Desenvolvimento de aplicações multiplataforma usando Javascript: Uma análise prática

Welker Arantes Ferreira, Marcel da Silva Melo

Instituto Federal Goiano – Campos Morrinhos (IFGoiano)
Morrinhos – Goiás - Brasil

welker3101@gmail.com, marcel.msmelo@gmail.com

Abstract. *The objective of this work is to present and analyze techniques and tools used in the development of hybrid multiplatform applications using technologies and tools aimed at web development. In the work an application is created for the main current platforms and aims to present advantages and disadvantages of the development of hybrid applications reusing the same source code for the portability of the application for several platforms.*

Resumo. *O trabalho tem como objetivo apresentar e analisar técnicas e ferramentas utilizadas no desenvolvimento de aplicativos multiplataformas híbridas utilizando tecnologias e ferramentas voltadas para o desenvolvimento web. No trabalho é criado um aplicativo para as principais plataformas atuais e visa apresentar vantagens e desvantagens do desenvolvimento de aplicativos híbridos reusando o mesmo código-fonte para a portabilidade da aplicação para várias plataformas.*

1. Introdução

O *smartphone* se tornou uma peça indispensável para a maioria da população brasileira. Com funcionalidades que vão além da ligação convencional, este dispositivo eletrônico se tornou o principal meio de acesso à internet no Brasil, segundo pesquisa do IBGE (2016).

Com o grande crescimento do uso de *smartphones* pela população e pelas grandes oportunidades que esse mercado vem oferecendo nos últimos anos, várias aplicações precisaram se adaptar a essa nova realidade. Aplicações e *sites* da *internet* precisam se adaptar aos diferentes tamanhos de telas e aos mais variados tipos de dispositivos para entregar o conteúdo passível de ser acessado pelo usuário.

Além de *sites* e aplicações web, há também um grande crescimento na demanda por aplicativos móveis (*apps*). Aplicativos móveis, ou *apps*, são sistemas indispensáveis aos *smartphones* e torna estes dispositivos ainda mais atrativos à população. Segundo Milani (2012), diversas empresas querem ter seus serviços e informações ao alcance dos clientes pelo *smartphone*.

Entretanto, desenvolver e manter um aplicativo para cada uma das principais plataformas atuais acaba sendo uma tarefa árdua (Bernardes e Miyake, 2016). Segundo Luquetti et al. (2015), o mercado de dispositivos móveis é ramificado por diferentes fabricantes e, por conta disso, existe uma gama de plataformas móveis. Segundo Martins et al. (2013), este é atualmente um dos principais desafios da computação móvel.

Isso se deve ao fato de que cada plataforma móvel possui seu conjunto de ferramentas de desenvolvimento, como linguagem de programação e ambiente de desenvolvimento (Bernardes e Miyake, 2016). Portanto, para que um aplicativo esteja presente em mais de uma plataforma é necessário desenvolver uma versão para cada plataforma utilizando seu respectivo conjunto de ferramentas de desenvolvimento. Esse

típico processo de desenvolvimento, chamado de desenvolvimento nativo, é um caminho apropriado para a criação de *apps* para cada plataforma. Porém, segundo Xanthopoulos e Xinogalos (2013), esse processo possui uma grande desvantagem: Não é possível reusar o código-fonte de uma plataforma para outra.

Segundo Bernardes e Miyake (2016), uma solução muitas vezes viável é utilizar a estratégia de desenvolvimento de aplicativos híbridos. O desenvolvimento híbrido consiste na criação de aplicativos móveis para várias plataformas reusando o mesmo código-fonte (Xanthopoulos e Xinogalos, 2013). Utilizando de tecnologias e linguagens padrões do desenvolvimento *web*, como o HTML, CSS e Javascript (W3C, 2017), o desenvolvedor que já tenha conhecimento e experiência em desenvolvimento *web* não precisará aprender novas linguagens para o desenvolvimento do aplicativo. Assim espera-se que acelere o tempo de desenvolvimento de aplicativos móveis e que esse processo de criação e manutenção do aplicativo seja facilitado.

O presente trabalho tem como objetivo apresentar e analisar técnicas e ferramentas utilizadas no desenvolvimento de aplicativos multiplataformas híbridas, utilizando tecnologias e ferramentas utilizadas comumente no desenvolvimento w*eb*. Para isso será criado um aplicativo híbrido voltado para as principais plataformas atuais, visando apresentar vantagens e desvantagens do desenvolvimento de aplicativos híbridos. Espera-se reutilizar o código-fonte o máximo possível para gerar o *app* para cada uma das plataformas testadas.

O trabalho está organizado da seguinte forma. Na Seção 2 são apresentados os trabalhos correlatos. Na Seção 3 é apresentado um referencial teórico sobre o desenvolvimento híbrido. Na Seção 4 é apresentado a metodologia para realização do trabalho. Na Seção 5 é apresentado os requisitos e o desenvolvimento do estudo de casos, além da portabilidade do sistema para cada uma das plataformas selecionadas. Por fim é apresentada a conclusão do trabalho, apontando os pontos positivos e negativos desta estratégia de desenvolvimento.

2. Trabalhos Correlatos

No trabalho de Xanthopoulos e Xinogalos (2013), os autores visam apresentar as principais tendências no desenvolvimento multiplataforma de aplicativos móveis. São apresentadas as principais abordagens para a criação de aplicativos móveis, visando basicamente os aplicativos *web* e híbridos. Os autores também realizam uma comparação entre as duas abordagens e apresentam pontos positivos e negativos.

No trabalho de Bernardes e Miyake (2016) é apresentado uma revisão sistemática sobre as diferentes abordagens usadas no desenvolvimento de aplicativos multiplataformas. Uma comparação entre as diferentes abordagens é realizada e são apresentados pontos fortes e pontos fracos de cada abordagem.

Palmieri et al. (2012) apresenta uma comparação entre as quatro principais ferramentas para o desenvolvimento de aplicativos móveis multiplataforma: Rhodes, PhoneGap, DragonRad e MoSync. No trabalho os autores apresentam cada uma das ferramentas selecionadas, apresentando as vantagens e desvantagens de cada uma.

O trabalho de Dalmasso et al. (2013) também apresenta um comparativo entre ferramentas para desenvolvimento multiplataforma, porém nesse trabalho outras ferramentas são utilizadas: PhoneGap, Titanium, JQuery Mobile e Sencha Touch. Os autores apresentam as ferramentas e as principais características de cada arquitetura. São apresentados exemplos de aplicações desenvolvidas usando PhoneGap e Titanium e são realizadas as comparações, onde os critérios avaliados foram: Uso de CPU, uso de memória e consumo de energia.

3. Desenvolvimento híbrido

A evolução dos *smartphones* abriu várias possibilidades, tanto para grandes empresas quanto para as *startups*. Entretanto, segundo Bernardes e Miyake (2016), essa evolução trouxe também o enorme desafio: portar os aplicativos para diversas plataformas. Para cada uma das principais plataformas do mercado é necessário implementar uma versão do aplicativo utilizando um kit de ferramentas de desenvolvimento específico, como SDK (*Software Development Kit*) que inclui uma linguagem de programação específica e uma API (*Application Programming Interface*) responsável por se comunicar diretamente com o dispositivo móvel. Também é necessário um ambiente de desenvolvimento, IDE (*Integrated Development Evironment*), específico.

Por exemplo, para desenvolver um aplicativo nativo para IOS, plataforma móvel utilizada nos dispositivos da Apple, como o iPhone e o iPad, é necessário dominar a linguagem Objective-C ou a linguagem Swift e a IDE utilizada no desenvolvimento é o X-code. No caso do Android, plataforma móvel da Google utilizada pela maioria dos dispositivos móveis do mercado, segundo o IDC (2016), é necessário dominar a linguagem Java e a IDE Android Studio.

Sendo assim, se uma empresa decide criar o mesmo aplicativo para as duas principais plataformas do mercado, IOS e Android, segundo o IDC (2016), usando a estratégia de desenvolvimento de aplicativos nativos, deverá criar dois projetos distintos com linguagens e características diferentes. Outro importante fator é o tempo despendido para aprender cada linguagem, onde cada uma delas possui características e recursos diferentes. Também pode ser necessário manter duas equipes de desenvolvedores, sendo uma equipe para cada plataforma. Essa duplicidade pode gerar funcionalidades diferentes em aplicativos que deveriam ser iguais.

Em contrapartida, a estratégia de desenvolvimento de aplicativos híbridos pode ser uma saída mais barata e bastante produtiva na maioria dos casos. Aplicativos híbridos encapsulam aplicações *web* construídas usando HTML, CSS e Javascript dentro de um *container* nativo (*UIWebView* no IOS ou *WebView* no Android) (Xanthopoulos e Xinogalos, 2013). O código-fonte do aplicativo gerado executa no navegador, como em aplicações *web*, porém todos recursos necessários para a execução do aplicativo final no dispositivo do usuário, navegador e código-fonte, é encapsulado no aplicativo final.

Usando a estratégia híbrida é possível criar um aplicativo como se fosse uma aplicação *web*, com a vantagem de permitir acesso a recursos nativos do *smartphone* como câmera, sistema de posicionamento global (*Global Position System*, GPS) e notificações. Além disso, utilizando tal estratégia, o aplicativo pode ser escrito uma única vez e o disponibilizado para diversas plataformas móveis, *web* e/ou *desktop*.

Atualmente existem diversas ferramentas voltadas para o desenvolvimento de aplicações híbridas que utilizam o conceito de reutilização de software, tanto para plataformas móveis quanto para plataformas *desktop*. Estas ferramentas criam uma camada extra que encapsula a aplicação e serve como ponte, permitindo a comunicação com o dispositivo e o acesso aos recursos nativos do mesmo.

4. Metodologia

Este estudo tem como método a pesquisa bibliográfica, por meio de revisão literária, juntamente a atividade prática, em torno da criação e análise da criação de um aplicativo multi-plataforma. Para realização da pesquisa bibliográfica, foram utilizados como base de dados livros, artigos científicos e sites na área de tecnologia e informação como fonte de publicações eletrônicas.

Para criação do estudo de casos, apresentado na seção a seguir, foram definidas algumas tecnologias e linguagens *web* utilizadas para criação de aplicativos híbridos. Os

critérios para definição das tecnologias e linguagens utilizadas foram: popularidade das tecnologias, facilidade na aprendizagem e experiência anterior nas tecnologias abordadas.

5. Estudo de Caso – NOTE PLUS

5.1. Requisitos do sistema

O aplicativo Note Plus foi desenvolvido com o intuito de ser multiplataforma, podendo assim oferecer uma melhor experiência ao usuário. As anotações são organizadas em três categorias: Empresarial, Pessoal e Acadêmico. As anotações são vinculadas a uma conta de usuário que deve ser criada no primeiro acesso. Esta conta de usuário, combinada ao recurso de criptografia ponta-a-ponta, garante a segurança das informações.

Para atender a todas as funcionalidades propostas foi necessário definir os requisitos de funcionalidade do sistema. São eles:

- **Cadastro:** Permitir que os usuários se cadastrem; Verificar se o login já existe antes de permitir o cadastro.
- ***Login*:** Validar credenciais de usuários e criar um *token* para cada usuário logado com o objetivo de identificá-lo; Permitir alteração de senha; Destruir os dados de sessão do usuário ao realizar o *logout* do sistema.
- **Anotações:** Permitir que o usuário visualize, edite e exclua suas anotações após fazer *login*.

A junção destes requisitos de funcionalidade resultou em um diagrama de caso de uso que pode ser visto na Figura 1.

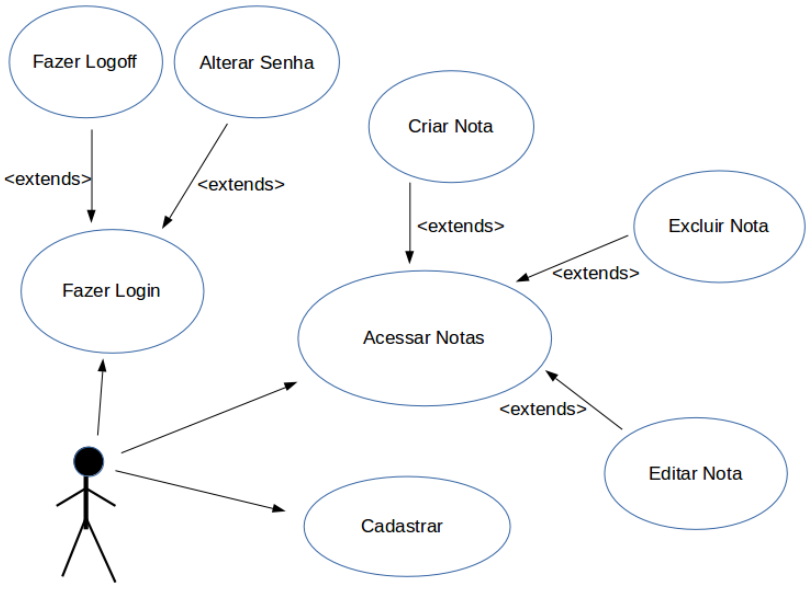

Figura 1: Requisitos funcionais do Note Plus

5.2. Desenvolvimento do aplicativo

Para desenvolvimento do projeto, foi criada uma arquitetura orientada a serviços (OASIS, 2006). Assim, o projeto foi dividido em três partes formando uma arquitetura

distribuída, sendo elas: *WebService;* banco de dados; e as aplicações-clientes, que representam as versões do aplicativo para uma das principais plataformas. Esta arquitetura é apresentada na Figura 2.

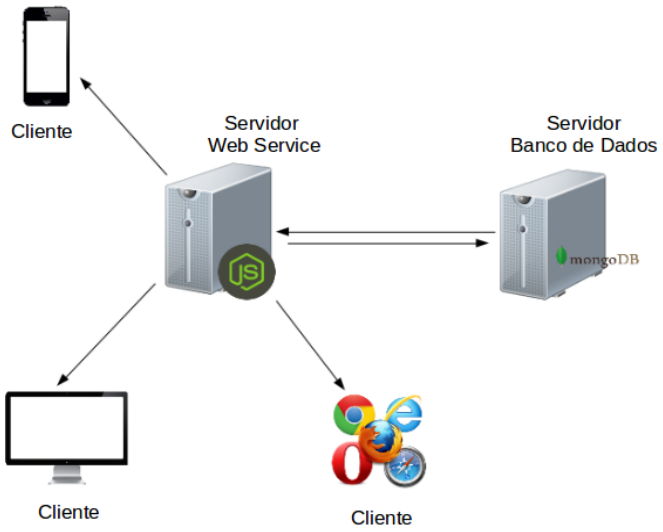

Figura 2. Arquitetura do sistema Note Plus.

Todas as versões das aplicações-clientes se comunicam com um *WebService* responsável por tratar as requisições dos usuários e se comunicar com o banco de dados. Este *WebService* foi desenvolvido em NodeJS (NodeJS, 2017), que utiliza a linguagem Javascript no lado do servidor. Desta forma foi possível aproveitar os conhecimentos em Javascript, reduzindo o tempo gasto no desenvolvimento de toda aplicação. Sua estrutura base foi criada a partir de um CLI *(Command Line Interface) c*hamado Express Generator *(*Express, 2017*)* que cria uma aplicação NodeJS em conjunto com o framework Express *(*Express, 2017*)*. Toda lógica e regra de negócios do aplicativo foi implementada como serviços do *WebService*.

Como base de dados para o sistema Note Plus, foi utilizado o MongoDB (MongoDB, 2017). O MongoDB é um banco de dados *NoSQL* orientado a documentos. Este banco de dados armazena documentos no formato *JSON*, o que facilita muito a comunicação e manipulação de dados por meio do NodeJS. O servidor de banco de dados não foi configurado e implementado do zero. Ao invés disso, foi utilizado o serviço de armazenamento da MLab (MLab, 2016), uma empresa focada em fornecer serviços de armazenamento do banco de dados MongoDB.

Para o desenvolvimento das telas da aplicação foram utilizados os padrões para o desenvolvimento *web* (W3C, 2017): HTML, CSS e Javascript. O desenvolvimento *web* permite o uso de frameworks e até mesmo bibliotecas de componentes *web*. Assim, para criação das interfaces de usuário, e utilizando critérios de familiaridade com as tecnologias e facilidade de aprendizagem, os *frameworks* AngularJS (AngularJS, 2017) e a biblioteca de componentes Bootstrap (Bootstrap, 2017) foram selecionados para criação das interfaces com o cliente.

O aplicativo-cliente foi desenvolvido como uma aplicação AngularJS (AngularJS, 2016), criado como um projeto separado e destinado a armazenar todo o código-fonte gerado ao decorrer do desenvolvimento da aplicação-cliente. Sempre que o código-fonte do aplicativo sofre alterações, e consequentemente é necessário atualizar todas versões do aplicativo, basta realizar uma cópia de todo código-fonte para cada

projeto específico dos aplicativos-clientes. Deste modo é possível centralizar todo o desenvolvimento em um único projeto, além de garantir total similaridade das funcionalidades.

Graças às características do AngularJS, como configuração de rotas para exibição das páginas *web* (*views*), essa exibição se torna dinâmica, de modo que a cada vez que o usuário navega por uma página diferente já são carregados os arquivos HTML e os controles (*controllers)* referentes a esta tela.

Para desenvolvimento do aplicativo Note Plus, foram criados dois *Services* do AngularJS, sendo que o primeiro é responsável por conter todas as regras de negócio e lógica do aplicativo, como: verificação de usuário logado; comunicação com o *WebService* para envio e recebimento de dados; e persistência de dados de login e *token* de autenticação enquanto o usuário estiver logado. Além de conter funções que são compartilhadas entre os diversos controles da aplicação, permitindo assim o reuso destas funções pelos vários controles da aplicação. O segundo *Service* funciona como uma camada de persistência, sendo capaz de persistir informações no armazenamento interno de dispositivos móveis. Assim o usuário pode ter acesso às suas anotações mesmo quando estiver *offline*.

5.3. Aplicativo WEB

Como todas as tecnologias utilizadas no desenvolvimento do aplicativo são tecnologias web, para disponibilização da versão *web* do aplicativo não foi necessário nenhuma adaptação no código-fonte.

Todo projeto foi copiado para a pasta *public* do *WebService*. A única exceção foi o arquivo *index.html* que foi copiado para a pasta *view* do projeto gerado pelo Express Generator. Por se tratar do arquivo HTML principal, o index.html deve ser carregado primeiro pelo navegador e por meio dele é que os outros arquivos são carregados. Deste modo, a versão *web* foi publicada junto ao *WebService*, utilizando o mesmo servidor, com o objetivo de economizar tempo de desenvolvimento e custos com infra-estrutura.

5.4. Aplicativo Desktop

Para versão *desktop* do aplicativo foi utilizado o *framework* Electron (Electron, 2017). O Electron utiliza das tecnologias *web*, como o HTML, CSS e Javascript, para criação de sistemas *desktop*. Sendo assim, a versão *desktop* do aplicativo Note Plus pôde reaproveitar todo o código-fonte desenvolvido para a versão *web* do sistema.

A única alteração no código-fonte foi realizada no arquivo *main.js,* responsável por criar a janela da aplicação. A alteração foi feita pois a classe *main.js* é responsável por iniciar a aplicação e definir as configurações importantes da aplicação, como: tamanho padrão da janela, nome da aplicação e o arquivo HTML que deve ser renderizado quando a aplicação for executada. O método *createWindow* deste arquivo sofreu algumas alterações para que tudo funcionasse corretamente, já que este método é o responsável por criar a janela principal da aplicação.

A pasta *app* contém todo o código-fonte, enquanto que a pasta *assets* contém o ícone que será exibido enquanto a aplicação estiver sendo executada. O processo de *build* é feito com o auxílio do plugin *electron-prebuilt* que utiliza uma versão pré-compilada do Electron específica para a plataforma-alvo. Isso é necessário para que o usuário não precise ter o Electron instalado em seu computador para executar o aplicativo. Ao compilar o projeto, a versão final do aplicativo é gerada na pasta *dist* ficando então isolada do restante do projeto.

Usando esse processo, o código-fonte *web* da aplicação foi compilado para os principais sistemas operacionais: Windows, Linux e Mac OS X.

5.5. Aplicativo Móvel

Para o desenvolvimento do aplicativo Note Plus em sua versão para dispositivos móveis, foi utilizado o framework Apache Cordova. O Apache Cordova é uma plataforma de desenvolvimento móvel com p*lugins* que permitem que o desenvolvedor acesse funções nativas do dispositivo (Apache Cordova, 2016).

Para criação dos aplicativos para dispositivos móveis, alterações no código-fonte foram necessárias. O principal objetivo das alterações foi a adaptação do aplicativo para ter um comportamento semelhante a um aplicativo nativo do *smartphone* e melhorar a experiência do usuário ao utilizar o Note Plus.

A primeira adaptação realizada foi o encapsulamento do método *onDeviceReady*, padrão do *framework Cordova*, dentro do método *run* do *AngularJS*. Essa adequação faz com que os dois *frameworks* inicializem juntos de modo que possam se comunicar entre si.

Para garantir uma navegação semelhante a dos aplicativos construídos com linguagem nativa, como por exemplo ao usar o botão "voltar", foi criado um *Service* e um *Listener* no projeto do AngularJS para gerenciar este comportamento. O *Listener* foi criado dentro da função onDeviceReady e executa sempre que o botão "Voltar" é acionado. Ao ser executado, o *Listener* onBackButton executa a função de tratamento do Service, que verifica qual é a rota em que o usuário se encontra para então direcioná-lo à próxima rota. A última alteração foi no arquivo *config.xml* que fica na raíz do projeto, neste arquivo são definidas configurações importantes como o nome e o ícone do aplicativo e também informações sobre o desenvolvedor. Alguns plugins também utilizam este arquivo para definir suas configurações.

6. Conclusão

O presente trabalho visou apresentar inicialmente as vantagens e dificuldades do desenvolvimento de um aplicativo compatível com as diversas plataformas disponíveis no mercado. Uma abordagem híbrida de desenvolvimento de aplicativos multiplataforma foi apresentada usando frameworks específicos e técnicas de desenvolvimento. Um estudo de casos foi desenvolvido com o objetivo de exemplificar a abordagem proposta inicialmente. Foi desenvolvido um aplicativo e foi gerado um aplicativo para cada uma das principais plataformas do mercado: *Web*, *Desktop* e Móvel. A estratégia de desenvolvimento de aplicativos híbridos se mostrou produtiva já que permite a criação de *apps* para diversas plataformas reutilizando grande parte do código-fonte gerado inicialmente.

Esta estratégia de desenvolvimento se mostrou ideal para empresas que têm recursos limitados ou almejam desenvolver aplicativos para várias plataformas de forma rápida. Entretanto, é necessário avaliar os requisitos de funcionalidade do aplicativo a ser desenvolvido. Como os aplicativos híbridos são executados no *Webview* das plataformas, sua capacidade de processamento e acesso aos recursos nativos do dispositivo como câmera e GPS é bem inferior à dos aplicativos construídos com linguagem nativa. Desta forma a estratégia a ser adotada fica condicionada aos requisitos de funcionalidade do projeto e aos recursos disponíveis para o desenvolvimento.

Como trabalhos futuros, será avaliado algumas características dos aplicativos híbridos como performance, uso de CPU e uso de memória. Uma comparação entre aplicativos híbridos e aplicativos nativos pode ser realizada com o objetivo de avaliar a qualidade dos aplicativos híbridos gerados.

Referências

Apache Cordova (2016). "Documentação Oficial do Apache Cordova", https://cordova.apache.org/docs/en/latest/, Março.

AngularJS (2016). "Documentação Oficial Angular JS", https://docs.angularjs.org/guide, Março.

Bernardes, T. F., Miyake, M. Y. (2016). "Cross-platform Mobile Development Approaches: A Systematic Review", In: IEEE Latin America Transactions, vol. 4, no. 4, pp. 1892-1898.

Bootstap (2017). "Documentação Oficial do Bootstrap", http://getbootstrap.com/, Março.

Dalmasso, I., Datta, S. K., Bonnet, C., Nikaein, N. (2013). "Survey, comparison and evaluation of cross platform mobile application development tools", In: 9th International Wireless Communications and Mobile Computing Conference (IWCMC '13), Sardinia, pp. 323-328.

Electron (2017). "Documentação Oficial do Electron", https://electron.atom.io/docs/, Março.

Express (2017). "Documentação Oficial do Express", http://expressjs.com/pt-br/, Março.

IBGE (2016). "Acesso à internet e à televisão e posse de telefone móvel celular para uso pessoal : 2015 / IBGE", Coordenação de Trabalho e Rendimento. – Rio de Janeiro : IBGE.

IDC (2016). "Smatphone OS Market Share, 2016 Q3". http://www.idc.com/promo/smartphone-market-share/os, Março.

Luquetti, L., Pires, D. F. e Neto Carvalho, S. (2015), "Desenvolvimento de aplicações para Dispositivos Móveis: Tipos e Exemplos de Aplicações na plataforma IOS", In: II Workshop de Iniciação Científica em Sistemas da Informação, Goiânia – GO, pp. 25-29.

Martins, C.; Antônio, A; Oliveira, C. A. (2013). "Os desafios para mobilização de aplicações baseadas em plataformas Web", In: Encontro Anual de Computação (Enacomp), P.294-300.

Milani, A. (2012). "Programando para iPhone e iPad: aprenda a construir aplicativos para IOS", Novatec, 1 ed.

MLab (2016), "Documentação Oficial do Mlab", https://mlab.com/, Março.

MongoDB (2017). "Documentação Oficial do MongoDB", https://docs.mongodb.com/, Março.

NodeJS (2017). "Documentação Oficial do NodeJS",https://nodejs.org/en/docs/, Março.

OASIS (2006). "Modelo de Referência para Arquitetura Orientada a Serviços 1.0", http://www.pcs.usp.br/~pcs5002/oasis/soa-rm-csbr.pdf. Março.

Palmieri, M., Singh, I. e Cicchetti, A. (2012). "Comparison of cross-platform mobile development tools", In: 16th International Conference on Intelligence in Next Generation Networks, Berlin, Germany, pp. 179-186.

W3C (2017). "Padrões de tecnologias web", https://www.w3.org/standards/webdesign/, Março.

Xanthopoulos, S. e Xinogalos, S. (2013). "A comparative analysis of cross-platform development approaches for mobile applications", In: Proceedings of the 6th Balkan Conference in Informatics (BCI '13). ACM, New York, NY, USA, pp. 213-220