INTERFACE TECNOLÓGICA

REVISÃO BIBLIOGRÁFICA SOBRE CONCEITO DE *PROGRESSIVE WEB APPLICATIONS (PWA)*

BIBLIOGRAPHICAL REVIEW OF PROGRESSIVE WEB APPLICATIONS (PWA) CONCEPT

Jonathas Kranmer Silva – jonathaskranmer.s@gmail.com

Fernando Tiosso – fernando.tiosso@fatectq.edu.br

Faculdade de Tecnologia de Taquaritinga (Fatec) – Taquaritinga - São Paulo – Brasil

DOI: 10.31510/infa.v17i1.713

RESUMO

Este artigo tem como objetivo proporcionar o entendimento a respeito das *Progressive Web Apps* (*PWAs*), utilizando tecnologias voltadas para o desenvolvimento *Web* devidamente inseridas no contexto do desenvolvimento *mobile* fazendo uso do navegador como motor de execução, tornando a aplicação independente de um sistema operacional, bem como suas evoluções ao longo do seu ciclo de vida, principais características e suas definições, além de algumas de suas limitações apresentadas até o presente momento da escrita deste trabalho. Com o objetivo de mapear os assuntos relacionados às *PWAs*, o estudo baseou-se, principalmente, em revisões bibliográficas a partir de livros, artigos, *website*, bem como notícias oficiais das empresas de *software* voltadas para a plataforma de dispositivos móveis. Pretende-se que após a leitura deste trabalho, o leitor seja capaz decidir sobre utilizar ou não uma *PWA* para atender os requisitos de um determinado projeto, mediante às necessidades estipuladas.

Palavras-chave: *PWA*; Aplicações *Web* Progressivas.

ABSTRACT

This article aims to enable the understanding of Progressive Web Apps (PWAs), using Web development technologies properly inserted in the context of mobile development by using the browser as the execution engine, making the application independent of an operating system, as well as its evolutions along its life cycle, its main characteristics and definitions, as well as some of its existent limitations until the time of the writing of this article. In order to map the issues related to PWAs, the study was mainly based on bibliographical research of books, articles and websites, as well as official news from software companies focused on the mobile device platform. It is intended that after reading this work, the reader will be able to decide whether or not to use a PWA to meet the requirements of a given project, according to the stipulated needs.

Keywords: PWA; Progressive Web Applications.

1 INTRODUÇÃO

A Pesquisa Nacional por Amostra de Domicílios (Pnad), divulgada pelo Instituto Brasileiro de Geografia e Estatística (IBGE em 2014), relatou um crescente número de dispositivos móveis no Brasil/Mundo, com um número de 80% de aparelhos com acesso à internet via *mobile*, ultrapassando os microcomputadores que obtiveram 77% de média.

Altermann (2013), realizou uma pesquisa com mais de 600 marcas presentes nos dispositivos móveis e concluiu que os aplicativos tendem a receber mais atenção do que os sites otimizados para dispositivos móveis dessas marcas. Ressalta-se que os dois fatores principais para avaliar as plataformas foi o tempo de navegação e a frequência de uso.

Entretanto, no contexto do desenvolvimento para dispositivos móveis, a criação de aplicativos para cada uma das plataformas e diferentes versões de sistemas operacionais pode ser desafiadora e pouco produtiva, pois há necessidade de domínios de ambientes totalmente diferentes.

Por outro lado, o desenvolvimento *web* depende, em grande parte, de tecnologias abertas e amplamente conhecidas entre os desenvolvedores, tais como: *HTML*, *CSS* e *JavaScript*. Desta forma, seria muito interessante que tais tecnologias abertas, em algum momento, pudessem ser aplicadas também para o desenvolvimento *mobile*.

Assim, com o objetivo de utilizar as tecnologias voltadas para o desenvolvimento *web* e inseri-las no ambiente de desenvolvimento *mobile* fazendo uso do navegador como motor de execução, buscando a independência do sistema operacional por parte da aplicação, o conceito de *Progressive Web Applications* (*PWA*) foi criado. De acordo com *Developers Google* A (2019), este conceito fornece uma experiência semelhante à de um aplicativo, quando o sistema passa a ser executado em computadores e dispositivos móveis por meio dos navegadores.

Webster (2018), apresenta a história da *PWA* em 2007, junto com o ano do primeiro lançamento do *iPhone*. Em 2007, a *web 2.0* estava em evolução e o padrão *HTML5* já estava em constante desenvolvimento. As páginas eram transformadas em dinâmicas com mapas interativos, alterando a maneira do uso da *web* em dispositivos e *desktop.*

No início da era dos *smartphones*, a visão da *Apple*, responsável pela plataforma iOS, sobre o uso de aplicações *web* ao invés de tecnologias nativas e específicas da plataforma pôde ser observada a partir de afirmação de Jobs (2007): "*Vocês já têm tudo o que precisam se querem saber como desenvolver aplicativos para o iPhone hoje: basta usar os padrões modernos da web*".

Mas, essa visão original logo foi alterada, visto que apenas um ano depois, em 2008, ocorreu o lançamento da loja de aplicativos da *Apple*, a *App Store*, uma grande evolução em direção ao oferecimento de aplicativos nativos para a plataforma *Apple* (2018).

No entanto, Firtman (2018) apresenta que após 11 anos de silêncio, com a chegada do iOS 11.3, lançado dia 30 de março de 2018, a *Apple* adicionou suporte ao conjunto básico de novas tecnologias por trás da ideia dos "*Progressive Web Apps*".

Recentemente, Firtman (2019) afirma que a atualização iOS 12.2 veio com mais recursos para *PWA*, mas ainda algumas limitações, mas, por outro lado, evidenciando incentivo ao uso deste padrão de desenvolvimento de aplicativos.

O presente trabalho foi composto por 5 (cinco) seções sendo elas: a Seção 2 (dois) apresenta os procedimentos metodológicos, a seção 3 (três) apresenta a definição de *PWA* junto com suas características, a seção 4 (quatro) discorre sobre as principais limitações de utilização obtidos e, por fim, a seção 5 (cinco) expõe as considerações finais do artigo.

2 PROCEDIMENTOS METODOLÓGICOS

O presente estudo baseou-se, principalmente, em revisões bibliográficas a partir de livros, artigos, *website*, bem como notícias oficiais das empresas de *software* para dispositivos móveis (*Android*, *Apple* e *Windows Phone*).

Desta forma, as informações coletadas foram analisadas e uma linha do tempo apresentando a evolução do conceito sobre *PWA* foi montada, junto as suas principais características e requisitos de funcionamento, bem como algumas de suas limitações.

3 *PROGRESSIVE WEB APPS (PWA)*

Pontes (2018), explica que a palavra progressiva tem a origem no termo *Progressive Enhancement*, que pode ser entendido mais facilmente pela palavra progressiva. Com base nesse contexto, a ideia de progressiva visa condicionar uma aplicação em atender o maior número de pessoas, plataformas e tecnologias possíveis.

Lima (2019), apresenta a *PWA* como um conceito de desenvolvimento progressivo no qual se utilizam algumas linguagens bases: *HTML*, *CSS* e *JS* como principais, podendo fazer uso de alguns *frameworks* para o auxílio do desenvolvimento da aplicação. Em prática, é um site em modo *mobile*, mas com uma complexidade um pouco maior, visto que existem algumas características específicas para definir a *PWA*

Para melhor praticidade e validação de uma *PWA*, Pontes (2018) e a *Developers Google* A (2019) ressaltam a existência da ferramenta *Lighthouse,* responsável por qualificar uma aplicação *web* em relação aos requisitos de melhoria progressiva, gerando relatórios completos e eliminando grande parte dos testes manuais.

Dentre as principais características de uma *PWA* avaliadas pela ferramenta *Lighthouse* pode-se citar o *service worker*, resposta 200 para uma solicitação *offline*, exibição de conteúdo caso o *JavaScript* do navegador esteja desabilitado, navegação em protocolo seguro (*HTTPS*) e o redirecionamento automático do servidor para o protocolo seguro caso a aplicação seja acessada através do protocolo *HTTP*, funcionamento eficiente mesmo em redes 3G, questionamento sobre a instalação, tela de abertura personalizada (*Splash Screen)*, barra de endereço personalizada, meta tag *viewport*, redimensionamento do conteúdo, *cross-browser*, vicissitude entre páginas e *URL* única.

Visando facilitar o entendimento das principais características supracitadas, as próximas seções abordarão cada uma delas de forma um pouco mais detalhada.

3.1 *Service Worker*

Service Worker é uma especificação da *W3C*, um *script* que o navegador executa em segundo plano, independente da página *web*. A publicação do *Developers Google* B (2019) esclarece que, por meio de *Service Worker* (não exclusivamente), é possível implementar recursos *offline*, como, por exemplo, organizar o cache dos arquivos estáticos da página *web*. Atualmente, de acordo com Pontes (2018), o *script* já inclui recursos como notificações *push* e sincronização de dados em segundo plano.

3.2 *Status* 200

O código *HTTP* 200 é a resposta de *status* de sucesso que indica que a requisição foi um sucesso. De acordo com Pontes (2018), uma das requisições é a simulação de tal *status* mesmo estando *offline*, guardando os dados localmente, ou seja, em cache com a ajuda do *Service Worker*.

3.3 Exibir conteúdo com o *JS* desabilitado

Como a *PWA* tem a característica de um funcionamento perfeito estando *online* ou *offline*, Pontes (2018) apresenta o conceito de ter como boas práticas uma página de erro ou

uma resposta programada caso o *JavaScript* esteja desabilitado no navegador do usuário ou não seja suportado no dispositivo.

Desta forma, faz-se necessário a utilização de uma simples tela informando motivo do não funcionamento do *App*, ou avisando que não é possível usar a aplicação. Ressalta-se que em aplicações *single page* (*SPA*), pode-se fazer uso do elemento *noscript* do *HTML*.

3.4 Utilização do protocolo *HTTPS*

Como toda aplicação necessita de segurança, torna-se obrigatório o uso do protocolo seguro *HTTPS*. Segundo *Developers Google* B (2019), a aplicação progressiva não pode ter seus dados transmitidos sem a utilização do protocolo seguro *HTTPS*, mesmo sendo aplicação na qual os dados são estáticos e não há interação entre o *front-end* e o *back-end*, onde este último é o responsável por prover os arquivos iniciais para serem armazenados localmente no navegador. O perigo dos dados serem interceptados por usuários não confiáveis sempre existe. A esse respeito, Pontes (2018) declarou:

> *"Como os arquivos estáticos ficarão armazenados localmente, a partir do segundo acesso não haverá download de arquivos. Ok, mas e se o primeiro acesso houver um man-in-the-middle (literalmente, um homem no meio da comunicação) provendo arquivos falso para, por exemplo, capturar seus dados e enviá-los para um servidor na nuvem?"*

Mediante esta premissa, a própria especificação do *Service Worker* salienta o uso do protocolo seguro *HTTPS*.

3.5 Todo tráfego de conteúdo *HTTP* deve ser redirecionado para *HTTPS*

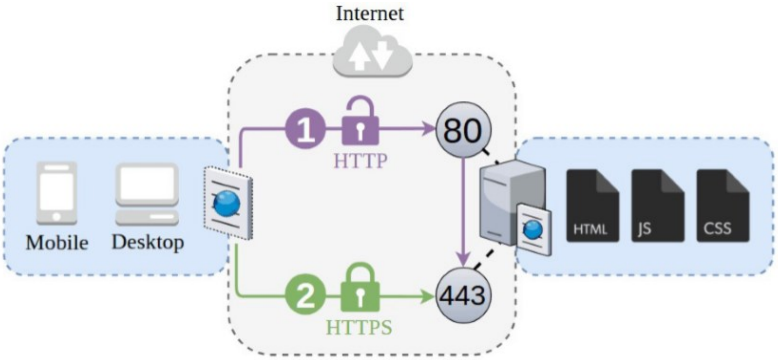
Por padrão, os navegadores do usuário final (sendo *mobile* ou *desktop*) acessam o servidor *web* por meio do protocolo *HTTP*. O acesso a um servidor sem criptografia ocorre quando o navegador recebe o endereço sem explicitar o protocolo.

Pontes (2018), cita que o correto é acessar o servidor *web* pelo protocolo *HTTPS*, ou seja, o protocolo *HTTP* acrescido do protocolo *TLS (Transport Layer Security).* Este protocolo é quem estabelece um canal de comunicação criptografado entre o navegador e o servidor *web*. No entanto, para utilizar este protocolo, há necessidade de digitá-lo no endereço do site, prática comumente esquecida pela maioria dos usuários.

Segundo Pontes (2018), por este motivo é indicado configurar o servidor *web* para sempre redirecionar o tráfego do protocolo *HTTP* (porta 80) para *HTTPS* (porta 443), conforme

apresentado na Figura 1. Desta forma, o navegador é obrigado a estabelecer o canal de comunicação segura.

Figura 1 - Diagrama comparativo entre *HTTP* e *HTTPS*



Fonte: Pontes (2018).

3.6 Carregar rapidamente no 3G

O tempo de resposta de uma aplicação é vital para uma primeira experiência positiva por parte do usuário. *Developers Google* B (2019), cita que toda experiência na *web* deve ser rápida, por meio de uma experiência interativa menor que cinco segundos para obter um conteúdo significativo na tela. Pontes (2018), apresenta uma pesquisa na qual uma aplicação que ultrapassa o tempo de três segundos para apresentar um primeiro resultado, já é considerada lenta por seus usuários. Por fim, *Developers Google* A (2019), reforça esses estudos citando que 53% dos usuários abandonam uma aplicação que exceda três segundos de carregamento. Assim, visando certificar a rapidez de uma *PWA*, estipulou-se que o tempo adequado para seu carregamento a fim dela estar disponível para interação do usuário, seria inferior a dez segundos, considerando acessos por dispositivos móveis por meio das redes 3G.

3.7 Deseja instalar o *App*?

Com o objetivo de prover maior facilidade de acesso para a aplicação, o usuário deve ser questionado se deseja instalá-la em seu dispositivo, assemelhando-se a um aplicativo nativo. As atuais especificações da W3C favorecem muito tal possibilidade, principalmente para usuário *mobile*. Segundo *Developers Google* B (2019), além das configurações de "instalação" estarem muito parecidas com aos aplicativos nativos no dispositivo, deve-se organizar ícones em tamanhos específicos para cada caso, como também, ao finalizar a "instalação", o aplicativo deve estar disponível no mesmo local dos outros aplicativos para ser iniciado. Dessa foram, o ícone, na verdade, será apenas um modo de inicializar um navegador na página específica da *PWA*.

3.8 *Splash Screen* personalizado

De acordo com Pontes (2018), o termo *Splash Screen,* conhecido comumente como tela de abertura, teve origem nos aplicativos *desktop*, onde uma pequena tela de inicialização com a logotipo da aplicação era exibida enquanto o *software* carregava.

No entanto, observou-se que essa característica não era exclusividade dos *desktops*, pois a *splash screen* funcionava simplesmente como uma página intermediária (um *preloading*) enquanto a aplicação era carregada, aspecto evidenciado de forma semelhante nas plataformas *mobile*.

Developers Google A (2019), cita o arquivo chamado "*Web App Manifest",* responsável por registrar o comportamento da inicialização de uma aplicação no navegador, podendo, por meio dele, configurar o *Splash Screen,* como, também, ícones referentes à tela inicial da aplicação. A Figura 2 mostra um exemplo de *Splash Screen mobile.*

Figura 2 - Exemplo de carregamento de uma *PWA* com *Splash Screen* personalizado

Fonte: próprio autor.

3.9 Barra de endereço personalizada

Pontes (2018), cita que mesmo que possa ser ocultado, a barra de endereço do navegador personalizada com as cores do site (principal cor) também é uma das verificações realizadas para certificar uma *PWA*, visando amenizar o impacto visual desta funcionalidade na tela da aplicação para uma melhor experiência de utilização pelo usuário final.

3.10 Meta *tag viewport*

De acordo com o site *Developer Mozilla* (2019), a *tag* <meta> de nome *viewport* (janela virtual) tem a função de melhorar a visualização da página nos dispositivos *mobile*. *Viewport* é

a área visível do usuário em uma página da *web*. Portanto, quanto menor a tela do dispositivo *mobile*, menor será a sua *viewport* (Pontes, 2018). Para um melhor desempenho, indica-se colocar a *tag* <meta> no elemento *head* do código, conforme mostra a Figura 3.

Figura 3 - Meta *tag Viewport*

```
<meta name="viewport" content="width=device-width, initial-scale=
1">
```

Fonte: Pontes (2018).

Analisando a Figura 3, nota-se que existem propriedades a serem preenchidas neste metadado: *name e content.* A primeira faz referência ao nome da meta *tag*. Já, a segunda, na chave *width*, controla o tamanho da *viewport*, ou seja, representa a largura estipulada para a visualização do conteúdo da aplicação no dispositivo. Ressalta-se que o valor *device-width* promove, em pixels de *CSS*, uma escala de 100% para qualquer tamanho de tela do dispositivo. A chave *initial-scale*, controla o nível de dimensão quando a página é carregada pela primeira vez. As propriedades *maximum-scale*, *minimum-scale* e *user-scalable* controlam a permissão para o usuário efetuar o redimensionamento da página.

3.11 Redimensionamento do conteúdo

Existem duas formas de se criar uma aplicação ampla para diferentes variedades de tamanhos de *viewport*: *layouts* estáticos e *layouts* responsivos, sendo este último de maior destaque na área de desenvolvimento para *web*.

A organização estática, no geral, oferece resultados de alta qualidade, mas perde no tempo de desenvolvimento e possui uma baixa manutenibilidade. Segundo Pontes (2018), protocolo HTTP possui várias informações no cabeçalho. Uma delas é a *User-Agent,* responsável por fornecer informações sobre o navegador, do sistema operacional do usuário, entre outras. Por meio dessa propriedade, pode-se descobrir qual dispositivo o usuário está utilizando (celular, *tablet* ou computador) e, com base nestas informações, provocar um direcionamento para uma *URL* específica condizente com a aplicação executada no dispositivo.

Regularmente, encontra-se o uso de dois links para o mesmo site. De acordo com Pontes (2018), nessa situação utiliza um domínio principal para *desktop* e um subdomínio para *mobile*. Com exemplo, pode-se citar o site da Kabum, tendo como servidor endereço mais utilizado o https://www.kabum.com.br e, como mini servidor, o https://m.kabum.com.br na qual é disponibilizada uma versão para *mobile*.

No entanto, a solução mais indicada, citada por Pontes (2018) e pelo site *Developers Google* (2019), consiste no *layout* responsivo. As boas práticas apontam para a construção da página que se auto organiza de acordo com o tamanho da tela do dispositivo, ocultando elementos, substituindo componentes e realizando outras modificações para tornar a aparência e o uso agradável para o usuário. Isso tudo é possível com ao uso de *HTML*, *CSS* (organizando as classes) e o uso de *@media queries*.

Uma *@media queries* é um trecho do *CSS* que permite alterar estilos para cada tamanho que é definido. A Figura 4 mostra o mesmo menu reproduzido em dimensões diferentes, um com cada item de menu detalhado e outro com apenas um ícone de acesso aos itens de menu, mais direcionado para telas de dispositivos menores.

Figura 4 - *@media queries*

Fonte: Materializecss (2019).

3.12 *Cross-Browser*

Para atender uma grande abrangência de usuários, deve-se utilizar códigos compatíveis com vários navegadores. Atualmente, os *frameworks CSS* resolvem esse problema, como, por exemplo o Pure.css, caracterizado pela sua especialização em estilização *mobile* e suas versões distintas para o uso em *browsers* antigos.

3.13 As vicissitudes das páginas

Este requisito preocupa-se justamente em monitorar o período de transição entre uma página e outra. Segundo Ponte (2018), as transições entre páginas não devem ser sensíveis, ou seja, devem transcorrer de forma imperceptível, proporcionando uma agradável experiência para o usuário.

Pontes (2018), refere-se que caso haja uma transição total, a qual é desaconselhável, há a preocupação com o desempenho no tempo da requisição, pois o navegador precisará baixar o

site completo (*HTML, JS, CSS,* imagens) e ainda renderizar o *DOM*, sempre que o usuário acessar uma das páginas do sistema.

Para resolver essa fragilidade, Pontes (2018) orienta evitar as transações completas por meio de construção de *Single-Page Application* (*SPA*), ou aplicação de única página. Nelas, a página é carregada apenas uma vez e as transições fluem sem necessidade de refazer o *download* dos arquivos e renderizar o *DOM*.

3.14 Cada página deve ter uma única *URL*

Pontes (2018), cita o termo técnico *bookmarkable* para as páginas que possuem uma *URL* definida, possibilitando que o usuário adicione aos favoritos do seu navegador. Destaca-se que, além de ser única é importante defini-la com um nome amigável e com lógica para o usuário final.

Para a aplicações não *SPA*, cujas transições são feitas totalmente, esse fator já está previamente configurado, mas ainda há necessidades de definir boas *URLs*. Em uma *SPA*, todavia, a criação de *URL* única para cada páginas se complica, pois os usuários farão a transição entre as *views* (páginas) sem recarregá-las. Como solução, o *React Router,* um *framework* complementar ao *React* se encarrega para organizar as rotas de aplicações.

4 LIMITAÇÕES SOBRE A UTILIZAÇÃO

O conceito *PWA* não é algo novo, mas ainda não está com todo o suporte para a sua devida funcionalidade. A seguir serão abordadas suas vantagens e desvantagens de acordo com plataformas e seus recursos.

Lima (2017), apresenta desvantagens significativas das *Progressive Web Apps* analisando diversas características. Apesar da *Google* e outras empresas estarem com altos investimentos nos *PWAs*, a *Apple* ainda não está, uma vez que existem duas *features* importantes ainda não suportadas pelo Safari: *push notifications* e funcionamento *offline.*

Contudo, Firtman (2018), complementa citando que "*O aplicativo pode armazenar dados off-line e arquivos apenas até 50MB*" para iOS e, em contrapartida, o *Android*, atualmente, pode armazenar mais de 50MB.

O iOS 11.3 beta aplica um regulamento de limpeza de cash para os navegadores, portanto, se o aplicativo não for usado durante 3 semanas, em média, o iOS liberará os arquivos do aplicativo. O ícone ainda estará lá na tela inicial e, quando acessado, o aplicativo será baixado novamente.

De acordo com Firtman (2019), atualmente com o iOS 12.2 a *Apple* disponibilizou os recursos: Geolocalização, Sensores (Magnetômetro, Acelerômetro, Giroscópio), Câmera, Saída de áudio, Síntese de fala (somente com fones de ouvido conectados), *Apple Pay*, *WebAssembly*, *WebRTC*, *WebGL* e outros recursos experimentais.

Os recursos da *PWA* para iOS 12.2 são limitados e não suportam todas as características apresentadas da na seção 3 (três). Firtman (2018), apresenta as principais limitações da aplicação para dispositivos iOS destacando a falta de acesso a alguns recursos, como, *Bluetooth*, *serial*, *Beacons*, *Touch* ID, ID de *Face*, *ARKit*, sensor de altímetro e informações sobre bateria. A *Apple* anunciou, na palestra da *WWDC* (2019), algumas novas melhorias em relação aos *PWAs*, disponível no pacote da atualização iOS 13 beta.

5 CONSIDERAÇÕES FINAIS

Durante o desenvolvimento deste trabalho, foi possível pesquisar e entender o conceito e a tecnologia da *PWA*, desde do início do conceito até a sua implementação nos navegadores e aceitações nos dispositivos móveis, destacando as limitações de cada sistema operacional do dispositivo *mobile*, as evoluções, atualizações nos últimos anos e as futuras atualizações que já foram divulgadas, evidenciando possíveis limitações em alguns cenários em meio a diferença de plataforma.

Foi observado que a disponibilidade total para diversas plataformas está em ascensão, visto que uma das características é o funcionamento em múltiplas plataformas, concluindo-se que não surgirão impactos muito divergentes mesmo que o cenário de execução seja em dispositivos totalmente fora das rotinas de atualização.

Assim, por meio deste artigo, tornou-se possível a compreensão dos pontos positivos e negativos do conceito da *PWA*, permitindo uma decisão de quando é possível sua utilização ou não, de acordo com a necessidade apresentada por um determinado projeto.

Como continuação deste trabalho, propõe-se a possibilidade de desenvolvimento de uma aplicação utilizando a tecnologia *PWA* e sua validação em diferentes plataformas.

REFERÊNCIAS

ALTERMANN, Dennis. **Usuários usam mais aplicativos do que sites *mobile-friendly* de marcas**. [S.I], 11 nov. 2013. Disponível em: https://www.midiatismo.com.br/usuarios-usam-mais-aplicativos-do-que-sites-mobile-friendly-de-marcas. Acesso em: 8 set. 2019.

APPLE. **A *App Store* completa 10 anos**. [S.I], 5 jul. 2019. Disponível em: https://www.apple.com/br/newsroom/2018/07/app-store-turns-10/. Acesso em: 8 set. 2019.

DEVELOPERS GOOGLE A. ***Progressive Web Apps***: Progressive Web Apps are now supported on the desktop! [S. I.], 9 maio 2019. Disponível em: https://developers.google.com/web/progressive-web-apps/. Acesso em: 1 ago. 2019.

DEVELOPERS GOOGLE B. **Seu primeiro *Progressive Web App***. [S.I], 3 jul. 2019. Disponível em: https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/?hl=pt-br. Acesso em: 8 set. 2019.

DEVELOPERS MOZILLA. **Usando a meta tag *viewport* para controlar o layout em navegadores de dispositivos móveis**. [S. I.], 23 mar. 2019. Disponível em: https://developer.mozilla.org/pt-BR/docs/Mozilla/Mobile/Viewport_meta_tag. Acesso em: 2 set. 2019.

FIRTMAN, Maximiliano. ***Progressive Web Apps on iOS are here***. [S. I.], 30 mar. 2018. Disponível em: https://medium.com/@firt/progressive-web-apps-on-ios-are-here-d00430dee3a7. Acesso em: 13 ago. 2019.

FIRTMAN, Maximiliano. ***What?s new on iOS 12.2 for Progressive Web Apps***. [S. I.], 26 mar. 2019. Disponível em: https://medium.com/@firt/whats-new-on-ios-12-2-for-progressive-web-apps-75c348f8e945. Acesso em: 27 ago. 2019.

G1 GLOBO. ***Smartphone* passa PC e vira aparelho nº 1 para acessar internet no Brasil**: Celular está em 80% das casas e PC, em 76,6%; dado do IBGE é de 2014. Pnad mostra que, pela 1ª vez, internet chega a mais de 50% das casas. [S.I], 6 abr. 2016. Disponível em: http://g1.globo.com/tecnologia/noticia/2016/04/smartphone-passa-pc-e-vira-aparelho-n-1-para-acessar-internet-no-brasil.html. Acesso em: 8 set. 2019.

LIMA, Matheus. **Introdução aos *Progressive Web Apps***. [S. I.], 1 fev. 2017. Disponível em: https://medium.com/tableless/introdu%C3%A7%C3%A3o-aos-progressive-web-apps-ad47ba24cddb. Acesso em: 27 jun. 2019.

MATERIALIZECSS. ***Navbar: um exemplo de menu responsivo***. Disponível em: https://materializecss.com/navbar.html. Acesso em: 8 set. 2019.

PONTES, Guilherme. ***Progressive Web Apps**:* Construa aplicações progressivas com React. Vila Maria São Paulo: Casa do Código, 2018. 458 p.

WEBSTER, David. ***Progressive Web Apps - adopt, adapt, or reject?*** [S.I.], 4 out. 2018. Disponível em: https://www.venndigital.co.uk/blog/progressive-web-apps-adopt-adapt-or-reject-93069/. Acesso em: 30 ago. 2019.