UTILIZAÇÃO DE PROGRESSIVE WEB APPS PARA DESENVOLVIMENTO DE APLICAÇÕES PARA DISPOSITIVOS MÓVEIS

Tiago Farias Costa[1]

Frederico Pires[2]

RESUMO

Com o aumento do número de smartphones e as consideráveis divergências nas linguagens de programação e nos ambientes de desenvolvimento de cada plataforma, cresce a busca por ferramentas que visam atender as plataformas em um único processo de desenvolvimento, atendendo a demanda de requisitos e reduzindo custos e prazos. As principais ferramentas híbridas utilizam aplicações web compiladas para nativo e/ou apenas linguagem web – caso do Progressive Web Apps (PWA) que visa entregar o melhor da web com o melhor dos apps nativos. O presente trabalho tem como objetivo mostrar os passos do desenvolvimento de uma aplicação utilizando PWA. Para isso, foi modelado e implementado um APP chamado "Meu Corte" para o mercado da beleza, nicho em amplo crescimento. Ao analisar todos os requisitos e verificar a falta de necessidade de recurso nativo, o PWA foi a melhor escolha para minimizar os custos, entregar uma experiência de usuário semelhante à de um aplicativo nativo e entregar um app sempre atualizado ao usuário.

Palavras-chaves: Progressive Web Apps. Dispositivo móvel. Aplicativo.

ABSTRACT

With the increasement in the number of smartphones and the considerable divergences in programming languages and environments progress of each platform, the search for tools to join both platforms in a single development process is growing, attend to requirements and reducing costs and deadlines. The main hybrid tools use compiled web applications for native and/or only for web language – like the Progressive Web Apps (PWA) case that aims to deliver the best of the web with the best of native apps. The present work aims to show the steps of the development of an application using PWA. For this an APP called "Meu Corte" for the market of beauty, a niche in broad growth, was modeled and implemented. By reviewing all requirements and verifying the native resource needlessneeds, PWA was the best choice to minimize costs, to give a user experience similar of a native app and tirn over na always updated app to the user.

Keywords: Progressive Web Apps. mobile device. applicative.

1 INTRODUÇÃO

De acordo com Capelas (2018), conforme a 27ª pesquisa anual de administração e uso de tecnologia da informação nas empresas realizada pela Fundação Getúlio Vargas de São Paulo (FGV-SP), no Brasil existem cerca de 168 milhões de *smartphones* em uso, um crescimento de 9% em relação a 2015. E a expectativa é de crescimento acentuado nos próximos anos, aumento em 40% do número atual.

Dentre esses dispositivos, de acordo com a empresa de análise de dados Kantar, o Sistema Operacional (S.O.) *Android* domina 93% do mercado nacional de dispositivos móveis, seguido do *IOS* com 5,8% e os sistemas *BlackBerry*, *Windows Phone* e outros chegam a 1,1% (KANTAR, 2018). Estes dados

[1] *Aluno do Curso de Pós-Graduação em Desenvolvimento de Software para Dispositivos Móveis da Faculdade Católica do Tocantins - tgfarias@gmail.com.*

[2] *Professor Orientador Faculdade Católica do Tocantins. frederico.pires@catolica-to.edu.br.*

demonstram a diversidade de sistemas e a dificuldade de um aplicativo atender toda essa demanda, pois possuem linguagem própria para o desenvolvimento.

Uma forma de atender aos diversos S.O. existentes é utilizar desenvolvimento híbrido, pois não utiliza linguagem específica de cada plataforma e é mais simples e rápido para o desenvolvimento. Mesmo limitando alguns recursos nativos como Global Positioning System (GPS), câmera, agenda de contatos de forma direta, a tecnologia híbrida abordada neste trabalho, se faz valer através de *container's* que atuam como intermediários entre os recursos da plataforma e dispositivos.

Estes aplicativos híbridos reduzem o custo do desenvolvimento da solução e o tempo para publicação, mas ainda são empacotados como um aplicativo nativo, ou seja, está disponível na loja de aplicativos. Existe ainda os *Web Apps* – não são aplicativos, mas sim sites desenvolvidos para funcionar utilizando o *browser* dos dispositivos, necessita de conexão e não estão presentes em lojas de aplicativos.

Os *Progressive Web Apps* são a evolução dos *Web Apps*, uma vez que são um conjunto de técnicas para desenvolver aplicações *web*, adicionando progressivamente funcionalidades que antes só eram possíveis em *apps* nativos. São para qualquer usuário, independentemente do *browser*, funciona *offline*, é possível adicionar um ícone na área de trabalho, seguro, aceita notificações e o usuário se sente em um aplicativo nativo (LIMA, 2018).

Neste trabalho será apresentado o projeto e desenvolvimento de um aplicativo comercial multiplataforma utilizando o *framework Ionic* com as linguagens *Web HTML5*, *CSS3* e *TypeScript*. O aplicativo será um PWA, ou seja, compilado para os *browsers*, e será chamado de "*Meu Corte*". Será utilizado como auxílio de um sistema de gestão integrada para barbearias e salões de beleza.

A escolha do mercado da beleza se deu principalmente pelo avanço desta área no país aliado à falta de uso de tecnologia neste nicho. Muitas vezes, todo o controle da empresa é realizado por meio de anotações e incessantes cálculos através de calculadoras, o projeto visa atender à demanda e informatizar este mercado carente em tecnologia, buscando acelerar os processos e oferecer maior transparência para os profissionais.

O aplicativo fará lançamento de comandas de serviços vinculando cliente e funcionário, pesquisa de comandas abertas para adicionar novo serviço à mesma, consulta de agendamentos cadastrados na retaguarda *web*, e consultas de relatórios com filtros diários, mensais e entre datas. Para acessar o aplicativo é necessário a solicitação de criação de usuário e senha pela retaguarda.

Após esta introdução, a seção seguinte deste artigo, trata do levantamento bibliográfico sobre as tecnologias de desenvolvimento de aplicativos *mobile*. Na terceira seção uma breve informação sobre o mercado da beleza e seu crescimento no âmbito nacional. Na quarta seção são apresentados os processos do desenvolvimento do aplicativo, cujos resultados são discutidos na quinta seção e, por fim, apresentada uma conclusão do trabalho.

2 REVISÃO DA LITERATURA

Antes de falar sobre o objetivo principal deste artigo (desenvolvimento híbrido utilizando PWA) é necessário conhecer outras formas para construir um aplicativo *mobile* como aplicativo nativo, *web apps* e aplicativo híbrido.

2.1 APLICATIVO NATIVO

Um aplicativo para ser chamado de nativo precisa ser programado em uma linguagem exclusiva para um sistema operacional e se encontrar na loja de aplicativos após aprovação. Os mais utilizados atualmente são o *Android* e o *iOS*. No *Android,* a linguagem de programação é *Java* e *Kotlin* e no *iOS* é *Objective-C* e *Swift*, com isso o aplicativo desenvolvido em uma linguagem não funciona para a outra.

Aplicativo nativo é mais confiável e rápido, apresentam uma melhor experiência para o usuário, funcionam sem internet e utilizam todos os recursos oferecidos pelos dispositivos móveis facilmente como câmera, GPS, *notificações push* e contatos. O ambiente de programação para o desenvolvimento é específico para cada linguagem, *Android Studio* para *Android* e *Xcode* para *iOS*.

Para desenvolver aplicações nativas, faz-se uso de um conjunto de ferramentas, bibliotecas e compiladores chamados de *Software Development Kits* – SDKs que são disponibilizados pelo fabricante do sistema operacional. Estes SDKs são exclusivos para cada sistema operacional. O código fonte da aplicação deve ser escrito na linguagem específica para cada plataforma. Em seguida, este código é compilado em um arquivo executável com uso das ferramentas disponibilizadas, incluindo as bibliotecas e arquivos adicionais (MURAROLLI; GIROTTI, 2015).

Segundo Lopes (2013), aplicativos nativos serão sempre mais rápidos que a *Web*, eles rodam diretamente no S.O. e, na maioria dos casos, são escritas nativamente para a plataforma específica, o que dá muita performance. Outro quesito é a questão da monetização, pois as lojas de *apps* já são plataformas de pagamentos integradas, facilitando a venda de *apps* e assinaturas.

A necessidade de desenvolver uma solução nativa para cada S.O. é uma das desvantagens de aplicativos nativos pois implica no aumento dos custos, do tempo e da quantidade de recursos para finalizar o projeto. Existem ainda necessidade de obter uma licença em cada uma das lojas de aplicativos aumentando ainda mais o custo final da solução.

2.2 WEB APPS

Segundo Eis (2018), o desenvolvimento *web* depende de três personagens principais, W3C (*World Wide Web*), que regulamenta, cria e sanciona padrões para a *web*. Os *browsers* importam essas regras e padrões e os desenvolvedores aplicam padrões em projeto para criação na web. O *HTML5* apresenta melhorias no controle de conteúdo multimídia, aprimoramento no uso *offline*, oferecendo uma experiência *web* totalmente melhorada para usuários.

Web Apps são aplicações desenvolvidas para a *Web*, utilizando as linguagens *HTML5*, *CSS* e *JavaScript* – tecnologias *web* padrão, no lado do cliente e outras linguagens como *PHP*, *Java* no lado do servidor. Trata-se de sites desenvolvidos exclusivamente para dispositivo móvel, se adaptando responsivamente a cada tamanho de tela dos *smartphones. Web App* não é um aplicativo real e não será encontrado em nenhuma loja de aplicativos.

O grande apelo da *Web* é ser independente de plataforma, ou seja, multiplataforma, pois são executados diretamente no *browser* do aparelho sem consumo de memória do dispositivo (Lopes, 2013). Faz-se necessário uma conexão com a internet para ser acessado e não faz uso de todas as funcionalidades dos *smartphones*, são mais lentos e menos seguros que os aplicativos nativos, pois não são integrados ao S.O.

Na *web,* os *sites* ou *web apps* têm o estilo diretamente ligado à identidade visual da marca e da empresa e é bastante comum ter uma linguagem visual única na *Web* independentemente da plataforma que está sendo executada (LOPES, 2013). Um exemplo clássico é o botão voltar nos navegadores instalados em qualquer plataforma e os aplicativos nativos em *Android* – possui botão físico, e no *iOS* – maioria dos *apps* possui o botão voltar.

O processo de distribuição e instalação dos *Web Apps* são bem mais facilitados que os aplicativos nativos, pois não são distribuídos em lojas de aplicativos, mas em servidores de hospedagem – sistema de computação centralizada que fornece serviços a uma rede. Esta característica facilita as atualizações do site, pois não possui a necessidade de enviar as novas funcionalidades para loja, o usuário sempre acessa a versão mais recente quando navega.

2.3 APLICATIVO HÍBRIDO

Geralmente, são desenvolvidos com um conteúdo *HTML5* e são executados dentro de um ambiente de processo nativo na plataforma do dispositivo. A execução é realizada por meio de um navegador embutido em tela cheia sem a barra de endereço e os demais controles, chamado *WebView*. Esses aplicativos são disponibilizados nas lojas de aplicativos nativos e podem ser executados sem estar conectados à internet (PREZOTTO, 2014).

Para acessar os recursos nativos do dispositivo móvel (câmera, GPS, gestos, contatos e etc.), os aplicativos fazem uso de uma camada de abstração escrita em *JavaScript* para fazer a ponte entre *Application Programming Interface* (API) nativa e conteúdo *Web*. Aplicativos híbridos ficam exatamente entre aplicativos nativos e *Web Apps*. Sua principal vantagem é permitir desenvolvimento multiplataformas.

Para um bom desempenho na produção de aplicações híbridas é necessário fazer uso de um *framework* de desenvolvimento multiplaforma que são blocos de códigos reutilizáveis que permite a produção de aplicações customizadas. Segundo Prezotto (2014), *frameworks* devem possuir linguagem unificadora, ou seja, a mesma para todas as plataformas e APIs que permitam acessar da mesma forma os recursos nativos dos sistemas operacionais.

Segundo Grillo (2015), o *Ionic* é um *framework* para desenvolvimento de aplicações para dispositivos móveis que visa o desenvolvimento de *apps* híbridas e de rápido e fácil desenvolvimento. É um *framework* baseado em *Angular* e *Cordova*, contando com uma gama de *plugins* e funcionalidades nativas para criar e compilar uma aplicação *mobile* ou uma *Progressive Web App* (PWA) perfeita (IONIC BRASIL, 2018).

Cordova é a solução mais comum atualmente e faz prover uma casca nativa para aplicativo escrito com linguagens da *web*, ele cria uma janela de navegador para o *app* e faz a comunicação das chamadas de código para chamadas nativas quando necessário (LOPES, 2016). Quando é utilizado o *Cordova*, todo o HTML é empacotado junto à casca nativa, se tornando um aplicativo normal.

O *apache Cordova* possui seu código aberto e permite o uso de tecnologias *web* padrão para desenvolvimento multiplataforma. E sua principal vantagem é o pacote de *plugins* que ele oferece acessando as funções do dispositivo de várias plataformas com um único código (PREZOTTO, 2014). Além dos *plugins* oferecidos no *Cordova*, existem contribuições de terceiros que podem ser agregados ao seu projeto ou até mesmo desenvolver seu próprio *plugin* (CORDOVA, 2018).

O *Ionic* possui uma ferramenta *CLI* (*Command Line Interface*) que permite executar diversos comandos para facilitar o desenvolvimento das aplicações, estes comandos vão desde criação de um novo projeto utilizando template pré-estabelecido até a compilação final do *app* (GOIS, 2017). Para criar um novo, o comando é *ionic start nome template*, os template já prontos básicos são o *blank*, *tabs* e *sidemenu*. E para testar uma aplicação basta inserir o comando *ionic serve*.

2.4 PROGRESSIVE WEB APPS

Segundo Developers (2018), os *Progressive Web Apps* são experiências que combinam o melhor da *Web* e o melhor dos aplicativos. Eles são úteis para os usuários desde a primeira visita em uma guia de navegador sem exigir instalações. São um conjunto de técnicas para desenvolver aplicações *web*, adicionando progressivamente funcionalidades que antes eram possíveis em *apps* nativos.

Para ser um PWA é necessário conter algumas características básicas como funcionar *offline*, disponível para qualquer usuário, experiência de um aplicativo nativo, aceitar *push notifications*, adaptável a qualquer tamanho de tela, facilitar a adição do ícone na tela principal do aparelho e rodar em *https.* Um PWA pode atender perfeitamente os requisitos de um aplicativo no qual não se usa recursos nativo, gerando economia considerável no desenvolvimento (DEVELOPERS, 2018).

Para o PWA rodar *offline* necessita de um *Service Worker* que é um *script* que seu navegador executa em segundo plano, separado da página *Web*, possibilitando recursos que não precisam de uma página *Web* ou interação do usuário. Não possui acesso ao *Document Object Model* (DOM), possui recursos de sincronização, notificações *push* e um ciclo de vida bem definido (JUSTEN, 2018).

Os *service workers* e a memória *cache* concedem algumas características nativas a este tipo de *web apps*, executando funções em segundo plano e *offline* podendo assim receber notificações como nos aplicativos nativos. Estas funcionalidades só têm uso, pois os principais navegadores estão adotando medidas para suportar PWAs como um todo. Dentre eles, o que possui menos progresso neste sentido é o *Safari* da *Apple* (MOTA, 2018).

O *Ionic framework* já possui suporte para PWA que agiliza os processos como o arquivo *manifest.json* – arquivo manifesto que segue a especificação da W3C, atribuindo ao aplicativo os ícones, a cor de fundo, tema e exibição em tela cheia, o *Service Workers* já vem criado ao iniciar uma aplicação utilizando *Ionic*, basta apenas referenciar o arquivo no *index.html* e servem para executar e fazer *cache* das requisições e arquivos do projeto.

A equipe do *Google Chrome* fornece uma biblioteca de alto nível para ajudar a lidar com as tarefas de um *Service Workers*, a *sw-toolbox*, auxiliando nas funções para oferecer uma experiência de usuário bem próxima de um aplicativo nativo. Existem algumas estratégias de cache para diferentes casos de uso, são elas: *cacheFirst*, *cacheOnly*, *networkFirst* e *networkOnly*.

Existem cases de sucesso utilizando PWA como Telegram, Twitter, *AliExpress*, Facebook, entre outros. São empresas que estão adotando esta tecnologia e abrindo o leque para usuários com dispositivos sem muito recursos. Para a empresa Google, a criação de um PWA de alta qualidade traz benefícios incríveis, facilitando a satisfação de seus usuários, o aumento de engajamento e o aumento de conversões.

3 MERCADO DA BELEZA

O mercado da beleza é bastante diversificado e lucrativo, o setor está se tornando cada vez mais forte e o Brasil tornou-se o terceiro país com o maior mercado estético no mundo, ficando atrás apenas dos Estados Unidos e China (DINO, 2018). Segundo Sebrae (2018), cerca de 7 mil salões de beleza são abertos por mês e ultrapassam a casa de 1 milhão de estabelecimentos.

Este é um nicho de mercado bastante segmentado, como grandes salões que reúnem serviços *premium* em um ambiente de alta classe e espaços segmentados como barbearias, esmalterias que priorizam um serviço eficiente e específico. Da mesma forma, há diversidade no tamanho do estabelecimento, que são desde um único funcionário até acima de vinte.

Para o crescimento do negócio é fundamental investir em software de gestão para controlar as atividades da empresa de forma eficiente, melhorando a organização, o atendimento ao público e otimizando funções e processos do estabelecimento, abandonando anotações diversas, eliminando erros de cálculos e economizando tempo na administração do seu negócio.

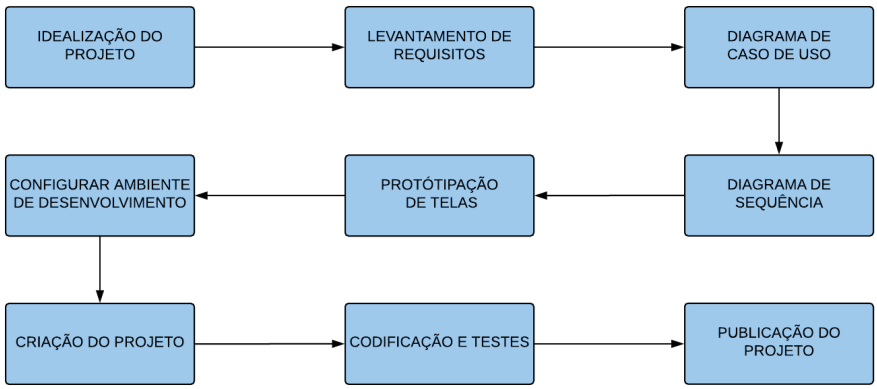
4 METODOLOGIA

Com base em uma pesquisa exploratória, utilizando pesquisa de forma bibliográfica atualizada e fundamentada para adquirir amplo conhecimento do assunto proposto e maior familiaridade com a solução do problema foi idealizado e executado o desenvolvimento de um aplicativo utilizando PWA cujo fluxograma na figura 1 apresenta a passos utilizados para publicação deste projeto no mercado.

Figura 1 – Fluxograma dos processos de desenvolvimento

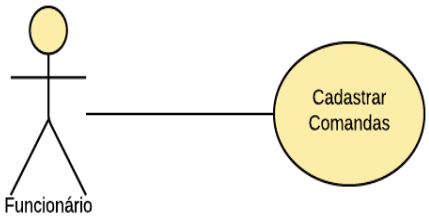
Fonte: AUTORES

A ideia proposta foi auxiliar os profissionais do mercado de beleza a efetuar lançamentos de serviços em comandas via aplicativo minimizando o tempo com anotações e falhas em cálculos de comissões. Os requisitos para produção do *app* foram: o cadastro de comandas envolvendo serviços e clientes, realizados por funcionários pré-cadastrados no sistema e o envio dos dados através de uma API para interoperabilidade entre aplicações.



Na figura 2 fica demonstrado o diagrama de caso de uso que é de extrema relevância para a definição dos caminhos a serem tomados pelo usuário, a fim de reduzir falhas.

Figura 2 – Diagrama de Caso de Uso



Fonte: AUTORES

A descrição do caso de uso contém todas as possíveis ações que o ator realizará para finalizá-lo. O quadro 1 contempla a descrição do caso de uso do requisito apresentado.

Quadro 1 – Descrição de Caso de Uso

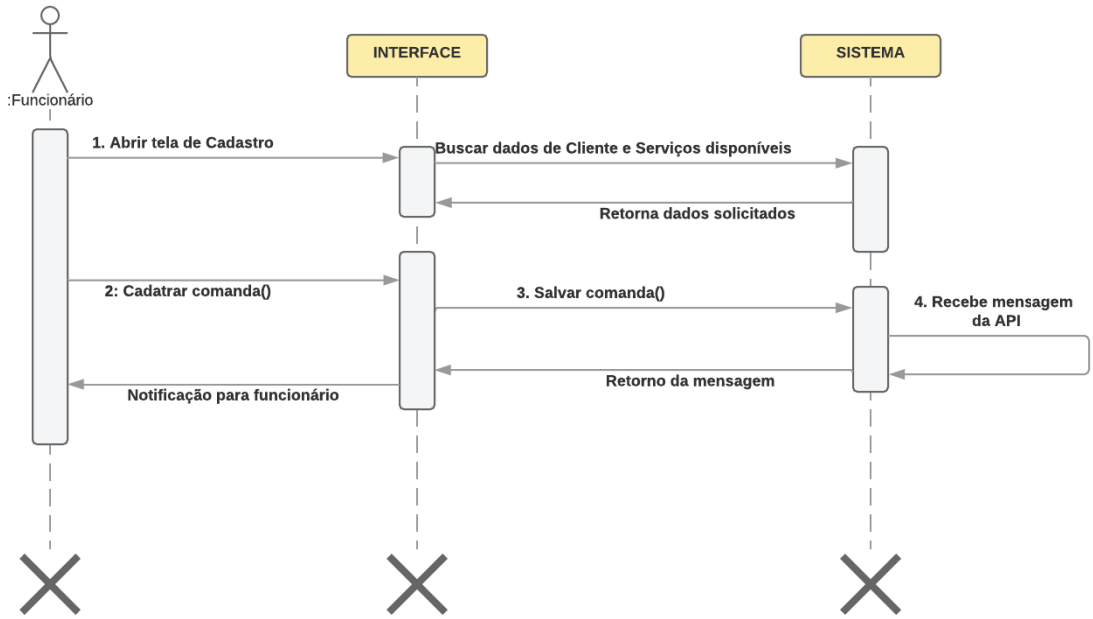
Caso de Uso:	**Cadastrar comanda**
Finalidade:	Realizar cadastro de comanda
Descrição:	Permite ao funcionário realizar um lançamento de comanda.
Ator:	Funcionário
Pré-condições:	Realizar autenticação no sistema
Fluxo Principal:	1. Selecionar serviço(s) 2. Calcular valores 3. Conferir resultado 4. Confirmar dados.
Fluxo Alternativo:	1. Inserir número de comanda 2. Selecionar cliente 3. Selecionar serviço 4. Editar valor do serviço 5. Adicionar valor para desconto 6. Calcular valores 7. Confirmar dados
Pós Condições:	Comanda cadastrada com sucesso.

Fonte: AUTORES

O diagrama de sequência representa como os objetos interagem na execução do comportamento total do caso de uso proposto. A ferramenta utilizada para gerar os diagramas foi o *Astah* que pode ser

visualizado na figura 3.

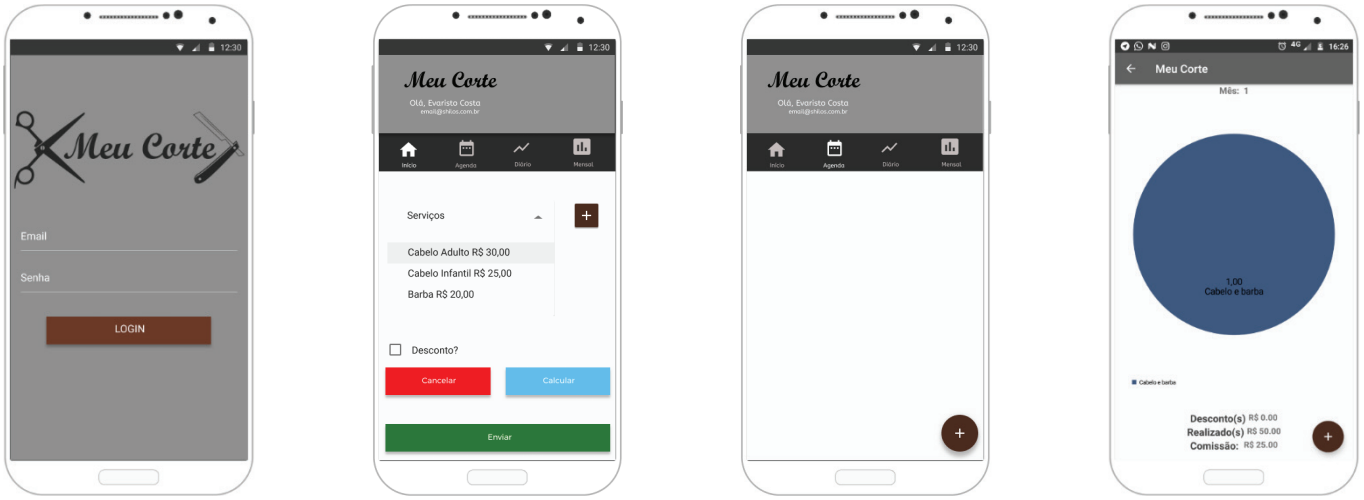
Figura 3 – Diagrama de Sequência



Fonte: AUTORES

O protótipo de tela é um esboço do *layout* do sistema. Foi utilizada a ferramenta *Figma* para prototipação de telas. A figura 4 apresenta o protótipo das telas do projeto.

Figura 4 – Protótipo das telas do aplicativo



Fonte: AUTORES

Para o desenvolvimento do projeto foi necessária a configuração do ambiente com a instalação do *Ionic* na versão 3, com o comando *npm install –g ionic*, logo após a instalação do *NodeJS* e o *NPM* que são dependências do *framework*, na parte *web* será utilizado *AngularJS* que é componente do *Ionic* e utiliza a linguagem *TypeScript*. Para escrever códigos foi utilizado o *Visual Studio Code,* com acesso direto ao *Prompt de Comando* (CMD).

Para criar o formulário de cadastro foi utilizado o comando *ionic generate page comanda*, gerando os arquivos *comanda.html*, *comanda.scss*, *comanda.ts* e *comanda.module.ts*, todos estes são gerados no diretório *src/pages/comanda*. Utilizando campos do *framework* ou do próprio *HTML5* é possível a confecção de um formulário com *layout* amigável para o usuário.

A conexão com API externa é realizada através de provedores de acesso, pelo comando *ionic generate provider getapi*, os métodos de requisições *GET* e *POST* utilizam protocolo *Http* para obter e enviar dados externos e *RequestOptions* para adicionar as requisições os parâmetros de cabeçalho, ambos importados do *@angular/http*. As requisições usam *Promisses* que são objetos que representam a conclusão ou falha de uma operação assíncrona.

Após a criação dos formulários e provedores de acesso à API, utilizando os comandos para realizar os testes necessários e verificação de possíveis falhas para correção, o comando *ionic serve* executa a aplicação em ambiente de testes e abre no navegador padrão da máquina. Usando o parâmetro *--lab* é possível realizar teste em todos os dispositivos utilizando a ferramenta *Ionic Lab*.

Para seguir os padrões do PWA, o *service workers* foi escrito para armazenar em *cache* todos os arquivos estáticos como imagens, estilos, arquivos JavaScript e HTML. O padrão utilizado para arquivos estáticos foi o *cacheFirst* e para os dados obtidos da API foi o *networkFirst* ambas estratégias são utilizadas para fornecer uma experiência *offline* ao usuário.

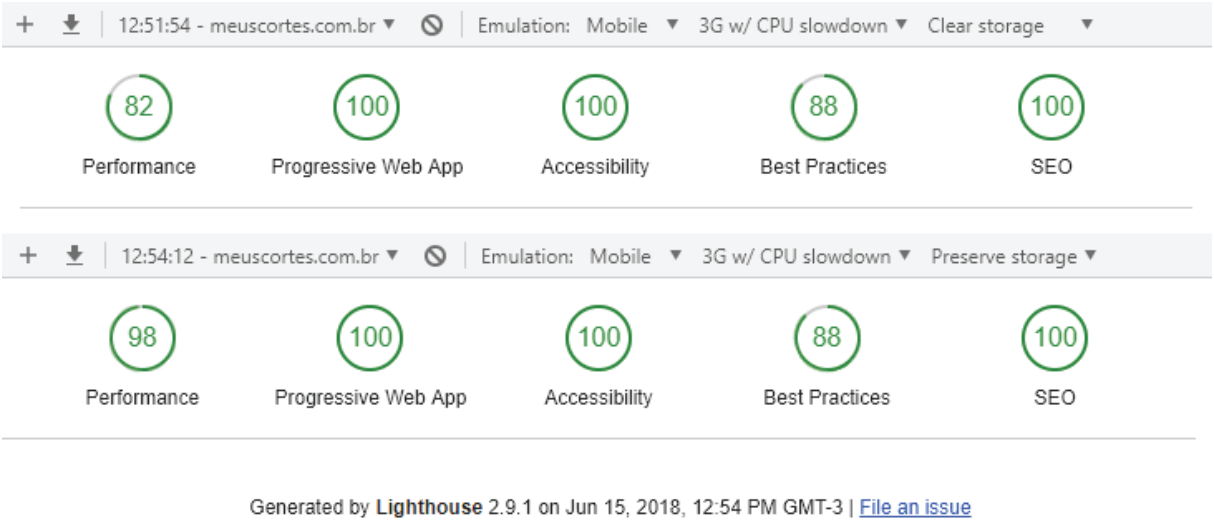
5 DISCUSSÃO DOS RESULTADOS

Os testes durante o desenvolvimento do aplicativo foram realizados com o navegador Google *Chrome*, utilizando as ferramentas de desenvolvimento (F12) integradas como *Toggle Device Mode* para simular um dispositivo móvel e a depuração remota de *WebViews* que permite testes em dispositivos reais, no endereço *chrome://inspect*. Toda esta fase de testes serviu para melhorar o *layout* do *app* e reduzir as falhas de programação.

Após toda a fase de teste, a compilação do projeto para produção é necessário rodar o comando para dar *build* no projeto [*npm run build --prod --release*], pois ele minimiza os códigos fontes deixando os scripts mais leves com isso eleva os índices de desempenho que serão apresentados a seguir. A implantação do aplicativo foi realizado em um servidor *web* com suporte para protocolo *https*, este sendo um requisito obrigatório para PWAs.

Os índices de desempenho são medidos pelo *Lighthouse* que realiza auditoria no projeto verificando itens como performance, acessibilidade, boas práticas, SEO (*Search Engine Optimization*) e principalmente se o app é um PWA com todos os seus requisitos. A figura 5 representa o resumo da auditoria realizada no aplicativo gerado neste trabalho.

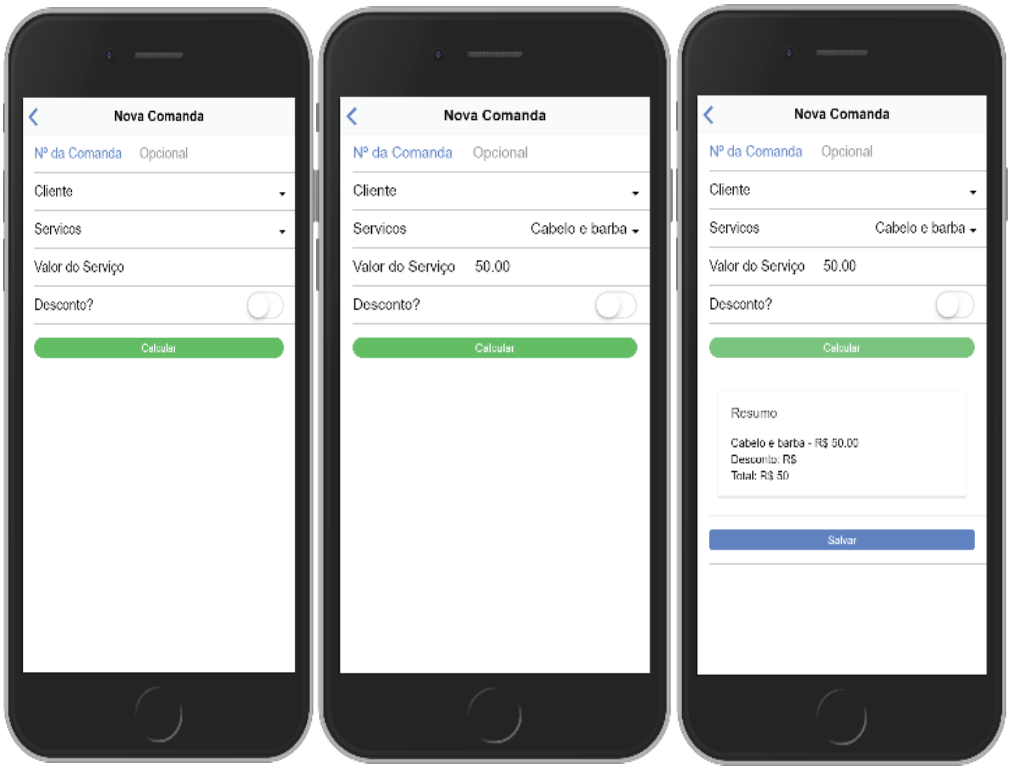
vFigura 5 – Resumo da auditoria do Lighthouse



Fonte: Auditoria *Lighthouse* no *Google Chrome*

A versão final do aplicativo proposto neste trabalho está demonstrada na figura 6. Após realizar login no *app*, o funcionário segue para tela de cadastro de comandas e preenche os dados principais para inserir o registro, os dados serão salvos na base de dados através da API conectada.

Figura 6 – Fluxo principal do Caso de Uso



Fonte: AUTORES

6 CONCLUSÃO

Ao utilizar PWA como tecnologia para entrega de aplicativos reduz-se totalmente o custo com lojas de aplicativos, sua publicação é realizada em servidores de aplicação *web*. Com o avanço das funcionalidades para apresentar o melhor da *web* com o melhor dos *apps*, o PWA se mostra capaz de minimizar as dificuldades para encontrar programadores especialistas em cada linguagem para S.O. nativo e oferecer uma boa experiência ao usuário.

O objetivo do presente trabalho foi desenvolver um aplicativo multiplataforma utilizando PWA como ferramenta de desenvolvimento. Foi possível desenvolver um único código utilizando linguagem padrão *Web* e apresentar nas duas principais plataformas para smartphones do mercado (*Android* e *iOS*) sem alteração da apresentação visual e com desempenho satisfatório.

A apresentação do aplicativo para as plataformas principais do mercado se mostrou bastante eficaz, conforme resultados do processo de auditoria da equipe do Google. Para o usuário final, após a instalação do PWA no primeiro acesso, fica invisível a escolha de tecnologia, pois o aplicativo executa como um nativo, com ícone na tela do dispositivo, velocidade no acesso das informações e com funcionamento *offline*.

É possível obter um excelente resultado utilizando esta ferramenta de desenvolvimento, mas é necessário analisar todos os requisitos do projeto, uma vez que se houver a necessidade de uso dos recursos nativos que não são abordados pelo PWA, esta não será a melhor escolha, é de extrema importância a análise de todos os requisitos do projeto antes da escolha de tecnologia.

REFERÊNCIAS

CAPELAS, Bruno. **Brasil chega a 168 milhões de smartphones em uso**. Disponível em: <https://link.estadao.com.br/noticias/gadget,brasil-chega-a-168-milhoes-de-smartphones-em-uso,10000047873>. Acesso em: 11 jun. 2018.

CORDOVA, A. **Architectural overview of Cordova platform**. Disponível em <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>. Acesso em: 13 de jun. 2018.

DEVELOPERS, Google. **Your first Progressive web app**. Disponível em <https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp>. Acesso em: 12 jun. 2018.

DINO. **O fortalecimento do mercado da beleza em 2018**. Disponível em: <https://exame.abril.com.br/negocios/dino/o-fortalecimento-do-mercado-da-beleza-em-2018>. Acesso em: 11 jun. 2018.

EIS, Diego. **Evolução Web**: Personagens principais no desenvolvimento. Disponível em: <https://digitalks.com.br/artigos/cliclo-desenvolvimento-web/>. Acesso em: 14 jun. 2018.

GOIS, Adrian. **Ionic Framework**: Construa aplicativos para todas as plataformas mobile. São Paulo: Editora Casa do Código, 2017.

GRILLO, Rafael. **Introdução ao Ionic Framework**. Publicado em 26 de fev. de 2015. Disponível em: <http://tableless.com.br/introducao-ao-ionic-framework/> Acesso em: 12 jun. 2018.

IONIC BRASIL. **Introdução ao Ionic 3.x**. Disponível em <ionicbrasil.com/introducao-ao-ionic-3-x/>. Acesso em: 12 jun. 2018.

JUSTEN, Willian. **Como fazer seu site funcionar offline com PWA**. Disponível em: <https://willianjusten.com.br/como-fazer-seu-site-funcionar-offline-com-pwa/>. Acesso em: 12 jun. 2018.

KANTAR. **Android vs. IOS**. Disponível em <https://www.kantarworldpanel.com/global/smartphone-os-market-share/>. Acesso em: 11 jun. 2018.

LIMA, Matheus. **Introdução aos Progressive Web Apps**. Disponível em <https://medium.com/tableless/introdução-aos-progressive-web-apps-ad47ba24cddb>. Acesso em: 11 jun. 2018.

LOPES, S. **A Web Mobile**: Programe Para um Mundo de Muitos Dispositivos. São Paulo: Editora Casa do Código, 2013.

LOPES, S. **Aplicações mobile híbridas com Cordova e PhoneGap**. São Paulo: Editora Casa do Código, 2016.

MOTA, Eduardo. **O que são as PWA e de que modo mudarão a forma como usamos os nossos dispositivos?** Disponível em: <https://pplware.sapo.pt/software/pwa-mudarao-forma-usamos-dispositivos/>. Acesso em: 14 jun. 2018.

MURAROLLI, P. L; GIROTTI, M. T. **Inovações Tecnológicas nas Perspectivas Computacionais**. São Paulo: Biblioteca 24 Horas, 2015.

PREZOTTO, Ezequiel Douglas. **Estudo de frameworks multiplataforma para desenvolvimento de aplicações mobile híbridas**. Universidade Federal de Santa Maria, nov. 2014. Anais do EATI, Frederico Westphalen – RS. Ano 4 n. 1, p.72-79. Trabalho de Conclusão de Curso.

SEBRAE. **Vale a pena montar um salão de beleza?** Disponível em: <www.sebrae.com.br/sites/PortalSebrae/artigos/vale-a-pena-montar-um-salao-de-beleza,efb8d62b2b886410VgnVCM1000003b74010aRCRD>. Acesso em: 11 jun. 2018.