Desenvolvimento de interface de dispositivos móveis para utilização com uma plataforma múltipla de dados

Mobile device interface development for use with a multiple data platform

DOI:10.34117/bjdv7n2-542

Recebimento dos originais: 20/12/2020
Aceitação para publicação: 29/01/2021

Eduardo Souza
Escola de Informática e Computação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ) – Rio de Janeiro – RJ – Brasil
E-mail: edux404@gmail.com

Raphael Melo
Escola de Informática e Computação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ) – Rio de Janeiro – RJ – Brasil
E-mail: raphael.mo@gmail.com

Jomar Ferreira Monsores
Escola de Informática e Computação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ) – Rio de Janeiro – RJ – Brasil
E-mail: jomarfm06@gmail.com

Carlos Otávio Schocair Mendes
Escola de Informática e Computação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ) – Rio de Janeiro – RJ – Brasil
E-mail: schocair@gmail.com

João Roberto de Toledo Quadros
Escola de Informática e Computação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ) – Rio de Janeiro – RJ – Brasil
E-mail: jquadros80@gmail.com

RESUMO
Essa pesquisa aborda o desenvolvimento de um modelo de interface aplicado a sistemas desenvolvidos para a internet ou desktop, que apresentam dificuldades de acesso e apresentação aos dados em dispositivos móveis. Ainda existem sistemas que trabalham em ambientes sem considerar acesso por dispositivos móveis, deste modo, prejudicando o acesso aos dados desses sistemas, proporcionando dificuldades de uso e fazendo ocorrer até a desistência no uso desse aplicativos 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. O padrão de interface proposto é baseado na arquitetura Model-View-Controller, e foi concebida para se adaptar a quaisquer tipos de dispositivos móveis.

Palavras Chaves: MVC, Padrões de Projeto, Linguagem Swift.

ABSTRACT
This research addresses the development of an interface model applied to systems developed for the internet or desktop, which present difficulties in accessing and presenting data on mobile devices. There are still systems that work in environments without considering access by mobile devices, thus harming access to data from these systems, providing difficulties of use and making it occur until the withdrawal of the use of these applications on these devices. Considering that the importance of these devices in data access has become increasing, this makes it necessary to adapt these systems to these environments. The proposed interface standard is based on the Model-View-Controller architecture, and is designed to adapt to any type of mobile devices.

Keywords: MVC, design patterns, Swift Language.

1 INTRODUÇÃO

O uso de dispositivos móveis para acesso a sistemas de informação (tablets, celulares e afins) tem sido muito frequente na atualizdade [Marques, 2018] [Mao et al., 2018] [Pereira et al, 2020]. Existem cada vez mais esenvolvedores de sistemas que se preocupam com o modelo de apresentação, de acesso e manipulação de dados nesses dispositivos, pelo momento de alto crescimento do uso dessas plataformas no acesso a vários tipos de sistemas diferentes [Subasi et al, 2018] [Johar et al., 2018]. Ainda existem sistemas cujo desenvolvimento não levam em conta a apresentação dos mesmo em dispositivos móveis ou mesmo para visualização em navegadores da internet, tendo suas interfaces concebida sem a devida adaptação para esses dispositivos oferecendo uma interface não-amigável para os usuários [Talukde et al, 2018]. Os dispositivos móveis costumam ter telas de visualização em tamanhos bem menores que monitores de 15”, com cerca de 8” a 10” no máximo, dificultando o uso e interação com a plataforma, pois as interfaces desenvolvidas não levam 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 quando apresentados em telas maiores, o que força o usuário a utilizar recursos nada amigáveis para manipular a tela, tais como, rolamento horizontal da página, usar artifício de *zoom* no navegador, entre outros recursos que se tornam necessários para gerenciar os dados no sistema [Schütz, 2013][Matoski et al, 2020]. A ideia é que as interações do sistema em dispositivos móveis possam serem as mais fluidas e simples possíveis, o suficiente para que o usuário tenha conforto no uso da aplicação, sem que haja risco de desistência no uso do aplicativo nos dispositivos móveis [Johar et al, 2018].

A proposta é desenvolver modelos de interfaces que sejam ao máximo adaptadas para ambientes de dispositivos móveis [Matoski et al, 2020], reduzindo-se as ações de rolamento horizontal ou de *zoom*, permitindo que os dados dos usuários sejam disponibilizados de forma mais amigável possível[Johar et al, 2018]. Um método para desenvolver aplicativos de fácil adaptação para dispositivos móveis, é através da construção e uso de modelos que manipulem a extração dados do site de origem e os formate para visualização amigável. A técnica mais adequada para esse propósito é a *web scraping* [Mitchel, 2015], que visa rearranjar os dados em um modelo de interface amigável, construído de acordo com os padrões do dispositivo destino [Pereira et al, 2020].

Para essa adaptação ocorrer sem que haja problemas, para se obter essa interface amigável, propõem-se uma fase de análise da estrutura do site origem, de modo a identificar qual a estrutura de HTML ou XML, e suas tags, estão disponíveis para a extração de dados, de modo que ela ocorra sem perda de informação [Mitchel, 2015]. Após essa análise, a interface é rearranjada para que o modelo de dados obtido possa ser apresentado no novo dispositivo destino. Essa etapa de extração de dados produz um modelo de uma interface mais adaptável ao ambiente de dispositivos móveis.

A proposta de interface dessa pesquisa se baseou na arquitetura conhecida como *Model-View-Controller* (MVC) [Vauple et al., 2018], na qual existe uma camada que gerencia os dados, obtendo, organizando e interpretando os mesmos, outra camada no qual ocorre a delegação das interações do usuário com essa distribuição dos dados e a camada final, que organiza os dados com os elementos do ambiente dos dispositivos destinos, definindo os posicionamentos, as dimensões e a apresentação final das informações e suas restrições. A fim de verificar a aplicabilidade dessa proposta de interface, denominada de FRW-MVC, focou-se na modelagem, extração e construção de um protótipo para um Portal Acadêmico de uma instituição pública de ensino, de modo a ser visualizado em dispositivos móveis, ou celulares, baseados em sistemas operacionais do tipo Android ou iOS [Vauple et al, 2018].

O Portal Acadêmico da instituição foi escolhido a partir de de uma pesquisa de intenção, na qual cerca de 435 alunos da Instituição, de vários níveis acadêmicos, responderam a uma pergunta de qual seria a maior necessidade, em termos de aplicativos de dispositivos móveis para serem acessados, observados entre os diversos sistemas oferecidos pela instituição via internet.

Para demonstrar essa proposta de modelo de interface, esse trabalho se apresenta com uma introdução, a explanação sobre o modelo escolhido, com a sua concepção, uma apresentação sobre o protótipo do modelo para o Portal Acadêmico, uma análise dos resultados obtidos e, por fim, a conclusão do trabalho.

2 O PADRÃO FRW-MVC PARA DISPOSITIVOS MÓVEIS

2.1 SOBRE O MVC

O padrão MVC é um proposta de Reenskaug (1979), sendo sua concepção para 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 comportamental dos dados, permitindo a extração, organização e interpretação dos mesmos, podendo ser trabalhado como um componente individual ou global;

b) View: que gerencia a formatação de saída da interface do sistema, possibilitando uma melhor visibilidade dos dados para o usuário; e

c) Controller: que é a camada que interpreta as entradas fornecidas pelo usuário, comandando a View e o Model para haver as alterações de acordo com todas as requisições feitas pelo usuário. A camada Controller associa os dados do Model com as informações da camada View, de acordo com a ação executada ou com os resultados obtidos do sistema. Um exemplo de um modelo padrão MVC pode ser visto na Figura 1.

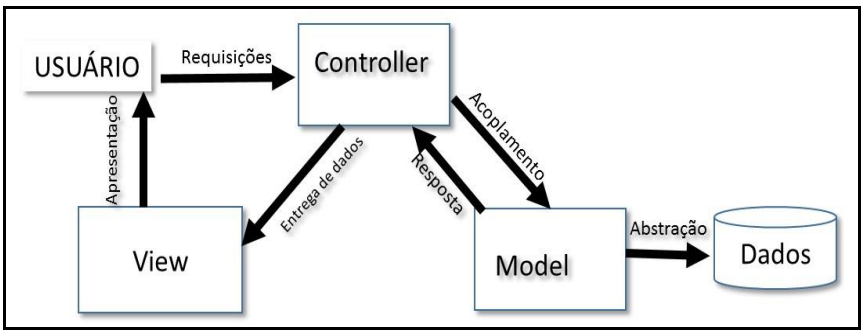

Figura 1. Exemplo de um modelo de padrão MVC para uma aplicação genérica, mostrando as interações entre suas camadas Model, View e Controler.

2.2 SOBRE O PADRÃO FRW-MVC PROPOSTO

O novo padrão de interface MVC, proposto e desenvolvido nesse trabalho, denominado de FRW-MVC, se utiliza d a comunicação entre as camadas através do padrão de projetos denominado *delegate* [Endrei et al, 2004]. Esse padrão é utilizado para reduzir o grau de acoplamento entre as camadas cliente e a camada de negócio do sistema inserido em um site da Web, sendo esse site aquele no qual se deseja construir a interface amigável para dispositivos móveis. O uso de *delegate* permite esconder de forma adequada os detalhes de implementação. O *delegate* também pode ser programado para agir como uma abstração, no lado cliente, para que os métodos de negócio do site sejam escondidos. Cada classe *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* nessa pesquisa foram realizadas para eles funcionassem tal qual um protocolo de comunicação, uma vez que a eles são delegadas as transferências de informações com estabelecimento das comunicações entre duas classes ou objetos quaisquer.

Os *delegates* também definem modelos de "contratos", de modo que qualquer classe que aderir a um deles vai tratar de implementar os métodos dessa classe, de modo a 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 *NetworkMode*l. Essa classe é utilizada como base para montar todos os perfis de modelos, cuja origem vem da camada Model. Nessa classe estão declaradas propriedades e métodos comuns entre os diversos outros Models, como, por exemplo, informações dos *headers* para as requisições via HTTP.

A partir dessa configuração, o *NetworkModel* vai ser utilizado para criar outros novos Models, que equivalem a versões do Model base. Cada uma dessas versões vai ser usada para 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 Model tipo 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 Model Login no dispositivo móvel, se o título recebido for "A - Página Inicial", o estado será ".success", indicando que o login feito no Model Login foi bem sucedido. Todos esses estados são definidos em uma lista de

estados possíveis, associadas ao método *NetworkModelState*. Existem vários outros tipos de Models criados, tais como, Model Disciplinas, Model Perfil, Model Notas entre outros, que representariam, no caso do protótipo, as diversas telas do site original.

No caso da formação de Models mais complexos, que vão envolver análise mais elaboradas 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 Models mais simples, que só são usados para armazenar dados de maneira estruturada, o tratamento é declará-los como classes que não vão poder 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 Model do FRW-MVC recebe uma resposta do site, ele a avalia. Se o título do HTML for apropriado, ele avisará uma classe delegate, ou irá postar uma notificação, no caso de Models dos tipos compartilháveis, indicando uma mudança de estado. Desse modo a camada *Controller* vai solicitar ao Model 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, a classe *delegate* (ligado à camada *Controller* da FRW-MVC) vai ser notificada e os dados são passados para serem formatados e serem exibidos no formato adequado ao 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.

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 apresentado a proposta da FRW-MVC.

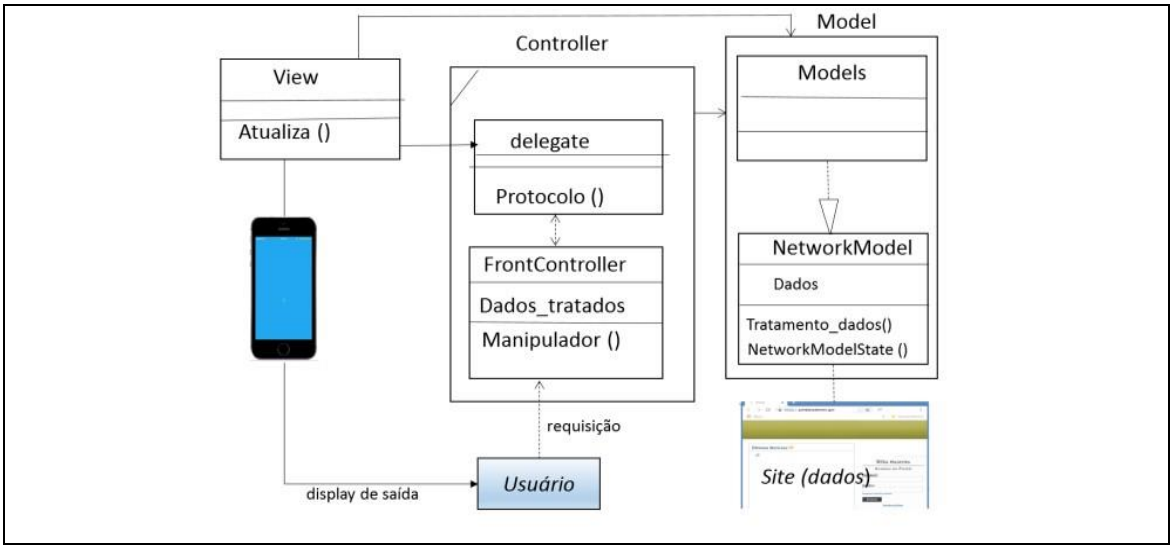

Figura 2. FRW- MVC, com suas camadas definidas, que é voltada para aplicativos, sendo usada para transformar telas de dados de sites comuns em telas manipuláveis e amigáveis para dispositivos móveis.

3 MODELAGEM PARA O PORTAL ACADÊMICO

3.1 PRINCIPAL JUSTIFICATIVA

O Portal Acadêmico da Instituição foi escolhido como protótipo do FRW-MVC devido a uma pesquisa realizada entre 435 estudantes de diversos níveis de ensino, para poder ver qual sistema do site da instituição seria o mais requerido para ser visto, de forma amigável, em dispositivos móveis.

Dos 435 alunos que responderam o questionário, que tinha apenas a pergunta de qual sistema do site da instituição deveria ser "transformado" para visualizações amigáveis para celulares e afins, 348 estudantes (80%) colocaram como primeira opção o Portal Acadêmico da instituição, e 87 estudantes colocaram esse portal como segunda opção (20%). 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.

A Figuras 3.a e 3.b apresentam as telas quando se acessa o Portal. A Figura 3.a é a tela de *login* vista por um dispositivo móvel (do tipo celular) e a Figura 3.b apresenta outra tela, vista pelo mesmo dispositivo, contendo os dados do estudante. Ambas telas são modelos retirados do Portal Acadêmico da instituição, sem a formatação fornecida pelo padrão FRW-MVC. Observa-se que, em ambas as telas, existe uma dificuldade de visualização dos dados, o que prejudica a manipulação dos mesmo no dispositivo móvel. Isso ocorre porque não há uma interface adaptável para dispositivos móveis, o que gera a tela com caracteres reduzidos, que vão exigir o artifício de zoom para melhor manipulada.

Figura 3.a. Tela original de *login* do Portal Acadêmico, como visto em um celular comum

Figura 3.b. Tela original de dados do estudante via Portal Acadêmico, vista através de um celular.

3.2 CONSTRUÇÃO DO PROTÓTIPO

O protótipo dessa pesquisa foi construído através 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: class {
    func ready(subject: Subject)
    func success(grades: [String])
}
```

Figura 4: Exemplo de um código Swift utilizado para definir uma classe *delegate* que trata das condições de estado dos diversos outros Models da camada Model do FRW-MVC.

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 será responsável por analisar os dados do HTML através de seletores CSS e gerar um documento legível, para que os dados possam ser preparados e formatados para saída. Existem 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 vai retornar 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 a classe *delegate* apropriada, 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 atua 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 SOBRE OS RESULTADOS OBTIDOS

O aplicativo baseada no FRW-MVC foi testado por 120 estudantes da instituição, sendo 75 estudantes do ensino médio-técnico, 29 estudantes da graduação e dezesseis da pós-graduação. Após ter sido instalado nos celulares desses estudantes, foram dados 37 dias para que eles testassem a interface como um todo.

Após os 37 dias de uso, no 38º dia foi iniciado uma pesquisa, que captou as percepções desses estudantes 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 ASPECTSO DO USO DO APLICATIVO PELOS ESTUDANTES

Nas Figuras 5.a, 5.b, 5.c e 5.d pode-se acompanhar o procedimento de uso do aplicativo desenvolvido através do padrão FRW-MVC. A Figura 5.a apresenta a tela de *login*, equivalente ao Model Login. Uma vez autenticado, o estudante acessa a tela da Figura 5.b, que contém a tela do Model Disciplinas as quais o estudante está matriculado no semestre (se o curso for semestral) ou ao ano (se o curso for anual). Já na Figura 5.c apresenta a tela do Model Notas, que apresenta os dados associados as notas da disciplina escolhida.

O estudante também pode ter acesso ao seu perfil (Model Perfil), visto na Figura 5.d. 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, a camada 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 Model Perfil.

Figura 5.a. Tela de Login do Model Login da FRW-MVC para o Portal Acadêmico, no qual o estudante fornece sua matricula e senha.

Figura 5.b. Tela que se tem acesso após o login, representando o Model Disciplinas, associada ao estudante, usando o mesmo framework.

Figura 5.c. Após escolher uma disciplina, tem-se acesso a tela de notas, do Model Notas da FRW-MVC, no qual pode-se observar e obter todas as notas atualizadas no sistema original.

Figura 5.d. O estudante também pode direcionar, por navegação das funcionalidades, ao Model Perfil, com a tela de perfil dele, no qual pode-se acessar e alterar os campos liberados.

4.2 ANÁLISE DOS RESULTADOS

57 estudantes, dos 120 que usaram o sistema, se propuseram a responder o questionário que lhes foi passado. Os resultados da pesquisa sobre o uso da ferramenta podem ser observados na Tabela 1.

Tabela 1. Resultados da pesquisa com 57 estudantes que utilizaram o aplicativo construído com o padrão FRW-MVC proposto nessa pesquisa.

Aspecto Examinado	***de "Sim"***	***de "Não"***	***de "Não sei"***	***Índice de aprovação***
Visibilidade	53	2	2	93%
Amabilidade	54	1	2	94%
Funcionalidade	57	0	0	100%
Facilidade	56	1	0	98%
Navegabilidade	52	2	3	91%
Conectividade	49	6	2	85%

A interface teve uma aprovação média acima de 90%. Desconsideram-se os que não souberam avaliar os aspectos examinados (os que responderam "Não sei"), por não saberem o significado da proposta, a aprovação chega a 95%. De um modo geral o aspecto que mais se destacou foi a Funcionalidade, isso porque o aplicativo permitiu acesso as funcionalidades de forma plena e estável. Depois dele, a Facilidade foi aspecto mais aprovado pelos usuários, significando que as expectativas em termos do fácil uso foram atendidas em sua quase totalidade.

O aspecto menos aprovado foi o da Conectividade, prejudicado por problemas de rede da instituição. Para essa questão acrescentou-se uma pergunta informal, o motivo da não aceitação plena, o que se informou foi que o carregamento da página excedeu o tempo esperado. Esse aspecto observado pelos que emitiram essas críticas foi devido a lentidão do ambiente de rede proporcionado pela rede da instituição e não foi considerada uma falha do projeto em si.

5 CONSIDERAÇÕES FINAIS

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

Houve uma aprovação significativa para o aplicativo, devido a já formam iniciados planos para desenvolver outros aplicativos, usando o FRW-MVC, para outras aplicações oferecidas pela instituição, tais como, Portal de Notícias, 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 pelo fomento ao trabalho.

REFERÊNCIAS

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.

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.

Matoski , A. et al (2020) Uso de dispositivos móveis como ferramenta de aprendizado: Riscos e Oportunidades. In: Brazilian Journal of Development, v6, n1, pp4673-4687, Brasil.

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

Pereira, J. O. S. et al (2020). Os impactos dos dispositivos móveis dentro do ambiente escolar nas turmas de ensino médio na Cidade de Araguatins-TO. In: Brazilian Journal of Development, v6, n8. Brasil

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.