espalhados no qual ele precisa se desviar para evitar que se machuque e interrompa sua jornada.

3.2 *Gameplay*

O ambiente do jogo é em 2D no formato *side-scrolling game*, ou seja, o personagem fica parado na tela e apenas o cenário que se movimenta, dando a impressão que o personagem está se movimentando.

[7] Jogos, normalmente em 2D e de rolagem lateral, que não possui um ponto final, continua sempre até que o jogador perca ou desista.

O objetivo do personagem é se desviar dos canos que aparecem da parte superior e inferior da tela, alinhados, deixando apenas um espaço livre entre eles por onde o personagem deve passar. Para cada passagem efetuada com sucesso, será contabilizado um ponto para o jogador.

O personagem começa a uma certa altura da tela, e será aplicado um efeito de gravidade para que ele chegue ao solo, ele precisa bater as asas para se manter sempre voando.

Por ser um *endless game*, não há condição de vitória, o mérito do jogador será alcançar o maior número possível de passagens entre os canos.

Caso o personagem se esbarre em algum dos canos, a partida é finalizada e serão apresentados os pontos que o jogador conseguiu naquela partida e a pontuação máxima já obtida no jogo.

Os canos aparecem aleatoriamente na tela, portando não há níveis de dificuldade, apenas a surpresa de qual posição os canos irão aparecer para atrapalhar o personagem.

3.3 Controles

O único controle do jogo serão toques na tela do dispositivo, a cada toque efetuado, o personagem bate suas asas e aumenta sua altitude. Vários toques permite que o personagem continue sempre em voo.

3.4 Universo

O universo do jogo é composto por um cenário misto de cidade e áreas verdes (Figura 5), e um solo (Figura 6) no qual terá tratamento de impacto caso o personagem o toque.

Serão adicionados canos (Figura 8) como obstáculo de forma aleatória para que o personagem desvie. Haverá também tratamento de impacto.

3.4.1 Dimensões

Figura 5 – Cenário	640x1136 pixels
Figura 6 – Solo	616x216 pixels
Figura 7 - Personagem Flappy - Bater das asas	68x48 pixels
Figura 8 - Canos (Obstáculos)	104x1136 pixels

3.4.1 Imagens

A imagem representada pela Figura 5, é o fundo (*background*) do jogo, na dimensão de 640x1136 *pixels*.

Figura 5 – Cenário

A imagem representada pela Figura 6, é o solo (*ground*) do jogo, na dimensão 616x216 *pixels*.

Figura 6 – Solo

A imagem representada pela Figura 7, é o personagem do jogo. Esta imagem é composta pelos três estados do personagem ao bater suas asas. A dimensão unitária de cada imagem é de 68x48 *pixels*.

Figura 7 - Personagem Flappy - Bater das asas

A imagem representada pela Figura 8, são os canos (obstáculos) do jogo. Esta imagem é composta pelos dois estados do obstáculo que pode ter origem pela na parte inferior da tela quanto na parte superior. A dimensão unitária de cada imagem é de 104x1136 *pixels*.

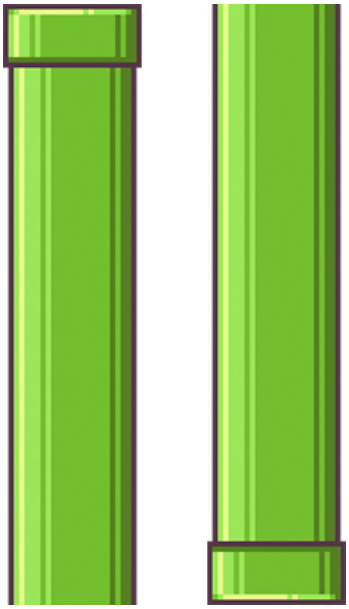

Figura 8 - Canos (Obstáculos)

A imagem representada pela Figura 9, são todos os elementos do jogo combinados para que tenha uma representação gráfica próxima a qual o jogo apresenta quando jogado.

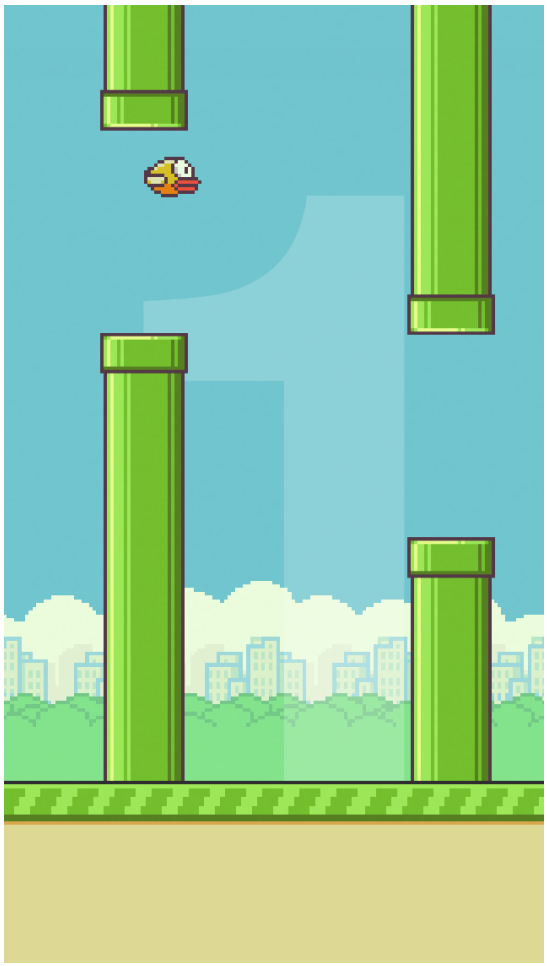

Figura 9 - Todos os elementos no cenário

4 ESTUDO DE CASO DO JOGO FLAPPY BIRD

Neste capítulo, foi analisado a forma de como o jogo foi desenvolvido usando as duas ferramentas em questão. Para melhor entendimento, este capítulo foi dividido em duas partes, disposto primeiro com códigos em Objective-C e posteriormente em JavaScript.

4.1 Análise de desenvolvimento com Xcode

Para o desenvolvimento com o Xcode, foi utilizada sua versão mais recente disponível ao público, a versão 6.0 disponibilizada em Setembro de 2014.

No que se refere à linguagem de programação Objective-C, é uma linguagem baseada na Linguagem C, porém orientada a objetos e dinâmica. Ela possui os tipos primitivos de dados, controle de fluxos e definição de classes e métodos, sendo considerada uma linguagem fortemente tipada (APPLE, INC. (B), 2014).

O código utilizado não foi desenvolvido especificamente para este trabalho, foi desenvolvido pelo programador Alexis Creuzot, em código aberto, podendo ser encontrado através repositório de códigos GitHub[8].

Assim como o jogo original, desenvolvido pelo programador vietnamita Nguyễn Hà Đông em menos de três dias (Hạ, 2014), o código de seu clone é relativamente simples, com poucas classes, sendo a maior delas com apenas 250 linhas.

O autor utilizou o *framework* SpriteKit[9], motor de jogos 2D padrão do iOS, disponibilizado a partir do iOS 7 no qual provê a infraestrutura de renderização de gráficos, animações e partículas, bem como laços de repetição e física básica para o desenvolvedor, ou seja, não foi necessária a utilização de bibliotecas externas para a conclusão do aplicativo.

Além das classes padrões disponibilizadas automaticamente pelo Xcode em seu modelo básico para projetos que serão desenvolvidos com o SpriteKit, foram utilizadas apenas cinco classes próprias, dois cabeçalhos de definição, são elas:

[8] https://github.com/kirualex/SpritYBird (Último acesso em 31 de Maio de 2015)

[9] *Framework* desenvolvido pela Apple para dar suporte a cálculos físicos, renderização e animação de imagens agrupadas (APPLE, INC. (C), 2014).

classe de pontuação, de cenário, do pássaro, uma classe com alguns cálculos matemáticos e outra para a rolagem do cenário.

Em suma, apesar da linguagem de programação ser um pouco mais difícil que as demais encontradas no mercado, o seu entendimento foi relativamente simples para quem detém algum conhecimento em programação.

4.2 Análise de desenvolvimento com Titanium Mobile

Para o desenvolvimento em Titanium Mobile, foi utilizada a versão mais recente disponível para o público de seu SDK, a versão 3.4.0.GA e, como citado anteriormente, a linguagem de programação utilizada é a JavaScript, porém com o SDK do Titanium, mesmo sendo uma linguagem estruturada, é possível trabalhar de forma parecida com uma linguagem orientada a objetos.

Não há nada parecido com o *Interface Builder* incorporado ao Titanium Studio, tornando seu desenvolvimento na camada gráfica mais trabalhoso, pois precisa ser criado via código de programação com suas definições básicas, como altura, largura e posicionamento, tornando o código um tanto extenso em alguns casos e há ferramentas de terceiros para a criação de interfaces gráficas, porém nenhuma completamente adequada e funcional ao Titanium Mobile ainda.

```
var lblName = Ti.UI.createLabel({
    text:'',
    top: 43,
    left: 20,
    width:280,
    color:'#A9A9A9'
});
self.add(lblName);
```

Figura 10 - Exemplo de criação de um objeto *Label* no Titanium Mobile

Assim como no Xcode, o código utilizado não foi desenvolvido especificamente para estre trabalho, foi elaborado pelos programadores Robert Bingham e Vitor Batista, em código aberto[10].

Dado o motivo de trabalhar como um intermediador, o processo de compilação pelo Titanium se torna um tanto mais lento, tanto quanto o modo de

[10] https://github.com/jankenpo/flappy-bird-clone/ (Último acesso em 31 de Maio de 2015)

depuração de código, quando necessário. Porém pouco se nota a diferença, que são altamente contornáveis com pequenos ajustes, na interface gráfica gerada pelo Titanium, que transforma o código em JavaScript em componentes nativos do iOS.

Diferentemente do Xcode, uma vez que no Titanium não há a definição de classes, todo o código do jogo foi criado em apenas um arquivo de 464 linhas, mas não há como fazer um jogo 2D com física sem a utilização de bibliotecas externas, e neste caso foi utilizado o *framework* Platino[11] da empresa Black Gate Games[12].

Por fim, o entendimento do código pode ser considerado médio, não por sua linguagem de programação, mas pela organização do código em um único arquivo, lembrando que as chamadas em JavaScript são assíncronas.

[11] *Framework* similar ao SpriteKit desenvolvido pela Black Gate Games para as plataformas iOS e Android, exclusivo para o Titanium Mobile.
[12] http://platino.io (Último acesso em 31 de Maio de 2015)

5 CONSIDERAÇÕES FINAIS

A partir do desenvolvimento deste jogo, considerado simples, conclui-se, como qualquer outra ferramenta de desenvolvimento, há suas particularidades, vantagens e desvantagens, mas que podem ser interessantes dependendo de cada situação.

O comportamento do aplicativo final acaba por se tornar praticamente o mesmo, precisando apenas de pequenos ajustes dependendo do *framework* de jogo utilizado, independente da maneira como foi desenvolvido. Portanto, o que define qual é a melhor ferramenta a ser escolhida se aplica justamente no processo de desenvolvimento que é realizado, tempo e recursos disponíveis, funcionalidades do celular que são utilizadas e comportamento do jogo ou aplicativo.

Em resumo, é possível destacar os seguintes fatores:

	Xcode	**Titanium Mobile**
Atualização	Por ser um produto da própria plataforma, está sempre atualizado quanto surge alguma nova funcionalidade no sistema operacional.	Por ser uma ferramenta de terceiros, sua atualização depende da empresa e/ou de desenvolvedores independentes que criar módulos.
Curva de Aprendizado	Objective-C torna a curva de aprendizado mais lenta.	JavaScript, uma linguagem altamente difundida, torna a curva de aprendizado mais rápida.
Custo Benefício	Disponibilidade de recursos humanos mais escassa e cara.	Facilidade em encontrar desenvolvedores que conheçam a linguagem, portanto é mais barato.

	Xcode	**Titanium Mobile**
Performance	Desenvolvedor interage sem intermediários com o sistema operacional, tornando o aplicativo final mais performático.	O *framework* do Titanium Mobile trabalha como intermediário, o que gera uma leve perda de performance e torna o processo obscuro para o desenvolvedor.
Desenvolvimento de Interface	Possui um criador de interface em sua IDE, tornando o desenvolvimento da parte visual mais simples.	Não possui um criador de interface, tendo que ser criada via código, gerando perda em produtividade.
Portabilidade	Por utilizar uma linguagem nativa da plataforma, há necessidade de reescrever todo o código.	Por existir o framework, boa parte do código pode ser reaproveitada para outra plataforma.

Se tratando do Xcode, ele possui vantagens de ser sempre atualizado em conjunto com o iOS, ou seja, toda e qualquer API[13] nova liberada pela Apple para o público final, automaticamente já se aplica ao Xcode para que todos os desenvolvedores possam ter acesso, diferente do Titanium, que módulos novos precisam ser desenvolvidos pela Appcelerator antes e só depois serão disponibilizados aos desenvolvedores. No mercado competitivo em que vivemos, isso pode se tornar desvantagem comercial.

Da mesma forma que a Appcelerator aplica novos módulos em seu SDK, é possível que um desenvolvedor experiente nas duas ferramentas, possa desenvolver seu próprio módulo nativo do iOS em Objective-C e distribuir para a comunidade de desenvolvedores em Titanium Mobile, para que possa ser utilizado por meio de chamadas em JavaScript, ou seja, mesmo que algum tipo de funcionalidade ou acesso a algum recurso do *smartphone*, como câmera ou

[13] *Application Programming Interface* - Interface de Programação de Aplicativos

acelerômetro, não estivesse disponível através do SDK do Titanium, é possível criar um módulo a parte para dar acesso a estes recursos através de um desenvolvedor independente.

O Objective-C, por ser uma linguagem orientada a objetos, fornece ao desenvolvedor todo o controle do código e classes dos componentes que nela existe, diferente do Titanium, que fornece apenas os comportamentos básicos com possibilidade de algumas adaptações. Qualquer mudança mais profunda de um objeto ou classe necessita da criação de um módulo novo para isso.

Por outro lado, ao se utilizar do JavaScript como linguagem de programação, que é altamente difundida e voltada para a Internet, sua curva de aprendizado é muito menor comparada ao Objective-C. Por ser uma linguagem exclusiva da plataforma da Apple, OS X e iOS, torna seu acesso mais difícil, uma vez que boa fatia do mercado ainda há predominância do Windows, da Microsoft (NET APPLICATIONS, 2014)[14]. Consequentemente, encontrar um profissional que conheça o processo de desenvolvimento em JavaScript, é mais fácil e financeiramente mais econômico do que encontrar que um profissional especializado em Objetive-C.

Um ponto muito importante que deve ser levado em consideração é a performance do aplicativo final, ambos os meios fornecem um resultado satisfatório, porém no Titanium Mobile, este processo não é tão transparente, não há garantias que todo o código de conversão para objetos nativos seguem as melhores práticas de desenvolvimento, portanto é preciso ter cautela no momento da escolha, pois aplicativos que podem demandar uma quantidade muito grande de processamento, como acesso a banco de dados local com muitos registros, o Xcode torna-se uma opção mais confiável.

Em relação à interface gráfica, o desenvolvimento com Xcode se torna mais atrativo por existir uma ferramenta de criação visual, tornando a pré-visualização do aplicativo possível, mesmo que não em sua forma final, sem a necessidade de compilação. Já o Titanium, apesar de possuir ferramentas de terceiros, que ainda não chegaram a um patamar aceitável, todo o desenvolvimento da interface se dá via código, desta forma, para que seja possível ter uma pré-visualização do aplicativo, a compilação do código será necessária.

---

[14] http://www.netmarketshare.com/operating-system-market-share.aspx (Último acesso em 31 de Maio de 2015).

Pode ser considerada uma vantagem para o Titanium, sua a facilidade em adaptação de código, pois, mesmo que escrito para que funcione especificamente para o iOS, poderá ser reaproveitado para que funcione em outro sistema operacional, como o Android por exemplo. Já um código desenvolvido com o Xcode, em Objective-C, precisa ser totalmente reescrito para que sua portabilidade seja possível em outro sistema operacional para dispositivos móveis.

Foi possível observar que, a escolha da melhor ferramenta de desenvolvimento para a plataforma iOS, depende muito da situação comercial e financeira do desenvolvedor ou empresa que pretende lançar um novo aplicativo. No quesito custo-benefício e possibilidade de migração de código para uma outra plataforma torna o Titanum Mobile uma opção mais atrativa. No quesito de experiência de usuário e utilização mais profunda de recursos disponíveis no iOS, o desenvolvimento nativo com Xcode se mostra uma opção mais interessante, principalmente por não haver um intermediador entre a linguagem e o sistema operacional.

REFERÊNCIA BIBLIOGRÁFICA

APPCELERATOR INC. **Appcelerator**, 2008. Disponivel em: <http://www.appcelerator.com>.

APPLE INC. (A). Xcode Overview. **Apple**, 2014. Disponivel em: <https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/chapters/about.html>. Acesso em: 19 Maio 2014.

APPLE, INC. (B). Programming with Objective-C. **Apple**, 17 Setembro 2014. Disponivel em: <https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>.

APPLE, INC. (C). About SpriteKit. **Apple**, 17 Setembro 2014. Disponivel em: <https://developer.apple.com/library/ios/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Introduction/Introduction.html>. Acesso em: 5 Julho 2015.

CANALYS RESEARCH. Smart phones overtake client PCs in 2011, 3 Fevereiro 2012. Disponivel em: <http://www.canalys.com/static/press_release/2012/canalys-press-release-030212-smart-phones-overtake-client-pcs-2011_0.pdf>. Acesso em: 7 Junho 2015.

FAYAD, M. E.; SCHMIDT, D. C. Object-oriented Application frameworks. **Communications of the ACM**, 40, n. 10, Outubro 1997. 32-38.

GARDNER, L. D.; GRIGSBY, J. **Head First Mobile Web**. [S.l.]: O'Reilly Media, 2012.
Hạ, Đ. Chàng trai viết game Flappy Bird gây sốt toàn cầu. **Thanh Niên**, 6 Fevereiro 2014. Disponivel em: <http://www.thanhnien.com.vn/pages/20140206/chang-trai-viet-game-flappy-bird-gay-sot-toan-cau.aspx>. Acesso em: 5 Julho 2015.

LOCAL WORLD LIMITED. As frustrated Stokies blast Flappy Bird game on Twitter we ask: What's your favourite smartphone app? **The Sentinel**, 31 Janeiro 2014. Disponivel em: <http://www.stokesentinel.co.uk/frustrated-Stokies-blast-Flappy-Bird-game-Twitter/story-20538032-detail/story.html>.

MUSCHENETZ, I. Titanium Studio 1.0 Preview with Titanium Mobile Debugging. **Appcelerator Blog**, 4 Abril 2011. Disponivel em: <http://www.appcelerator.com/blog/2011/04/titanium-studio-1-0-preview-with-titanium-mobile-debugging/>.

PAANANEN, T. **SMARTPHONE CROSS-PLATFORM FRAMEWORKS – A CASE STUDY**. [S.l.]: [s.n.], 2011. Disponivel em: <http://www.theseus.fi/bitstream/handle/10024/30221/110510_Thesis_Timo_Paananen.pdf?sequence=1>.

PEREZ, S. Appcelerator Launches Titanium Studio: Mobile, Desktop & Web Development in One. **ReadWrite**, 13 Junho 2011. Disponivel em:

<http://readwrite.com/2011/06/13/appcelerator-launches-titanium-studio-mobile-desktop-web-development-in-one>. Acesso em: 5 Julho 2015.

POLLENTINE, B. **Appcelerator Titanium Smartphone App Development Cookbook**. [S.l.]: Packt Publishing Ltd., 2011.

SILVA, P. L. M. **DESENVOLVIMENTO DE APLICATIVOS INTEROPERÁVEIS PARA DISPOSITIVOS MÓVEIS:** AVALIAÇÃO DE TECNOLOGIAS E IMPLEMENTAÇÃO. [S.l.]: [s.n.], 2012. Disponivel em: <https://docs.google.com/file/d/0B33wqGyN4nUIWDhuZkZCRUxPaFU/edit>.

THE OPEN GROUP. The UNIX System -- History and Timeline -- UNIX History. **The UNIX System, UNIX System**, [s.d.]. Disponivel em: <http://www.unix.org/what_is_unix/history_timeline.html>. Acesso em: 5 July 2015.

VOX MEDIA, INC. iOS: A visual history. **The Verge**, 16 Setembro 2013. Disponivel em: <http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad>. Acesso em: 27 Abril 2014.

W3SCHOOLS. JavaScript Tutorial. **W3Schools**, 1999-2014. Disponivel em: <http://www.w3schools.com/js/default.asp>.