Desenvolvimento de uma interface para dispositivos móveis associada a uma plataforma múltipla de dados

Eduardo Souza[1], Raphael Melo[1], Jomar Monsores[1], Carlos O. S. Mendes[1], João R. de T. Quadros[1]

[1] Escola de Informática e Computação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ) – Rio de Janeiro – RJ – Brasil

{edux404, raphael.mo, jomarfm06,schocair,jquadros80}@gmail.com

Abstract. *This paper approaches the development of an interface model for Internet or desktop applications, which presents difficulties in accessing data on mobile devices. Since there are still systems that work in environments without considering mobile devices, access to the data from these systems can lead to difficulties in use and even the abandonment of access in these devices. Considering that, the importance of these devices in data access has become increasing, it is necessary to adapt these devices to these environments. The interface layer of this paper is based on the Model-View-Controller architecture, adapted for any type of mobile device.*

Resumo. *Esse trabalho aborda o desenvolvimento de um modelo de interface para aplicativos voltados para a internet ou desktop, que apresentam dificuldades de acesso aos dados em dispositivos móveis. Como ainda existem sistemas que trabalham em ambientes sem considerar dispositivos móveis, o acesso aos dados desses sistemas pode proporcionar dificuldades de uso e até a desistência de acesso nesses dispositivos. Tendo em vista que, a importância desses dispositivos no acesso a dados tem se tornado crescente, isso faz com que haja necessidade de adaptações desses sistemas para esses ambientes. A camada de interface desse trabalho é baseada na arquitetura Model-View-Controller, adaptada para quaisquer tipos de dispositivos móveis.*

1. Introdução

O uso de dispositivos móveis para acesso a sistemas de informação (tablets, celulares e afins) tem crescido muito nos tempos atuais [Marques, 2018] [Mao et al., 2018]. Cada vez mais os desenvolvedores de sistemas se preocupam com a forma de acesso e manipulação de dados nesses dispositivos, tendo em vista justamente o crescimento do uso dessas plataformas no acesso a sistemas [Subasi et al., 2018] [Johar et al., 2018]. Nem todo sistema desenvolvido para a internet tem sua interface concebida para visualização e manipulação em plataformas com dispositivos móveis, e nem sempre a adaptação dessas telas a esses dispositivos oferece uma interface amigável para os usuários [Talukde et al., 2018].

Como dispositivos móveis costumam ter telas de visualização em tamanhos menores que monitores de 15”, usar e interagir com a plataforma apresenta maior

dificuldade, caso a interface desenvolvida não levar em conta a adaptação para esse tipo de ambiente de visualização [Talukde et al., 2018].

Alguns sites são desenvolvidos em formato renderizado e por isso se ajustam mais para serem apresentados em telas maiores, forçando o usuário a utilizar recursos não amigáveis, tais como, ter de rolar horizontalmente a página ou acessar o zoom do navegador, para gerenciar os seus dados [Schütz, 2013]. O esperado é que as interações em dispositivos móveis sejam fluidas e simples o suficiente para que o usuário se sinta confortável com a aplicação, sem correr o risco de abandonar o uso do aplicativo nessas plataformas [Johar et al., 2018].

A ideia é construir interfaces que sejam adaptadas ao ambiente de dispositivos móveis, reduzindo-se as ações de rolamento horizontal ou de zoom, permitindo que os dados necessários aos usuários estejam disponíveis de forma mais amigável [Johar et al., 2018]. Um modo de desenvolver aplicativos adaptáveis para visualização nesses dispositivos é por meio de modelos que trabalhem com a extração dados do site de origem. Pode-se usar, para isso, a técnica conhecida como *web scraping* [Mitchel, 2015], que vai rearranjar esses dados em um modelo de interface construído para o dispositivo destino.

Para essa adaptação ocorrer sem problemas, de modo que se possa ter essa interface amigável, tem-se uma fase de exame da estrutura do site origem, de modo a identificar a estrutura sendo HTML ou XML e suas *tags*, para que a extração de dados ocorra sem perda de informação [Mitchel, 2015]. Após esse exame, arranja-se a interface para o modelo de dados obtido e então as informações são apresentadas. A fase de extração de dados vai produzir um modelo de uma interface adaptável ao ambiente do dispositivo destino (no caso, dispositivos móveis).

A interface desse trabalho foi baseada na arquitetura *Model-View-Controller* (MVC) [Vauple et al., 2018], na qual há uma camada que gerencia os dados, obtendo, organizando e interpretando os mesmos, uma no qual vai ocorrer a delegação das interações do usuário com a distribuição dos dados e a final que organiza os dados com elementos do ambiente dos dispositivos móveis, para definir posicionamentos, dimensões e apresentação das informações, com suas devidas restrições.

Para verificar a aplicabilidade da interface proposta, denominada de FRW-MVC, esse trabalho foi focado na modelagem, extração e construção de um protótipo para um Portal Acadêmico de uma instituição pública, para ser visualizado em dispositivos móveis baseados em *Android* e *iOS* [Vauple et al., 2018].

A escolha para utilizar o Portal Acadêmico da instituição para construir um protótipo foi obtida a partir de uma pesquisa de intenção, com cerca de 329 alunos da Instituição, de vários níveis acadêmicos, no qual se perguntava qual a maior necessidade, em termos de aplicativos de dispositivos móveis, observados entre os diversos sistemas oferecidos pela instituição via internet.

Esse trabalho está dividido em uma introdução, a explicação sobre o modelo escolhido e sua concepção, uma explicação sobre o protótipo do modelo sobre o Portal Acadêmico, uma análise dos resultados obtidos e, por fim, a conclusão do trabalho.

2. A FRW-MVC para Dispositivos Móveis

2.1 Padrão MVC

O padrão MVC foi proposto por Reenskaug (1979), sendo que sua concepção era de se construir uma ligação entre o modelo sistêmico geral de um usuário e o modelo adequado para um ambiente computacional [Aniche et al., 2018]. Um modelo ou interface baseado no MVC tem as suas classes e objetos separados em três camadas de responsabilidades [Aniche et al., 2018]:

a) *Model*: responsável pela gerência do comportamento dos dados, permitindo sua extração, organização e interpretação, podendo ser trabalhado como um componente individual ou global;

b) *View*: que vai gerenciar a formatação de saída da interface do sistema, possibilitando a visibilidade dos dados para o usuário; e

c) *Controller*: que é a que interpreta as entradas fornecidas pelo usuário, comandando a *View* e o *Model* para se alterarem de acordo com todas as requisições feitas ao sistema pelo usuário. O *Controller* associa os dados do *Model* com a *View* correta, de acordo com a ação executada ou com os resultados obtidos do sistema.

Na Figura 1 pode-se ver um exemplo de um modelo padrão MVC.

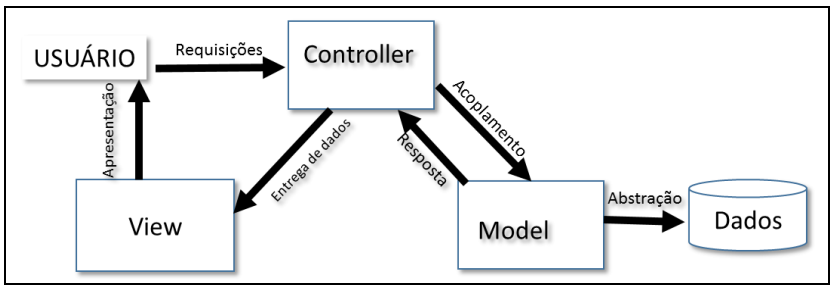

Figura 1. Exemplo de um modelo de padrão MVC para uma aplicação genérica.

2.2 MVC para o desenvolvimento proposto

A interface MVC desenvolvido nesse trabalho, a FRW-MVC, utiliza a comunicação entre as camadas através do padrão denominado *delegate* [Endrei et al., 2004]. Esse padrão é utilizado no projeto para reduzir o acoplamento entre as camadas cliente e a camada de negócio do site, que é aquele no qual se deseja construir a interface para dispositivos móveis. O uso de *delegate* permite esconder detalhes de implementação.

O *delegate* pode ser programado para agir como uma abstração, no lado cliente, para os métodos de negócio do site. Cada *delegate* serve para esconder, por exemplo, a complexidade da busca por recursos para um tipo de aplicação distribuída, ou com *data sources*, com o fim de obter conexões a fontes de dados desses ambientes. Sua utilização torna transparente para os usuários os detalhes de serviços do site origem.

A implementação dos *delegates* para esse trabalho é feita para eles funcionarem como um protocolo de comunicação, pois a eles são delegadas as transferências de informações e estabelecidas comunicação entre duas classes ou objetos quaisquer.

Eles definem “contratos”, de modo que qualquer classe que aderir a um desses *delegates* vai implementar os métodos dessa classe para satisfazer as necessidades de uma outra classe. Isso é visto como uma vantagem de se implementar a comunicação via *delegates*, pois assim, qualquer classe pode ser delegada a outra, bastando programar o protocolo de comunicação adequado.

A interação entre a camada *Model* da FRW-MVC e o site de origem dos dados é feita através da classe *NetworkModel*. Essa classe é utilizada como base para montar todos os perfis de modelos, originados da camada *Model*. Nela estão declaradas propriedades e métodos comuns entre os diversos *Models*, como, por exemplo, informações dos *headers* para as requisições via HTTP.

A partir dessa configuração, o *NetworkModel* é utilizado para criar novos modelos (*Models*) que equivalem à versões do modelo base. Cada uma dessas versões vai avaliar respostas dos pedidos e requisições feitas ao site e vão atribuir um estado representativo da situação do modelo em si. Por exemplo, para um modelo tipo *Login* (Modelo *Login*) de um site A, se o título do HTML recebido for “A - Login”, o estado atribuído será *“.ready*”, ou seja, pronto para criar o Modelo *Login* no dispositivo móvel, se o título recebido for “A - Página Inicial”, o estado será *“.success*”, indicando que o *login* feito no Modelo *Logi*n foi bem sucedido. Todos esses estados são definidos em uma lista de situações possíveis, associadas ao método *NetworkModelStat*e.

Existem vários outros tipos de modelo criados, tais como, Modelo Disciplinas, Modelo Perfil, Modelo Notas entre outros, que representariam, no caso do protótipo, as diversas telas do site original. No caso da formação de modelos complexos, que envolvem análise de páginas HTML do site A, o estado é alterado para *“.ready”* quando o HTML é recebido, e para *“.success”*quando a análise é concluída e os dados ficam prontos para serem acessados no formato adequado ao dispositivo móvel. Para modelos mais simples, que só são usados para armazenar dados de maneira estruturada, o tratamento é declará-los como classes que não vão herdar as características de outras classes base.

Mesmo que o site não ofereça uma apresentação dos dados visível, é possível utilizar métodos especiais (de acordo com a linguagem escolhida) para executar as tarefas, a fim de extrair os dados diretamente do HTML. Quando o um modelo do FRW-MVC recebe uma resposta do site, ele a avalia, se o título do HTML for apropriado, ele avisará um *delegate* (ou irá postar uma notificação, no caso de modelos compartilhados) de uma mudança de estado, então o *Controller* vai solicitar ao modelo para analisar o HTML e extrair os dados necessários. Um outro método analisará o HTML para gerar um documento legível para extração dos dados usando seletores *Cascading Style Sheets* (CSS).

Quando a análise é finalizada, o *delegate* (ligado à camada *Controller* da FRW-MVC) é notificado e os dados são passados para serem formatados e serem exibidos no dispositivo móvel, sendo que o trabalho final dessa formatação é realizado pela camada *View* dessa interface. Convém destacar que cada dado passado para formatar a saída tem como informação anexada seus aspectos de restrições, ou seja, os dados só podem ser alterados pelo usuário se as restrições do dado no site original assim o permitirem.

No caso de existirem *captchas*, para tratar de segurança, poder-se-á utilizar uma *webview* própria, que seria apresentada só para o usuário validar esses *captchas*. Isso é facilitado, porque as requisições do usuário são tratadas pela classe *FrontController*, que interage com os *delegates* para montar a saída de acordo com os comandos fornecidos pelo usuário.

Na Figura 2 é visto a representação da FRW-MVC proposta.

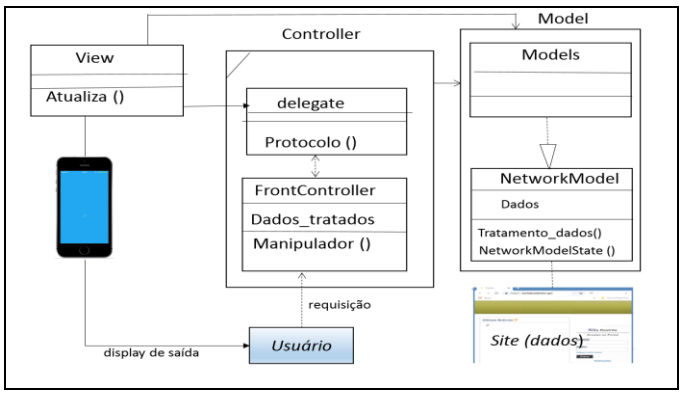

Figura 2. FRW- MVC voltada para aplicativos para transformar telas de dados de sites comuns em telas manipuláveis e amigáveis para dispositivos móveis.

3. Protótipo do Portal Acadêmico

3.1 Justificativa

O Portal Acadêmico da Instituição foi escolhido como protótipo do FRW-MVC devido a uma pesquisa realizada entre 329 alunos de diversos níveis de ensino, para verificar qual sistema do site da instituição seria o mais requerido para ser visto, de forma amigável, em dispositivos móveis.

Dos 329 alunos que responderam o questionário, que tinha apenas a pergunta de qual sistema do site da instituição deveria ser "transformados" para visualizações amigáveis para celulares e afins, 247 alunos (75%) colocaram como primeira opção o Portal Acadêmico da instituição, e 32 alunos colocaram esse portal como segunda opção (9,7%). Desse modo, foi percebido, que grande parte da comunidade discente manifestou o desejo de ter um acesso mais amigável, em seus dispositivos móveis, das informações dispostas no Portal Acadêmico da instituição.

Nas Figuras 3.a e 3.b podem ser vistas como são as telas de acesso ao Portal *in natura*. A Figura 3.a mostra a tela de *Logi*n vista por um celular e a Figura 3.b apresenta outra tela, vista pelo mesmo celular, contendo os dados do aluno. Ambas foram obtidas do Portal Acadêmico da instituição, sem que tivesse sido utilizado a FRW-MVC desenvolvido nesse trabalho.

Como pode ser observada, em ambas as telas há dificuldade de apresentação da informação e sua manipulação nesse tipo de dispositivo. Isso ocorre pelo fato do sistema não ter uma interface adaptável para dispositivos móveis, gerando uma tela com caracteres pequenos, que exigem zoom para melhor ser gerenciada.

Figura 3.a. Tela de *Login* do Portal Acadêmico visto através de um celular.

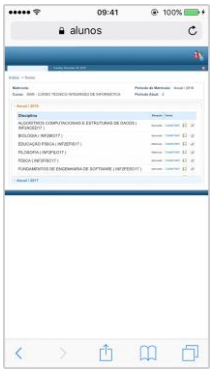

Figura 3.b. Tela de dados do aluno, via Portal Acadêmico, visto por um celular.

3.2 Construção do protótipo

O protótipo desse trabalho foi construído com o uso da linguagem *Swift* [Galvão, Castro Junior & Moreira, 2017]. Apesar do *Swift* ter sido criado, originalmente, para desenvolvimento em plataformas com *Apple-iOS*, é possível utilizá-lo para desenvolver aplicativos também para ambientes com *Android* [Zhang et al., 2012], isso facilitou a escolha dessa linguagem como a mais adequada para gerar interfaces visuais para as mais populares plataformas de dispositivos móveis.

A programação das classes *delegates* do protótipo foi realizada para que houvesse a gerência os dados das páginas obtidas pela camada *Model*. Um exemplo de parte da programação de uma das classes *delegate* pode ser visto na Figura 4.

```
protocol SubjectDelegate {
    func ready(subject: Subject)
    func success(grades: [String])
}
```

Figura 4: Exemplo de código Swift para classe *delegate* que trata das condições de estado dos *Models* da camada *Model*.

O Portal Acadêmico não foi criado com APIs próprias para que apresentar os dados em *JavaScript Object Notation* (JSON) [2018], que é um formato padrão aberto para troca de dados. Contudo, com o uso de uma biblioteca do *Swift* que trata de dados de páginas HTML (*SwiftSoup*), é possível extrair os dados diretamente do site do portal.

Quando o *Model* da FRW-MVC recebe uma resposta do servidor do site, ele avalia essa resposta. Com o título do HTML, ele avisará a uma classe *delegate* designada, ou informará com uma notificação de erro. Caso ocorra uma alteração de estado da página (uma nova página é carregada, por exemplo), o controlador requisitará ao *Model* para analisar essa mudança, identificando o novo HTML do site, e fará a extração dos novos dados necessários.

A FRW-MVC analisa os dados do HTML via seletores CSS e gera um documento legível de modo que os dados possam ser preparados para saída. Há métodos que são específicos para extrair o valor de um *input* ou de um texto de um componente da página do site. Quando essa análise e extração é finalizada, a classe

delegate designada vai fazer uma nova notificação e distribuir os dados para formatação final de saída.

A FRW-MVC também permite que se exiba os relatórios disponíveis em um site. Para esse caso, ao invés do pedido de um relatório retornar um HTML, o pedido retorna um arquivo, que é transformado em formato PDF. Esse arquivo PDF é salvo em um diretório temporário dentro do aplicativo, e quando o download termina, a camada *Model* da FRW-MVC notifica o *delegate* apropriado, passando o caminho do diretório no qual o arquivo está localizado, mas sem carregar os dados do relatório em si. Esse arquivo PDF é então exibido por uma *webview* própria, que apresenta o arquivo diretamente do diretório.

Uma opção de compartilhamento é exibida, na qual o usuário pode salvar o documento em serviços de armazenamento ou o compartilhar em aplicativos de comunicação, como por exemplo por *emails* ou *Whatsapp*.

Destaca-se que o aplicativo desenvolvido através da FRW-MVC não prescinde de uso de qualquer navegador específico, pois o trabalho do aplicativo é realizar a conexão via internet e atuar como uma espécie de navegador dedicado a função de acesso e gerência de dados do portal.

O protótipo desenvolvido não permite que se modifique as notas e informações de disciplinas, tal como ocorre no site original, mas permite alterações dos dados de perfil, de acordo com as regras de restrições do site original.

4. Resultados obtidos

O aplicativo baseada no FRW-MVC foi testado por 39 alunos da instituição, sendo vinte alunos do ensino médio-técnico, dez alunos da graduação e nove da pós-graduação. Após ter sido instalado nos celulares desses alunos, foram dados 29 dias para que eles testassem a interface como um todo.

Após os 29 dias de uso, no 30º dia foi iniciado uma pesquisa, que captou as percepções desses alunos em relação à interface (as respostas estão apresentadas no subitem de discussão de resultados). Foram dados dois dias para que eles respondessem a um questionário que continha perguntas simples (cada uma com três respostas: “sim”, “não”, “não sei”), para verificar os seguintes aspectos:

- Visibilidade: “A interface apresentou melhor visibilidade dos dados em relação ao site original?”

- Amabilidade: “ A interface se mostrou mais amigável do que o uso direto do sistema através do site?”

-Funcionalidade: “ A interface apresentou acesso as funções do Portal, de modo que pudesse atender as expectativas do usuário?”

- Facilidade: “ A interface mostrou aspecto de fácil uso em comparação com o acesso direto ao site?”

- Navegabilidade: “ O acesso as funcionalidades da interface se mostraram navegáveis e de fácil compreensão de uso?”

- Conectividade: “Durante os dias de uso, a conexão ao Portal, via aplicativo, se mostrou estável?”

4.1. Utilização do aplicativo

Nas Figuras 5.a e 5.b pode-se acompanhar o procedimento de uso do aplicativo desenvolvido através da FRW-MVC. A Figura 5.a apresenta a sequências de telas principais acessadas ao se utilizar o aplicativo. A primeira é a tela de *Login*, equivalente ao Modelo *Login*, na qual, uma vez autenticado, o aluno acessa a tela do Modelo Disciplinas as quais o aluno está ligado no semestre (se o curso for semestral) ou ao ano (se o curso for anual). Ao final é apresentado a tela do Modelo Notas, que retorna os dados associados à disciplina escolhida.

O aluno também pode ter acesso ao seu perfil (Modelo Perfil), visto na Figura 5.b. Devido ao grande volume de informações presentes nesta página, a FRW-MVC organiza os dados em uma estrutura dicionário, no qual cada título de informação é uma chave, associada a um valor. Por exemplo, se “Estado” é uma chave, “Rio De Janeiro” é um valor. Com isso, o *Controller* pode acessar os dados de forma mais rápida, que é essencial para manter a interface fluida e amigável, sem ser necessário montar uma estrutura diferente para cada dado apresentado na página do Modelo Perfil.

Figura 5.a. Sequência de telas apresentadas ao longo do uso do aplicativo do Portal Acadêmico, baseado na FRW-MVC, na qual a primeira tela representa o *Login*, a seguir as disciplinas associadas ao aluno e, por fim, a apresentação das notas da disciplina escolhida.

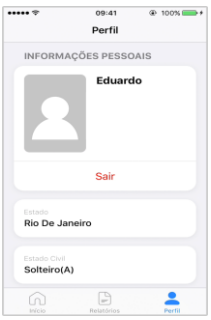

Figura 5.b. O aluno também pode se direcionar, por navegação das funcionalidades, ao Modelo Perfil, com a tela de perfil do aluno, no qual ele pode acessar e alterar os campos permitidos.

4.2 Discussão sobre os resultados

Os resultados da pesquisa sobre o uso da ferramenta podem ser observados na Tabela 1. Todos os 39 alunos responderam às perguntas sobre os aspectos do aplicativo.

Tabela 1. Resultados da pesquisa da utilização do aplicativo com a FRW-MVC por alunos da instituição.

Aspecto Examinado	*Σ de "Sim"*	*Σ de "Não"*	*Σ de "Não sei"*	*Índice de aprovação*
Visibilidade	33	2	4	85%
Amabilidade	34	2	3	87%
Funcionalidade	37	2	0	95%
Facilidade	36	2	1	92%
Navegabilidade	32	2	5	82%
Conectividade	39	0	0	100%

A interface teve uma aprovação média acima de 80%. Desconsideram-se os que não souberam avaliar os aspectos examinados (os que responderam "Não sei"), por não saberem o significado do aspecto, a aprovação chega a 95%.

De um modo geral o aspecto que mais se destacou foi a Conectividade, isso porque o aplicativo se conectou de forma plena e estável ao site. Depois dele, a Funcionalidade foi aspecto mais aprovado pelos usuários, significando que as expectativas em termos de funções que se desejava do aplicativo foram atendidas em sua quase totalidade.

O aspecto menos aprovado foi o da Navegabilidade, para essa questão foi perguntado, informalmente, o motivo da não aceitação plena, o que se informou foi que nas telas de perfil e de disciplinas ainda se precisaria rolar a página. Esse aspecto observado pelos que emitiram essas críticas é devido a uma falsa expectativa que na tela do celular todos os dados seriam apresentados em formato grande, o que é inviável para telas mais pequenas.

5. Conclusão

Conforme observado, a aceitabilidade do aplicativo foi alta e foi possivel demostrar que uma interface construída através da FRW-MVC apresenta melhoras significativas no acesso e apresentação dos dados em dispositivos móveis em relação a telas originais. A FRW-MVC proporcionou uma interface amivágel, com navegabilidade funcional e que representou de forma integral as funcionalidades oferecidas pelo site origem.

Houve uma aprovação significativa para o aplicativo, devido a isso tem-se planos para desenvolver outros aplicativos, usando o FRW-MVC, para outras aplicações oferecidas pela instituição, tais como, Portal de Noticias, Cadastramento para Editais, Portal do Professor, entre outros correspondam às necessidades da comunidade.

Além disso, tem-se a ideia de ampliar o projeto e usar a FRW-MVC para aplicativos fora do ambiente acadêmico, que prescindam de serem melhores apresentados em dispositivos móveis, com interface mais amigáveis e funcionais.

Agradecimentos: Os autores agradecem a FAPERJ, Capes, CNPq e ao CEFET-RJ pela ajuda ao trabalho.

6. Referências Bibliográficas

Aniche, M.; Bavota, G.;Treude.M; Gerosa, M.A. & Deursen, A. (2018). Code smells for Model-View-Controller architectures. In: Empirical Software Engineering, v23-4, pages 2121–2157.

Endrei, M.; Ang J.; Arsanjani A.; Chua, S.; Comte, P.; Krogdahl, P. & Luo M. (2004). NEWLING Tony. Patterns: Service-Oriented Architecture and Web. In: IBMRedbooks, 2004, New York:USA.

Galvão, E.B.B.; Castro Junior, A.B.C. & Moreira, E. P. (2017). Desenvolvimento de aplicativo para auxílio em levantamento topográfico em plataforma iOS na Linguagem Swift 3.1. In: Encontros Universitário da UFC, v2,n1, Brasil.

Johar, R. A. ; Fakieh, E. ; Allagani, R. & Qaisar, S. M. (2018). A smart home appliances control system based on digital electronics and GSM network. In: 15th Learning and Technology Conference (L&T). Arabia Saudita.

JSON (2018). Introducing JSON. Disponível em http://json.org . Acesso em julho 2018.

Mao, T.; Cao,C.; Peng, X. & Han, W. (2018). A Privacy Preserving Data Aggregation Scheme to Investigate Apps Installment in Massive Mobile Devices, In: Procedia Computer Science, v129, pages 331-340.

Marques, C. G. (2018). Mobile technologies for tourism and culture. In: Superavit – Revista de Gestão & Ideias, v3, pages 67-77, Portugal.

Mitchell, R. (2015). Web Scraping com Python: Coletando dados na web moderna. São Paulo, Brasil: Novatec.

Reenskaug, T. (1979). MODELS - VIEWS - CONTROLLERS. Technical note, Xerox PARC, Dezembro, USA.

Schütz, F. (2013). Web design. 22a edição. Ed. UTFPR, Curitiba-Brasil.

Subasi, A.; Radawn, M.;Kurdi, R. & Khatleb, K. (2018). IoT based mobile healthcare system for human activity recognition. In: 15th Learning and Technology Conference (L&T). Arabia Saudita.

Talukde S.; Witherspoon,S.;Srivastava,K. & Thompson, R. (2018). Mobile Technology in Healthcare Environment: Security Vulnerabilities and Countermeasures. In: arXiv:1807.11086v1, Cornwell University.

Vaupel,S.; Taentzer,G.; Gerlach, R. & Guckert,M. (2018). Model-driven development of mobile applications for Android and iOS supporting role-based app variabilit. In Software & Systems Modeling,v17-1, pages 35–63.

Zhang, Y.; Yang, M.;Zhou, B.; Yang, Z.; Zhang, W. & Zang, B. (2012). Swift: a register-based JIT compiler for embedded JVMs. In: Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments, pages 63-74. England-UK.