INTERFACE TECNOLÓGICA

# PROGRESSIVE WEB APPS: avaliar uma aplicação em diferentes navegadores e sistemas operacionais

## *PROGRESSIVE WEB APPS: evaluate an application in different browsers and operating systems*

Brian Kazuyoshi Izaki – brian.izaki@gmail.com
Faculdade de Tecnologia de Taquaritinga – Taquaritinga – São Paulo - Brasil

Fernando Tiosso – fernando.tiosso@fatectq.edu.br
Faculdade de Tecnologia de Taquaritinga – Taquaritinga – São Paulo - Brasil

Helena Macedo Reis – helena.macedo@ufpr.br
Universidade Federal do Paraná (UFPR) – Jandaia do Sul – PR – Brasil

**DOI: 10.31510/infa.v18i1.1154**
Data de submissão: 17/04/2021
Data do aceite: 09/07/2021
Data da publicação: 30/07/2021

## RESUMO

Este trabalho tem como objetivo avaliar a compatibilidade de uma *Progressive Web Application* (PWA) com relação ao seu suporte para uma experiência de aplicação nativa nos Sistemas Operacionais (SOs) Android e iOS utilizando os navegadores Chrome, Firefox, Samsung Internet e Safari. Por meio de pesquisas bibliográficas em livros, documentações e sites de fabricantes, observou-se quais as características mais relevantes que poderiam ser utilizadas durante a análise da aplicação para dispositivos *mobiles* até o presente momento, proporcionando o desenvolvimento e a análise de uma PWA devidamente auditada e reconhecida pelo aplicativo Lighthouse. Assim, por meio das avaliações realizadas, constatou-se que o sistema operacional Android e o navegador Chrome ofereceram maior suporte para uma experiência de aplicação nativa utilizando a abordagem progressiva.

**Palavras-chave:** PWA. Aplicação para dispositivos móveis. Lighthouse. React.

## ABSTRACT

This work aims to evaluate the compatibility of a Progressive Web Application (PWA) in relation to its support for a native application experience on Android and iOS Operating Systems (OSes) using Chrome, Firefox, Samsung Internet and Safari browsers. Through bibliographic searches in books, documentation and manufacturers websites, it was observed which are the most relevant characteristics that could be used during the analysis of the application for mobile devices until the present moment, providing the development and analysis of a properly audited PWA and recognized by the Lighthouse app. Thus, through the

evaluations carried out, it was found that the Android operating system and the Chrome browser offered greater support for a native application experience using the progressive approach.

**Keywords:** PWA. Application for mobile devices. Lighthouse. React.

## 1 INTRODUÇÃO

Segundo Lepage e Richard (2020), as *Progressive Web Apps* (PWAs) pertencem ao novo ciclo da evolução Web, pois caracterizam-se por aplicações que podem fornecer uma experiência de aplicação nativa ao usuário final com disponibilidade e confiabilidade, alcançando qualquer pessoa, lugar e/ou dispositivos.

Atualmente, além de serem amplamente utilizadas por empresas como Twitter, Tinder, Spotify, Pinterest, entre outras espalhadas pelo mundo, as tecnologias que envolvem uma PWA estão em constantes aprimoramentos, como, por exemplo, as utilizadas no projeto aberto do Chromium, contando com contribuidores como Google, Microsoft e Intel e o projeto Capabilities Project, nomeado de Fugu Project, que visa permitir que *websites* realizem as mesmas ações que aplicações nativas por meio dos navegadores (LIEBEL, 2020).

No entanto, ainda existem diferenças no suporte de algumas características da PWA entre os principais *browsers* utilizados no mercado, tais como Chrome, Firefox, Safari e Samsung Internet e seus respectivos sistemas operacionais Android e IOS. Nos *browsers*, as principais diferenças ocorrem em caminhos para instalação da aplicação, suportes para acessar recursos nativos e o arquivo de *manifest* (PONTES, 2018, p. 150). Já nos sistemas operacionais, destacam-se ícones, *splash screens*, apresentação dos recursos nativos como acesso às câmeras e microfone, integração com outras aplicações e notificações.

Com o objetivo de avaliar as principais características de uma PWA em navegadores e sistemas operacionais utilizados em dispositivos móveis visando certificar o seu suporte para uma experiência de aplicação nativa, este trabalho proporcionou o desenvolvimento de uma PWA utilizando o conceito de *Single Page Application* (SPA) e a biblioteca React, onde o usuário poderá fazer anotações de livros com imagens e/ou textos de partes que julgar mais importante durante uma leitura.

Assim, a Seção 2 apresenta uma breve fundamentação teórica das tecnologias envolvidas no desenvolvimento da aplicação progressiva, a Seção 3 apresenta o processo de desenvolvimento de uma aplicação, a Seção 4 relata a avaliação das características da aplicação desenvolvida durante sua configuração e funcionamento nos browsers Chrome, Firefox, Safari

e Samsung Internet por meio dos sistemas operacionais Android e IOS e, por fim, na Seção 5 serão apresentadas as considerações finais da avaliação realizada e temas que poderão ser abordados futuramente dentro do contexto das PWAs.

## 2 FUNDAMENTAÇÃO TEÓRICA

Com o intuito de fundamentar as tecnologias e metodologias utilizadas para a avaliação de uma PWA entre navegadores e sistemas operacionais para dispositivos móveis, realizou-se uma revisão da literatura por meio de pesquisas bibliográficas em livros, documentações e sites de fabricantes, destacando as características mais relevantes que poderiam ser analisadas com o objetivo de nortear esse trabalho (PIZZANI et al, 2012, p. 54).

Segundo a PNAD (2020, p. 7), 99,2% dos domicílios que utilizavam a internet no Brasil em 2018 dispunham de um telefone móvel celular para este fim. Portanto, com base nesta pesquisa, o presente artigo tomou como foco o desenvolvimento de uma interface de usuário voltada para dispositivos móveis.

Neste contexto, o uso de navegadores para dispositivos móveis torna-se fundamental para o funcionamento das PWAs. Assim, para determinar quais deles seriam utilizados no experimento, tomou-se como base as informações apresentadas pela empresa Statcounter (2020) que fornece dados dos navegadores mais utilizados em dispositivos mobile no Brasil, sendo o Chrome líder com 78,4%, seguido do Safari com 13,65% e, logo depois, o Samsung Internet com 6,43% de utilização. No entanto, apesar do navegador Firefox não estar entre os mais utilizados, ele é muito difundido em meio a comunidade de código livre, principalmente na área de desenvolvimento de sistemas, e, por isso, também foi utilizado na avaliação.

Haja vista o conceito da aplicação progressiva, outra etapa importante é a criação de um protótipo que, segundo Sommerville (2011, p. 30), pode ser utilizado para apresentar conceitos, experimentar opções de projeto e conhecer mais sobre o problema e suas possíveis soluções, permitindo que o usuário possa ter o primeiro contato com a aplicação de forma mais rápida.

Com o objetivo de suportar o desenvolvimento da aplicação, utilizou-se o conjunto de ferramentas e bibliotecas disponíveis no módulo Create React App (CRA), tais como: Babel, Webpack e React (REACT, 2020a), visando facilitar a preparação de um ambiente de desenvolvimento otimizado para a produção de uma SPA com base nos conceitos de aplicação progressiva, além da utilização da biblioteca Pure.css para estilização dos componentes desenvolvidos.

O motivo de destaque para a utilização da biblioteca React foi a facilidade que ela proporciona no desenvolvimento da interface do usuário (IU), pois utiliza componentes em seu desenvolvimento que facilitam a captura das reações dos usuários, bem como um conjunto de depurações para otimizar a IU (REACT, 2020b). Ademais, o Babel é um *transpiler* que possibilita a conversão de código JavaScript (JS) para diferentes versões, mantendo a compatibilidade do JS entre os navegadores novos e antigos (PONTES, 2018, p. 79). Por fim, o Webpack possui o objetivo de "[...] juntar os arquivos JS, organizar as imagens, minificar o Cascading Style Sheet (CSS) e injetar scripts no HyperText Markup Language (HTML)." (PONTES, 2018, p. 106) possibilitando a construção de uma aplicação pronta para ser disponibilizada em produção.

Com relação aos conceitos voltados para estilização dos componentes da aplicação, utilizou-se o *framework* Pure.css por possuir códigos predefinidos e otimizados para formatação e, devido a sua vantagem de ser leve em relação ao tamanho do arquivo quando comparados às demais bibliotecas existentes no mercado, possibilita o rápido carregamento no primeiro acesso da aplicação (PONTES, 2018, p. 66 e ANDRADE, 2018).

Para verificar se os requisitos de uma PWA foram corretamente atendidos, utilizou-se a ferramenta Lighthouse que, de acordo com o artigo do Google Developers (2017), é uma ferramenta de auditoria automatizada que verifica o desempenho da página web e gera relatórios que podem demonstrar falhas e suas possíveis soluções, mas, seu principal foco, atualmente, é verificar se a aplicação atende aos requisitos de uma PWA, como a opção "adicionar na tela inicial", suporte offline, utilização do protocolo HTTPS, utilização de *meta-tags* para otimização da página, entre outras verificações.

Portanto, após o embasamento teórico e a justificativa da escolha das tecnologias presentes no ambiente de desenvolvimento de uma aplicação progressiva, conclui-se esta fundamentação teórica e apresenta-se a seção de materiais e métodos utilizados no experimento.

## 3 MATERIAIS E MÉTODOS

Com o intuito de avaliar as principais características de uma aplicação progressiva em diferentes navegadores e sistemas operacionais para dispositivos móveis, este artigo apresenta o processo de desenvolvimento de uma aplicação progressiva que permite o usuário realizar anotações sobre livros, contendo funções de cadastro e exclusão dessas anotações, fotos por meio do acesso a câmera do dispositivo e armazenamento dos dados utilizando os locais

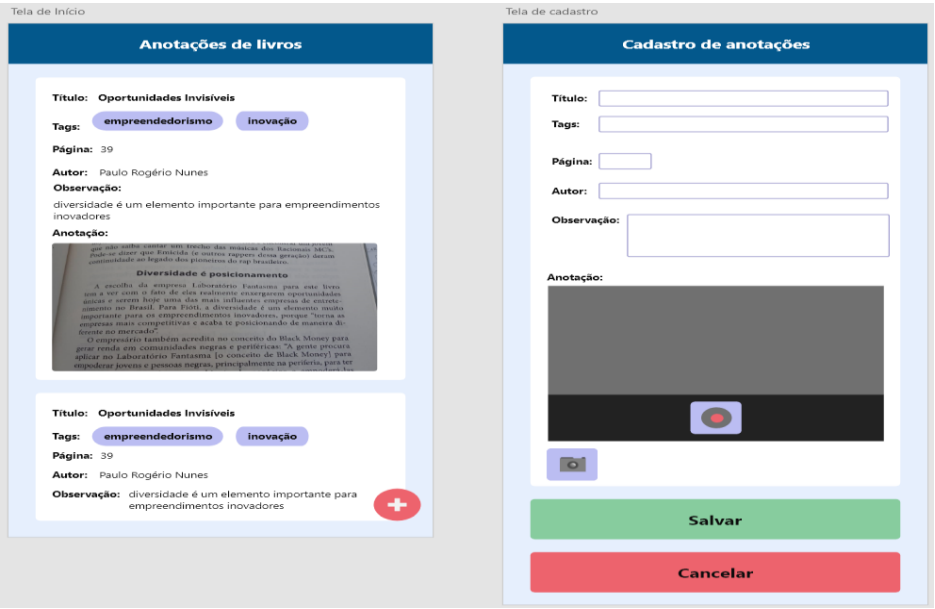
disponíveis por intermédio dos navegadores. Assim, as seções a seguir mostram as etapas deste processo de desenvolvimento.

### 3.1 Construção do Protótipo

A construção do protótipo foi realizada com o suporte da ferramenta Adobe XD (ADOBE, 2021), demonstrando as principais características que uma anotação deveria possuir junto a um design intuitivo para o usuário final.

Conforme mostra a Figura 1, os seguintes atributos foram propostos para cada anotação: título, *tags* (palavras chaves da anotação), página, autor, observação e uma anotação (deve ser sempre imagem). Para o design, foram propostos ícones que apresentam as funcionalidades que podem ser ativadas pelos botões em cores vivas, visando captar a atenção do usuário e proporcionar uma boa experiência de utilização da aplicação.

**Figura 1 – Protótipo do projeto**




**Fonte: próprio autor**

Após o desenvolvimento do protótipo, foi possível dar início ao desenvolvimento da aplicação, descrito na seção subsequente.

### 3.2 Desenvolvimento da aplicação

Para construção do projeto, optou-se pela utilização da ferramenta Visual Studio Code para desenvolver os códigos da aplicação, do ambiente de execução Node.js junto ao Node Package Manager (NPM) para permitir a utilização de bibliotecas terceiras e, por fim, da plataforma Vercel, que possibilitou a hospedagem da SPA final para testes da aplicação com a ferramenta Lighthouse apresentada anteriormente.

O primeiro passo para o desenvolvimento do projeto foi utilizar o comando *npx create-react-app anote --template cra-template-pwa,* no terminal do computador, para criar um ambiente de desenvolvimento, além de excluir os arquivos padrões que vinham com o ambiente e que não seriam utilizados devido ao escopo do projeto. Neste cenário, salienta-se a criação automática do arquivo manifest.json seguido de sua alteração manual com dados referentes ao projeto como imagens do logo, cor do tema, nome, customizações na *splash screen* e ícone da página inicial do dispositivo, entre outros utilizados pelo dispositivo para possibilitar a instalação. Contudo, este arquivo não é lido pelo sistema operacional iOS, onde tem-se a necessidade de utilizar no HTML da aplicação as *tags* "*link*" com o atributo "*rel*" junto ao valor "*apple-touch-icon*" para ícones e "*apple-touch-startup-image*" para a *splash screen*. Ambos devem ser seguidos pelo atributo "*href*" com o caminho da imagem referentes ao projeto.

O segundo passo permitiu a criação do layout da tela inicial e de cadastro propostos no protótipo. Neste contexto, destaca-se o uso da biblioteca React, para desenvolver a interface dinâmica e estática da aplicação e, da biblioteca Pure CSS para estilizar o layout e manter sua responsividade.

No terceiro, ocorreu o aperfeiçoamento da tela de cadastro para o acesso à visualização da câmera pela aplicação, salientando o uso da interface *Web Real-Time Communication* (WebRTC) que permite a aplicação web ter acesso aos dispositivos de mídias do dispositivo móvel, como microfone e câmera.

O quarto passo introduziu as bibliotecas React Router e React Router DOM, que permitiram o uso de apenas um arquivo de HyperText Markup Language (HTML**),** possibilitando que a página não tenha a necessidade de se atualizar ao ser direcionada para outra página conforme a url da aplicação é alterada na barra de navegação.

O quinto passo e o mais importante para a aplicação se tornar progressiva pode ser evidenciado pelo comando descrito na linha 23 da Figura 2, pois é responsável por ativar o

arquivo Service Worker padrão, que vem configurado no projeto inicial, automatizando as configurações referentes ao funcionamento dos serviços executados em segundo plano.

**Figura 2 - Código que utiliza a biblioteca React Router DOM e Service Worker**

```
ReactDOM.render(
  <BrowserRouter>
    <Switch >
      <Route path="/" component={Home} exact/>
      <Route path="/cadastro" component={Cadastro}/>
      <Route path="/teste" component={Teste}/>
    </Switch>
  </BrowserRouter>,
  document.getElementById('root')
);

serviceWorker.register();
```

**Fonte: próprio autor**

O sexto passo foi necessário para criar as ações de cadastro, listagem e exclusão de anotações apresentadas na Figura 3. Para estas ações, foi necessário manipular o armazenamento do navegador de maneira que os dados da aplicação persistam mesmo após o término da sessão e, para isto, foi utilizada a funcionalidade *localStorage* da API WebStorage que permite armazenar os dados no formato chave/valor dentro do navegador.

**Figura 3 – Classe para acessar o *localStorage***

AnotacaoDAO.js M

src > DAO > AnotacaoDAO.js > ...

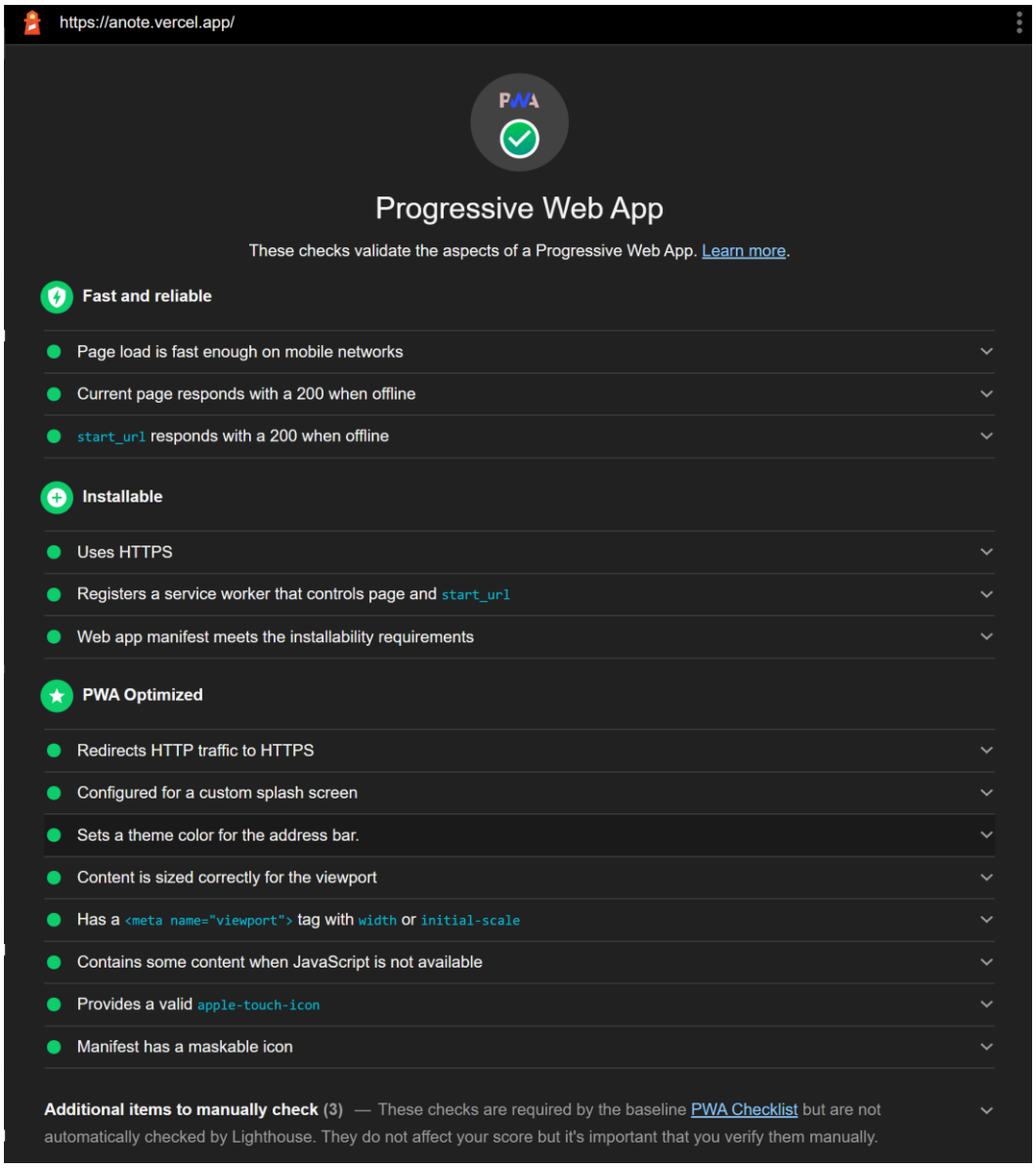
```
class AnotacaoDAO{
  constructor(){
    this.key = 'anotacoes';
  }

  cadastrar(json){
    const data = JSON.stringify(json);
    window.localStorage.setItem(this.key, data);
  }

  listar(){
    const data = window.localStorage.getItem(this.key);
    return JSON.parse(data) || [];
  }

  deletar(id){
    const anotacoes = this.listar();
    let itemPosition;

    for(let i = 0; i < anotacoes.length ; i ++){
      if (anotacoes[i].id === id){
        itemPosition = i;
      }
    };

    anotacoes.splice(itemPosition, 1);
    this.cadastrar(anotacoes);
  }
}

export default AnotacaoDAO;
```

**Fonte: próprio autor**

O sétimo passo do desenvolvimento teve o objetivo de publicar a aplicação na plataforma de hospedagem Vercel, visando usufruir de um protocolo HTTPS que é obrigatório em aplicações progressivas (PONTES, 2018, p. 5).

O oitavo e último passo realizado foi uma auditoria da aplicação por meio da ferramenta Lighthouse com o objetivo de certificar se os requisitos mínimos de uma PWA foram cumpridos. Logo, na Figura 4, pode ser observado que todos os requisitos de verificação automática foram cumpridos, comprovado pelo destaque das características na cor verde.

**Figura 4 – Auditoria do Lighthouse**




**Fonte: próprio autor**

Por fim, com o desenvolvimento, publicação e auditoria na ferramenta Lighthouse concluídos, foi possível partir para a próxima etapa do projeto responsável por avaliar a aplicação.

## 4 RESULTADOS E DISCUSSÃO

Analisando-se a aplicação desenvolvida na seção 3.2, a Tabela 1 abordou algumas características relacionadas aos sistemas operacionais Android versão 8.0.0 e IOS versão 12.4.8.

**Tabela 1 - Características por Sistemas Operacionais (SO)**

| Característica | Android | iOS |
|---|---|---|
| Mensagem sobre instalação | ✔ | ✘ |
| Uso da câmera | ✔ | ✔ |
| Enquanto a PWA estiver ativa ela é reconhecida pela tela de visão geral do dispositivo | ✔ | ✔ |

**Fonte: Próprio Autor**

Ademais, a Tabela 2 mostra as características relacionadas aos navegadores Chrome, Firefox e Samsung Internet para Android e apenas o Safari para IOS. Ressalta-se que no momento da escrita deste artigo a avaliação do Safari no Android não foi possível devido à falta de suporte, bem como o Samsung Internet no IOS. Além disso, no IOS, os navegadores Chrome e Firefox não apresentaram a opção para instalar a PWA.

**Tabela 2 - Características por navegadores**

| | Android | | | iOS |
|---|---|---|---|---|
| **Características** | **Chrome** | **Firefox** | **Samsung Internet** | **Safari** |
| *Splash Screen* | ✔ | ✘ | ✔ | ✔ |
| *Push notification* sobre instalação | ✔ | ✘ | ✘ | ✘ |
| *Push notification* enquanto a PWA está ativa | ✔ | ✔ | ✘ | ✘ |

| | Android | | | iOS |
|---|---|---|---|---|
| **Características** | **Chrome** | **Firefox** | **Samsung Internet** | **Safari** |
| Permissão para acesso da câmera | ✔ | ✔ | ✔ | ✘ |
| Botão descrito para instalar nas opções do navegador | ✔ | ✔ | ✔ | ✔ |
| Ícone do navegador junto ao ícone da PWA na *home screen* | ✘ | ✔ | ✔ | ✘ |
| Funcionamento *offline* | ✔ | ✔ | ✔ | ✔ |
| Comportamento de desinstalação | ✔ | ✘ | ✘ | ✔ |

**Fonte: do próprio autor**

Com base na Tabela 1, a característica “mensagem sobre instalação”, o SO iOS redirecionou para a tela de início, mas não apresentou notificações *toast* na tela do celular, diferentemente do Android que apresentou uma notificação *toast* indicando que a aplicação foi instalada. Ressalta-se que, segundo o Developers Android (2021), notificações *toast* repassam um feedback simples acerca de uma operação em uma janela pop-up que se extingue após alguns segundos.

Referente à característica “push notifications sobre instalação” da Tabela 2, os navegadores Firefox e Samsung Internet para Android não realizaram um comportamento que houvesse semelhança com uma instalação, além disso, o ícone da PWA instalada divide espaço com o ícone do navegador que o instalou. Por outro lado, o navegador Chrome apresentou *push notifications* sobre o andamento e conclusão da instalação e um ícone da PWA foi alocado semelhante ao ícone de uma aplicação. Já no iOS, o navegador Safari adiciona diretamente a aplicação e seu ícone ficou semelhante ao do Chrome no Android.

Com relação à característica “permissão para acesso da câmera” da Tabela 2, nos navegadores do Android todos solicitaram a permissão do usuário para acesso à câmera do dispositivo. Ressalta-se que, na opção de trocar a câmera frontal para a traseira, foi percebida uma lentidão durante a troca no Chrome, enquanto, no Firefox e Samsung Internet a troca foi instantânea. No iOS, o Safari não apresentou a permissão na PWA instalada, logo, não foi possível acessar a câmera para cadastrar uma imagem. Salienta-se que, a fim de verificar mais

a fundo essa característica, também foi realizado o acesso da aplicação direto pelo navegador e, neste contexto, a permissão para o acesso à câmera foi concedida, possibilitando utilizá-la.

A respeito da característica "comportamento de desinstalação" da Tabela 2, no iOS a desinstalação da PWA instalada pelo Safari apresentou uma opção igual ao de uma aplicação nativa. No Android, a PWA instalada pelo Chrome apresentou a opção de desinstalação, porém, para a aplicação instalada pelo Firefox e Samsung Internet não havia essa funcionalidade e para simular o comportamento de desinstalação, a aplicação foi removida da tela inicial.

Com as devidas análises realizadas acerca dos navegadores, a próxima seção discutirá a conclusão do experimento.

## 5 CONSIDERAÇÕES FINAIS

Sabe-se que, grandes empresas da tecnologia vêm auxiliando no aprimoramento da abordagem de PWA com projetos que visam ampliar o acesso a recursos dos dispositivos. Porém, não são todos os navegadores e SOs que possuem a possibilidade de entregar por completo uma experiência de aplicação nativa. Com base nisso, o presente artigo apresentou pesquisas bibliográficas, desenvolvimento e avaliação da aplicação "Anote" em diferentes SOs e navegadores. Como resultado, foi possível vislumbrar o funcionamento de algumas características, devidamente registradas nas tabelas 1 e 2 construídas para esta avaliação.

Foi observado que o Android é o SO mais flexível para PWA, pois o processo de instalação estava presente nos navegadores Chrome, Firefox e Samsung Internet. Ressalta-se que, o navegador que mais se destacou neste SO foi o Chrome, devido à experiência de instalação, ícone na *home screen* e o momento da desinstalação.

Em contrapartida, as observações sobre o iOS geram uma pequena sensação de falta de compatibilidade dos navegadores Chrome e Firefox ao instalarem a PWA, pois apenas o Safari realizou essa funcionalidade completamente, ou seja, demonstrando uma limitação de navegadores que podem utilizar uma aplicação progressiva nessa plataforma.

Assim, mediante as características avaliadas no presente trabalho, observa-se que as características que possuem divergência entre os navegadores no Android são poucas, contemplando, assim, um maior número de requisitos para este SO. No entanto, caso o foco seja dispositivos iOS, a abrangência de requisitos do cliente se torna mais limitada, além da possibilidade de os usuários não utilizarem o Safari, impedindo-os de obterem uma completa experiência no uso de uma PWA.

## REFERÊNCIAS

ADOBE. **Adobe XD**. Adobe, 16 abr. 2021. Disponível em: <https://www.adobe.com/br/products/xd.html>. Acesso em: 16 abr. 2021.

ANDRADE, A. P. **Os principais frameworks CSS**. TreinaWeb blog, 25 out. 2018. Disponível em: < https://www.treinaweb.com.br/blog/os-principais-frameworks-css>. Acesso em: 1 nov. 2020.

DEVELOPERS android. **Visão geral das notificações Toast**. Developers Android, 27 jan. 2021. Disponível em: <https://developer.android.com/guide/topics/ui/notifiers/toasts?hl=pt-br>. Acesso em: 20 fev. 2021.

GOOGLE developers. **Auditar apps da Web com o Lighthouse**. Google Developers, 26 set. 2017. Disponível em: <https://developers.google.com/web/tools/lighthouse>. Acesso em: 1 nov. 2020.

LEPAGE, P.; RICHARD, S. **What are Progressive Web Apps?** Web.dev, 6 jan. 2020. Disponível em: <https://web.dev/what-are-pwas>. Acesso em: 19 jun. 2020.

LIEBEL, C. **Making the web more powerful with project fugu**. International JavaScript Conference, 31 mar. 2020. Disponível em: <https://javascript-conference.com/blog/making-the-web-more-powerful-with-project-fugu>. Acesso em: 31 mai. 2020.

PIZZANI, L.; SILVA, R. C. da; BELLO, S. F.; HAYASHI, M. C. P. I. **A arte da pesquisa bibliográfica na busca do conhecimento**. RDBCI: Revista Digital de Biblioteconomia e Ciência da Informação, Campinas, SP, v. 10, n. 2, p. 53-66, 2012. DOI: 10.20396/rdbci.v10i1.1896. Disponível em: https://periodicos.sbu.unicamp.br/ojs/index.php/rdbci/article/view/1896. Acesso em: 4 nov. 2020.

PNAD contínua. **Acesso à internet e à televisão e posse de telefone móvel celular para uso pessoal 2018**. Rio de Janeiro: IBGE, 2020. 12 p.

PONTES, G. **Progressive Web Apps: Construa aplicações progressivas com React**. Vila Maria São Paulo: Casa do código, 2018. 443 p.

REACT. **Crie um novo React App**. React, 1 nov. 2020a. Disponível em: <https://pt-br.reactjs.org/docs/create-a-new-react-app.html>. Acesso em: 1 nov. 2020.

_____. **React – Uma biblioteca JavaScript para criar interfaces de usuário**. React, 1 nov. 2020b. Disponível em: < https://pt-br.reactjs.org>. Acesso em: 1 nov. 2020.

SOMMERVILLE, I. **Engenharia de Software**. 9.ed. São Paulo: Pearson Prentice Hall, 2011.

STATCOUNTER globalstats. **Mobile Browser Market Share Brasil**. Statcounter GlobalStats, out. 2020. Disponível em: <https://gs.statcounter.com/browser-market-share/mobile/brazil>. Acesso em: 16 nov. 2020.