UNIVERSIDADE DO VALE DO TAQUARI
CURSO ENGENHARIA DE SOFTWARE

PROGRESSIVE WEB APPS: UM NOVO PARADIGMA DE DESENVOLVIMENTO FRONTEND

Guilherme Maccali

Lajeado, dezembro de 2021

Guilherme Maccali

PROGRESSIVE WEB APPS: UM NOVO PARADIGMA DE DESENVOLVIMENTO FRONTEND

Trabalho de conclusão de curso apresentado no Centro de Ciências Exatas e Tecnológicas, da Universidade do Vale do Taquari – Univates, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Dr. Alexandre Stürmer Wolf

Lajeado, dezembro de 2021

“Culpar programadores e programadoras tem sido a abordagem predominante por meio século de desenvolvimento de Software: Isso ainda não resolveu o problema, então é hora de olhar para direções diferentes.” Boris Baizer

RESUMO

Nos dias atuais, a qualidade de *software* está sendo cada vez mais exigida tanto no desenvolvimento quanto na utilização do usuário final, além da diversidade de dispositivos e plataformas aumentando e a escassez de mão de obra qualificada na tecnologia intensificando. Desse modo, as *Progressive Web Apps* (PWA) têm-se tornado uma opção cada vez mais conveniente na produção de *frontend*, pois têm a capacidade de serem executadas nas mais diversas plataformas, alcançando qualquer dispositivo que tenha um navegador. Pensando nisso, o presente estudo apresentou a criação de um sistema de denúncias e demonstrou os recursos e benefícios existentes no processo de desenvolvimento PWA em três plataformas distintas – Windows, Android e iOS. Esta pesquisa explorou e descreveu os recursos utilizados na aplicação desenvolvida, comparando os resultados entre as plataformas e gerando métricas com a finalidade de testar a viabilidade do uso multiplataforma da abordagem PWA e da *Trusted Web Activities* (TWA) na plataforma Android. Dessa forma, concluiu-se que o uso da PWA e TWA mostrou-se eficiente, pois os recursos funcionaram de forma satisfatória em todas as plataformas testadas, confirmando as comparações realizadas pelos trabalhos relacionados, alcançando, por meio da ferramenta Google Lighthouse, 90% de performance, 97% de acessibilidade, 100% em boas práticas e 100% *Search Engine Optimization* (SEO), atendendo todos critérios verificados, assim como todos os parâmetros requeridos pela PWABuilder.

Palavras-chave: PWA, Progressive Web Apps, TWA, NextJS.

ABSTRACT

Nowadays, *software* quality is being increasingly demanded both in development and in the use of the end user, in addition to the diversity of devices and platforms increasing and the scarcity of qualified labor in the technology intensifying. Thus, *Progressive Web Apps* (PWA) have become a progressively more convenient option in *frontend* production, as they have the ability to run on the most diverse platforms, reaching any device that has a browser. With that in mind, this study presented the creation of a whistleblower system and demonstrated the resources and benefits that exist in the PWA development process on three different platforms – Windows, Android and iOS. This research explored and described the resources used in the developed application, comparing the results between platforms and generating metrics in order to test the feasibility of using the PWA approach and *Trusted Web Activities* (TWA) on the Android platform. Thus, it was concluded that the use of PWA and TWA proved to be efficient, as the resources worked satisfactorily on all tested platforms, confirming the comparisons made by the related works, reaching, through the Google Lighthouse tool, 90% of performance, 97% accessibility, 100% of best practices and 100% of *Search Engine Optimization* (SEO), meeting all verified criteria, as well as all parameters required by PWABuilder.

Keywords: PWA, Progressive Web Apps, TWA, NextJS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
APK	Android Package
APP	Application
CSR	Client-Side Rendering
CSS	Cascading Style Sheets
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
IBGE	Instituto Brasileiro de Geografia e Estatística
IOT	Internet Of Things
ISR	Incremental Static Regeneration
JS	JavaScript
JSON	JavaScript Object Notation
JSX	JavaScript XML
NFC	Near Field Communication
PoC	Prova de Conceito
PWA	Progressive Web Apps
RWD	Responsive Web Design
SEO	Search Engine Optimization

SPA	Single Page Application
SSG	Static Site Generation
SSL	Secure Socket Layer
SSR	Server Side Rendering
SW	Service Worker
TS	Type Script
TWA	Trusted Web Activity
URL	Uniform Resource Locator
UX	User Experience
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

É possível que em algum momento hajam problemas que tendem a levar desenvolvedores a pensar ou até mesmo reclamar que alguma tecnologia usada não sirva ao propósito esperado, Martin (2011). Porém, Lehman (1996) enfatiza a necessidade de resiliência no desenvolvimento de *software*.

Segundo Figueiredo (2007) esta evolução é muito importante para que novos sistemas mais complexos e de qualidade possam ser executados, assim como no livro “Garantia da Qualidade” de *software* de Bartié (2002) ou no livro “Código Limpo” do autor Martin (2011), onde ambos buscam definir táticas, técnicas, e padrões para a melhor entrega possível pensando no futuro.

Muitas das técnicas abordadas mencionadas por Bartié (2002), promovem o aumento na qualidade dos produtos, tornando-se diferenciais competitivos, assim como mencionado por Martin (2011), muitas delas podem ser controversas cabendo a cada equipe decidir quais fazem sentindo em seu contexto.

E para auxílio a estas ideias que, segundo Tandel e Jamadar (2018), em 2015 surge o termo *Progressive Web App* (PWA), criado pela Google, sendo Berriman *designer* e Alex Russel engenheiro do Google Chrome os colaboradores responsáveis.

Segundo Silva e Costa (2018) apud Teixeira (2016), PWA é uma abordagem moderna para criação de aplicativos, que utilizam recursos dos navegadores modernos para tornar a experiência do usuário mais nativa possível, utilizando tecnologias já comuns da *web* como *Cascading Style Sheets* (CSS), *HyperText*

Markup Language (HTML) e *JavaScript* (JS), em conjuntos as *Application Programming Interface* (API)s dos navegadores.

Segundo Souza e Cintra (2018), aplicação *web* progressiva consiste em um modelo de aplicação com a maior abrangência de dispositivos bastando apenas um navegador para funcionar, e não havendo em muitos casos a necessidade da criação de aplicações nativas para plataformas como Android ou iOS por exemplo. Este pensamento concorda com o proposto por Martin (2011), que aponta a importância da não repetição de código, onde segundo Souza e Cintra (2018) pode encarecer o projeto, e segundo Bartié (2002) pode afetar a qualidade final.

Evoluindo neste caminho, existe o termo *Trusted Web Activities* (TWA), que segundo Alemu (2020), é uma abordagem adicional, que dá a possibilidade da adição de uma PWA na Google Play Store de forma simples, de acordo com Garcia (2020) e Alemu (2020), é uma abordagem que outorga mais direitos de acesso ao sistema pela aplicação, direitos antes não possíveis em outras abordagens de encapsulamento como *WebView* adotada anteriormente para páginas *web* serem disponibilizadas via *Application* (APP).

Assim como o TWA dá a oportunidade a uma PWA de ser instalada como um APP nativo, o *Service Worker* (SW) dá a um site comum a oportunidade de ser uma PWA, segundo Neto (2020). Lehman (1996) e Bartié (2002) mencionam a evolução de *software* de formas diferentes, Lehman (1996) de forma resiliente, e Bartié (2002) baseada no aumento da qualidade e satisfação do cliente, sugerindo ferramentas para lidar de forma mais sólida com essa abordagem.

1.1 Problema a ser resolvido e público-alvo

Segundo Oliven (2010), há no Brasil vários aspectos culturais que estão de certa forma ligados a violência no país, desde opressões até movimentos culturais populares.

No contexto pandêmico atual, também há aumentos em casos de violência doméstica contra a mulher não apenas no Brasil, mas também em outros países segundo Viera e Garcia (2020), muitos deles associadas ao isolamento social. O autor também relata um aumento de 18% nas denúncias registradas no período de 1 a 25 de março de 2020.

Por conseguinte, Okabayashi e Tassara (2020), relatam o aumento de casos de feminicídio de mais de 38%, de 2018 a 2019, e de 138%, de 2019 a 2020, comparando o primeiro trimestre de cada ano, Okabayashi e Tassara (2020) também afirmam de que por estar muito próximos do agressor, as vítimas, por constrangimento, não buscam ajuda através de denúncias.

Segundo Platt e Guedert (2020), em Santa Catarina houve diminuição de denúncias voltadas a violência a crianças e adolescentes, porém o autor alerta a sociedade para possíveis casos não relatados, outra vez ligando os pontos com Okabayashi e Tassara (2020), dizendo ter alguma dificuldade para ter o contato com alguma instituição de proteção.

O presente trabalho busca demonstrar o novo paradigma de desenvolvimento denominado PWA e TWA, em uma aplicação que visa facilitar o usuário final em denúncias tanto criminais quanto de cunho de alerta, onde uma pessoa poderá enviar denúncias, afim de alertar as autoridades pertinentes, auxiliando no convívio social geral.

A aplicação desenvolvida poderá ser executada por meio de qualquer dispositivo que tenha um navegador utilizando a PWA ou por meio de um aplicativo Android através da TWA, podendo exibir uma abordagem direta e moderna, contando com ampla compatibilidade entre dispositivos e uso de *hardware*, facilitando assim o registro e atendimento das denúncias.

1.2 Objetivos

Os próximos tópicos trazem à tona o assunto do presente trabalho. O objetivo geral visa apresentar as novas abordagens de desenvolvimento moderno voltadas a PWA, já os objetivos específicos delimitam o que será tratado dentro deste assunto.

1.2.1 Objetivo geral

Utilizar uma PWA, discutindo, exemplificando e aplicando seus conceitos, a fim de desenvolver uma aplicação que oportuniza a seus usuários enviarem denúncias ou alertas de delitos que possam ocorrer na cidade como um todo. De forma rápida e acessível.

1.2.2 Objetivos específicos

- Apropriar novas técnicas de desenvolvimento de *software* voltadas ao *frontend* sendo elas PWA e TWA;
- Aplicar técnicas de otimização em aplicações *web*;
- Aplicar os recursos disponíveis da *web* nas API dos navegadores, para acesso aos recursos nativos pertinentes ao projeto;
- Abordar vantagens e desvantagens dos métodos de desenvolvimento impostos no estudo;
- Desenvolver um aplicativo de denúncias utilizando os conceitos PWA e TWA.

1.3 Estrutura do trabalho

O trabalho em questão é estruturado em 6 capítulos. O primeiro capítulo introduz objetivos da pesquisa, público-alvo e discute sobre a técnica PWA, no segundo capítulo é apresentado o referencial teórico, com o que foi estudado para a aplicação da abordagem de desenvolvimento PWA e as tecnologias utilizadas.

No terceiro capítulo, compara trabalhos que utilizaram a metodologia de desenvolvimento PWA e que possuem recursos e funcionalidades semelhantes de desenvolvimento, mostrando o que abrange ou não cada um deles.

O quarto capítulo detalha, as metodologias na elaboração do presente trabalho, assim como métodos científicos, modo de abordagem, fins da pesquisa e procedimentos técnicos da pesquisa em questão.

O quinto capítulo detalha os passos seguidos para o desenvolvimento dessa aplicação seguindo a abordagem de desenvolvimento PWA, assim mostrando a aplicação com suas telas e funções, e também mostrando problemas que podem ocorrem nesse processo.

No sexto capítulo é apresentada as considerações finais. E por fim são exibidas as referências bibliográficas consultadas para a fundamentação teórica do presente trabalho.

2 REFERENCIAL TEÓRICO

No capítulo anterior foi introduzido o tema a ser estudado, com a introdução, objetivos e público-alvo. Neste capítulo será abordado as ferramentas e conteúdos pertinentes ao tema em questão.

2.1.1 Progressive Web Apps

Ceconi (2018) apud Mozilla (2018), descreve PWA como uma nova filosofia de desenvolvimento, que vai além da *web* tradicional utilizando esta abordagem, também afirma que o principal intuito de uma PWA é “Entregar a melhor experiência possível para o usuário”.

Segundo Tandel e Jamadar (2018), há uma grande crescente no uso dos *smartphones* comparados com a plataforma *desktop*, superando a plataforma *desktop* em números no ano de 2014, indicando a importância do desenvolvimento de aplicações *mobile frendly* ou *mobile first.*

Segundo Costa e Pires (2018), PWA chega com o intuito de entregar o melhor dos dois mundos, referenciando os mundos *mobile* e *web*, visando possuir apenas o que há de melhor em cada um.

O procedimento de construção de uma PWA deve seguir alguns pilares, segundo Ceconi (2018), Biørn-Hansen e Majchrzak (2017) apud Osmani (2015), e são

eles: progressivo, independente de conectividade, *app-like*, sempre atualizado, seguro, descobrível, re-engajável, instalável e linkável.

Nas próximas seções, são apresentados os pontos considerados como base para uma o desenvolvimento de uma PWA.

2.1.2 Progressivo

Segundo Tandel e Jamadar (2018) independente do país que estejam, do *browser* que utilizam, uma PWA funcionará por contar com aprimoramento progressivo.

De acordo Ceconi (2018) e Sheppard (2017) uma PWA visa o desenvolvimento progressivo, ou seja, navegadores mais novos tem mais funções que antigos, até mesmo Chrome, Opera, Safari, Brave e Firefox podem ter diferenças perante as inúmeras funções em suas versões atuais. Essas diferenças podem ser visualizadas acessando o site What Web Can Do Today com o *browser* escolhido.

Na Figura 1 é exemplificado o funcionamento em camadas no desenvolvimento progressivo, onde cada etapa tem como objetivo complementar o sistema, agregando experiência ao usuário.

Figura 1 – Representação do desenvolvimento em camadas

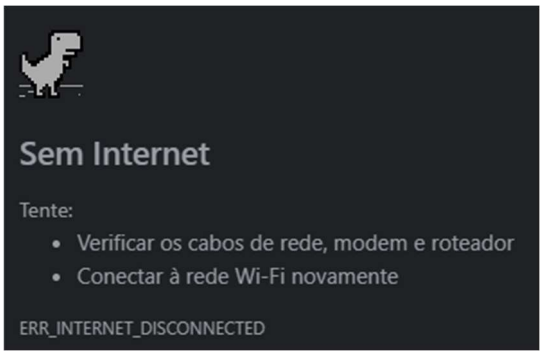
Fonte: Ceconi, 2018.

Ceconi (2018) e Sheppard (2017) concordam que o aprimoramento progressivo dá-se a maneira de implementar o sistema levando em consideração também estas funções, sendo possível aplicar cada parte do sistema de forma gradativa, e oferecendo fluxos alternativos aos usuários caso alguma destas funções estejam faltando.

2.1.3 Offline first

Um dos principais pontos de uma PWA é o funcionamento *offline*, pois segundo Ceconi (2018) a maioria das aplicações *web* necessitam estar conectadas para funcionar, caso contrário, apresentam ao usuário uma tela característica do navegador dizendo que está *offline*, como representado na Figura 2.

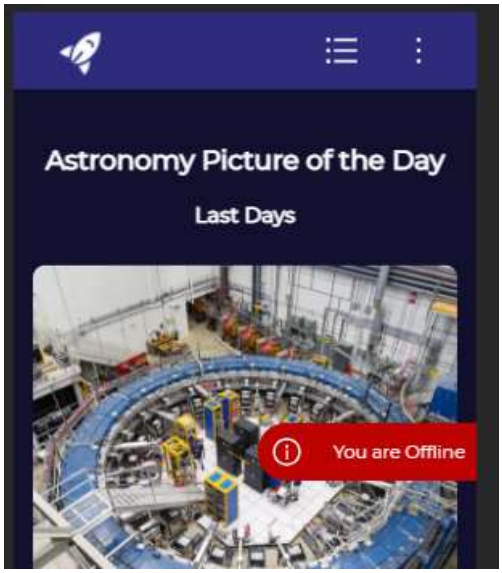
Figura 2 - Exemplo de resposta offline de uma aplicação web padrão



Fonte: Autor, 2021.

A Figura 3 apresenta a resposta de uma PWA quando o usuário está sem *Internet*. Segundo Ater (2017), é correto informar quando o usuário está desconectado, podendo apresentar ou não dados em *cache* na aplicação que foram armazenados na última chamada sem falha na conexão com o APP.

Figura 3 - Exemplo de feedback e resposta com cache de uma PWA



Fonte: Autor, 2021.

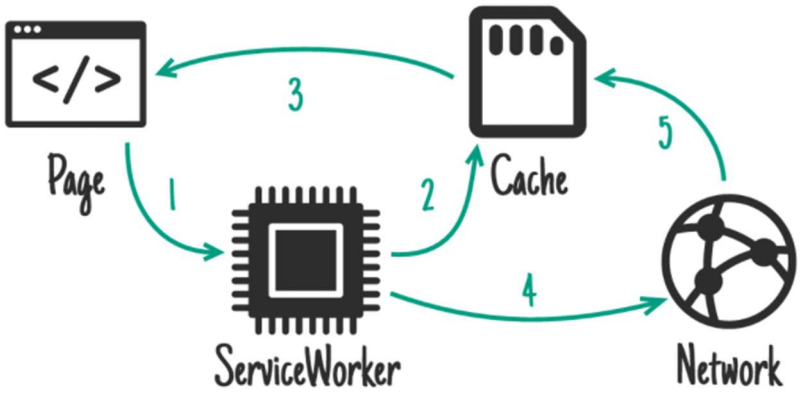
Segundo Ater (2017) é possível tratar o problema de conexão de forma progressiva, informando o usuário de tal falta, além de ser possível a amostragem de dados que se encontram em *cache*.

O SW segundo Ater (2017) e Ceconi (2018) é o que faz a gestão do *cache* a ser armazenado assim que ele é registrado no navegador, esta parte armazenada em *cache* é denominada por Tandel e Jamadar (2018) como *app shell.*

Este *app shell* se comporta como uma casca segundo Tandel e Jamadar (2018), é uma das partes da aplicação que fica guardada em *cache* no navegador, onde sempre estará disponível mesmo que o usuário esteja *offline* segundo Ceconi (2018). Nas Figuras de 4 a 8 pode-se observar algumas destas táticas, e analisar seu fluxo através dos números apontados em cada seta.

Na Figura 4 é representada a técnica denominada *Stale While Revalidate.* Segundo Ceconi (2018), é observado na seta número 1 uma requisição sendo feita, imediatamente na seta número 2 o *Service Worker* busca o que estiver armazenado no *cache* retornando ao cliente na seta número 3, durante este processo na seta número 4 o *Service Worker* busca novos dados do servidor na *Internet*, alimentando o *cache* como representado na seta número 5, disponibilizando os novos dados para a próxima requisição.

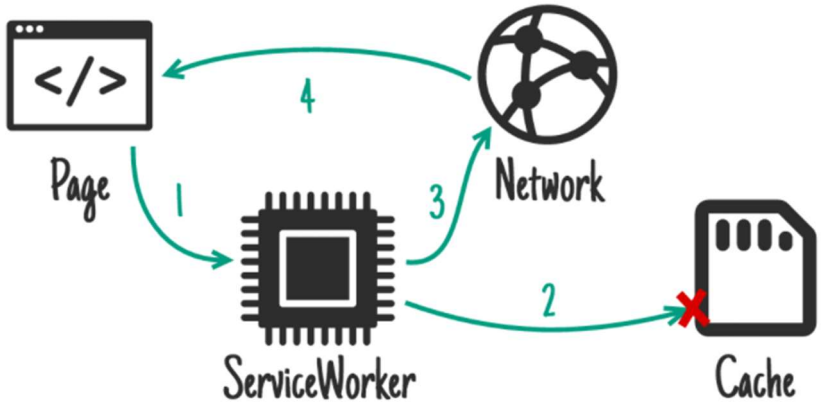
Figura 4 - Representação da estratégia de cache chamada Stale While Revalidate



Fonte: Google Developers WorkBox, 2021.

Na Figura 5 é representado a estratégia de *cache* denominada *Cache First* onde segundo Ceconi (2018), consiste no *Service Worker* buscar sempre em *cache,* somente se não haver resposta em *cache* o SW buscará no servidor.

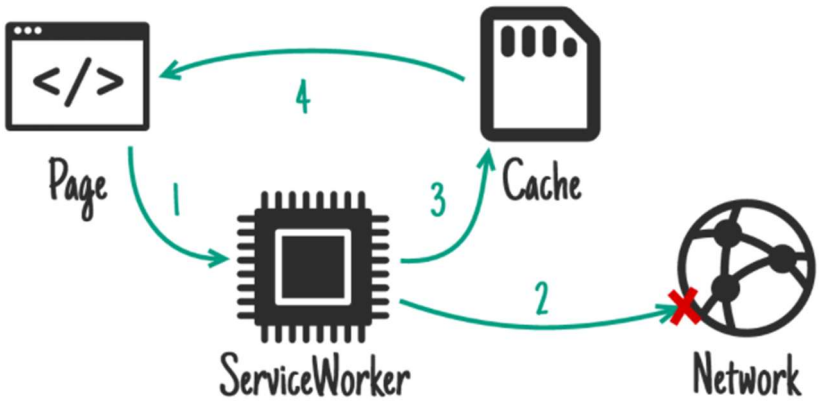
Figura 5 - Representação da estratégia de cache chamada Cache First



Fonte: Google Developers WorkBox, 2021.

Na Figura 6 é representado a estratégia de *cache* denominada *Network First.* Ceconi (2018) expõem que primeiramente os dados são buscados através da rede, e se houver falha, o SW responderá com o que está armazenado em *cache.*

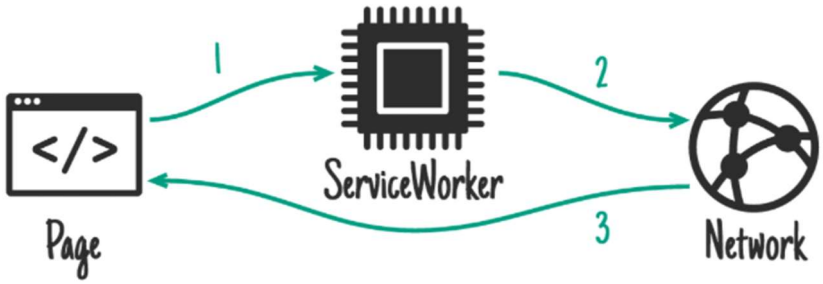
Figura 6 - Representação da estratégia de cache chamada Network First



Fonte: Google Developers WorkBox, 2021.

Na Figura 7, segundo Ceconi (2018) demonstra a forma mais comum de resposta de uma aplicação *web*, denominada como *Network Only,* onde o SW aponta diretamente para a rede não mantendo *cache*.

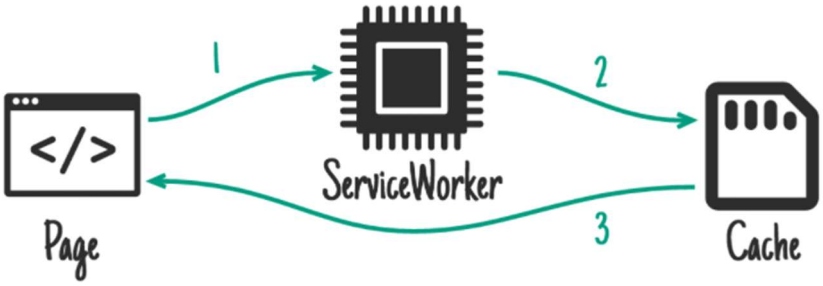
Figura 7 - Representação da estratégia de cache chamada Network Only



Fonte: Google Developers WorkBox, 2021.

Na Figura 8 é apresentado o *Cache Only,* onde o SW responderá somente com o que estiver em *cache* (CECONI, 2018).

Figura 8 - Representação da estratégia de cache chamada Cache Only



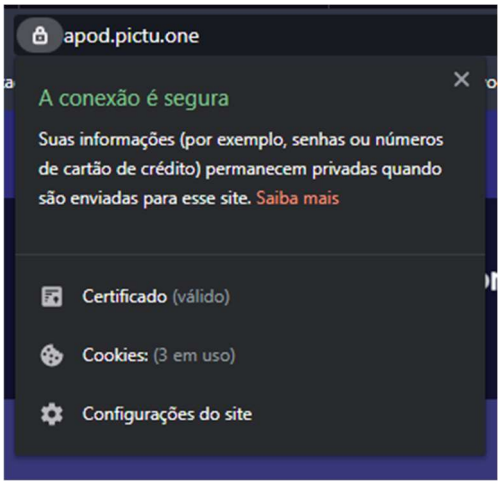
Fonte: Google Developers WorkBox, 2021.

2.1.4 Sempre atualizado e seguro

Segundo Costa e Pires (2018) e Yoseph (2020), para que uma PWA possa ter acesso as funções nativas dos dispositivos que a acessão um dos pré-requisitos é ter o certificado *Secure Socket Layer* (SSL) ativo (Figura 9), sendo acessado via *Hyper Text Transfer Protocol Secure* (HTTPS).

Segundo Tandel e Jamadar (2018) e Ceconi (2018), uma PWA é atualizada no momento que o usuário a acessar, estando sempre atualizada, nunca necessitando a intervenção do usuário para a atualização.

Figura 9 - Representação de certificado SSL ativo em uma página web



Fonte: Autor, 2021.

2.1.5 App-like

Segundo França (2015) descreve que o *Responsive Web Desgin* (RWD) como indispensável no desenvolvimento *frontend* moderno, dado o número de dispositivos e tamanhos de tela que tem-se atualmente.

Segundo Ceconi (2018) o conceito *mobile first* consiste em dar prioridade no desenvolvimento de uma tela para dispositivos móveis, expandindo posteriormente

para dispositivos com telas maiores, *mobile first* é outra premissa da *web* progressiva citada na seção 2.1.1 e seção 2.1.2.

Outro ponto importante é que as tecnologias *web* estão se igualando em termos de recursos, comparados com tecnologias nativas segundo Eriksson (2018), tudo isso libera a possibilidade de criação de aplicações que antes não seriam possíveis por limitação de comunicação com o sistema.

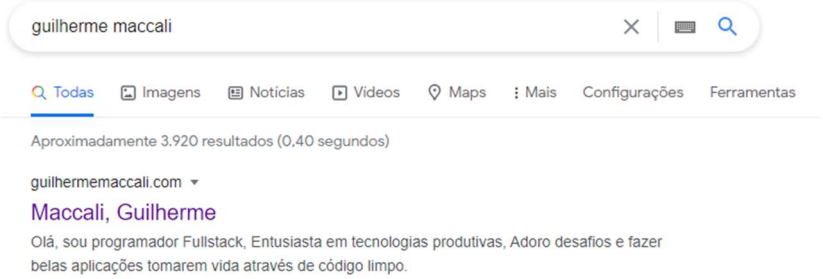
Além da evolução do armazenamento em *cache*, tem-se a possibilidade do uso de recursos como câmera, geolocalização, banco de dados interno do navegador como o *IndexedDB*, para o auxílio do usuário e um melhor funcionamento da aplicação, trazendo esse aspecto de app nativo ao sistema.

2.1.6 Compartilhável, descobrível e envolvente

Search Engine Optimization (SEO), segundo Imperatriz e Krzyzanowski (2018), nada mais é do que um conjunto de técnicas aplicadas para a otimização da aplicação, a fim de melhorar o seu ranking nos resultados de busca, que vão de técnicas de organização de conteúdo até métricas de monitoramento, utilizando ferramentas para sua melhor execução, o que se bem executada, torna a aplicação mais descobrível.

A Figura 10 representa um resultado de busca utilizando as palavras “guilherme” e “maccali” onde o resultado compõem-se do *title* e *description* do *site* procurado.

Figura 10 - Exemplo do resultado de SEO

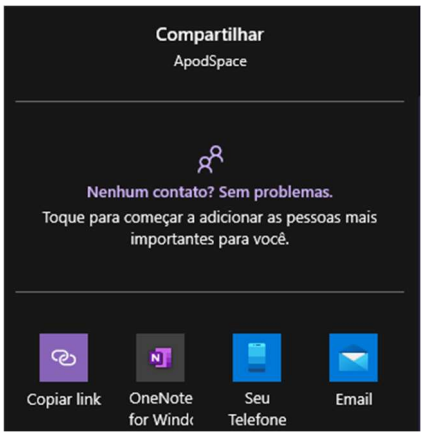


Fonte: do Autor.

Assim como uma *web app* normal, uma PWA também é de fácil compartilhamento, segundo Souza e Cintra (2018) e Ceconi (2018), com apenas um *link* é possível fazer o compartilhamento para outras pessoas sem a necessidade de algo mais complexo, tornando-a, compartilhável e envolvente.

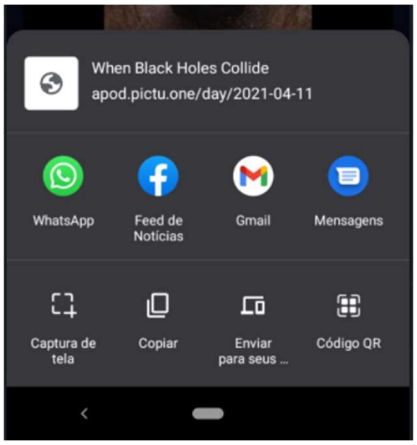
Nas Figuras 11 e 12 tem-se o comportamento da *Share API* ao ser chamada tanto para as plataformas *desktop* quanto *mobile.*

Figura 11 - Exemplo da Share API presente em navegadores desktop



Fonte: do Autor.

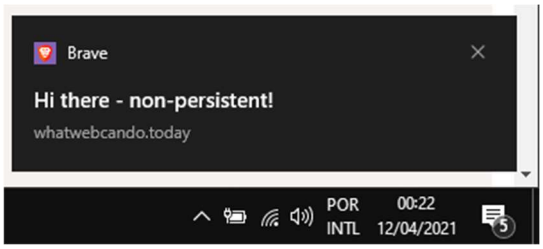
Figura 12 - Exemplo da Share API presente em navegadores mobile



Fonte: do Autor.

Outra possibilidade de uma PWA, é poder contar com *push notifications* que segundo De Souza e Cintra (2018) facilitam o reengajamento do seu público como um todo, tornando ainda mais envolvente. A Figura 13 representa uma notificação sendo recebida no sistema Windows através do navegador Brave.

Figura 13 - Exemplo de Push Notification na plataforma Windows



Fonte: do Autor.

Segundo Ceconi (2018) apud Gaunt (2018), as notificações podem vir da própria aplicação em sua execução ou a partir de um serviço externo como o *One Signal* utilizada pela autora.

2.1.7 Instalável e padronizável

O *Web App Manifest* segundo Ceconi (2018), é um arquivo no formato *Javascript Object Notation* (JSON), que deve ser servido de forma pública a quem acessa a aplicação, nele é possível fazer diversas configurações para uma PWA. É possível definir, por exemplo, os nomes curtos e longos da aplicação, descrição, tipo de exibição, orientação de exibição, cores temáticas, *base url*, ícones, *splash screen* e menu de acesso direto da aplicação, como indicado na Figura 14.

É por meio deste do arquivo de manifesto que segundo Tandel e Jamadar (2018) e Ceconi (2018), é possível passar as informações básicas para que uma PWA possa vir a ser instalada em um dispositivo.

Figura 14 - Exemplo de arquivo de manifesto

```
{
    "name": "Apod",
    "short_name": "Apod",
    "description": "Amazing Images from Nasa day by day",
    "display": "standalone",
    "orientation": "portrait",
    "theme_color": "#2c2c7b",
    "background_color": "#2c2c7b",
    "start_url": "/",
    "icons": [
      {
        "src": "/icons/icon48.png",
        "sizes": "48x48",
        "type": "image/png",
        "purpose": "any maskable"
      },
      {
        "src": "/icons/icon96.png",
        "sizes": "96x96",
        "type": "image/png",
        "purpose": "any maskable"
      },
      {
        "src": "/icons/icon192.png",
        "sizes": "192x192",
        "type": "image/png",
        "purpose": "any maskable"
      },
      {
        "src": "/icons/icon512.png",
        "sizes": "512x512",
        "type": "image/png",
        "purpose": "any maskable"
      }
    ]
}
```

Fonte: do Autor.

Segundo Tandel e Jamadar (2018) o manifesto pretende tornar ainda mais fácil o descobrimento da aplicação na *Internet*, reforçando a importância da *web* moderna, e dando ênfase aos motores de busca.

2.2 Trusted Web Activities

Segundo Alemu (2020), *Trusted Web Activities* (TWA) é uma maneira de introduzir uma PWA na loja de aplicativos do Google (Google Play), foi introduzida em 2019, e é a maneira considerada oficial para fazer esta ação.

Segundo Alemu (2020) essa abordagem deixa a experiencia ainda mais nativa, pois o usuário pode instalar através da loja como um APP convencional. Alemu (2020) destaca o tamanho final de uma *Android Package* (APK) gerado por esse meio, que no exemplo do autor teve um tamanho final de 2 MB, liberando também ações no

sistema onde não seriam possíveis em uma execução na técnica de *web view* já praticada anteriormente.

Alemu (2020) apud Kirupa (2019), mostram que a técnica de *web view* é o ato de embutir um navegador (normalmente o padrão do sistema) em um APP nativo dando impressão ao usuário de estar utilizando o APP nativo.

2.3 Ferramentas

Ferramentas de *software* tornam-se importantes para o *debug* da aplicação, sendo possível aprofundar-se nas funcionalidades da sua aplicação como um todo facilitando o processo de construção da aplicação segundo Trindade (2020).

Segundo De Souza e Cintra (2018) as ferramentas para o desenvolvimento de uma PWA não se diferem das utilizadas em uma aplicação *web* padrão, mas com essa abordagem atenua-se os benefícios ao usuário final, a Figura 15 expõem as principais tecnologias utilizadas no presente estudo.

Figura 15 – Tecnologias utilizadas no desenvolvimento da aplicação

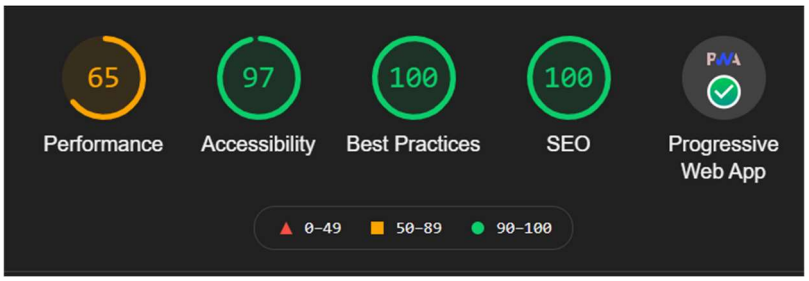
Fonte: do Autor.

2.3.1 Lighthouse

A ferramenta LightHouse desenvolvida pela Google tem o intuito de fazer a medição e verificar pontos base para melhoras em uma aplicação *web*, servindo de base para a construção de uma PWA segundo De Souza e Cintra (2018).

A ferramenta pode ser executada através da extensão disponível para Chrome, ou pelas ferramentas de desenvolvedor do navegador segundo Ceconi (2018), onde a ferramenta gera métricas e verifica 5 pontos da aplicação em seu relatório, sendo eles: performance, acessibilidade, boas práticas, SEO, e os parâmetros necessários para a aplicação ser considerado uma PWA.

Figura 16 - Exemplo das principais métricas obtidas com a ferramenta Lighthouse



Fonte: do Autor

Ceconi (2018) expõem também, que estes itens possuem *checklists* onde unindo-os tornam-se as métricas finais apontadas da Figura 16. Algumas delas serão consideradas no presente projeto, sendo abordadas de forma mais específica a última métrica.

2.3.2 ReactJS

ReactJS é segundo Hoang (2020) uma biblioteca JS para construção de interfaces ao usuário de forma componetizável, que segundo Vu e An (2020), teve seu início em 2013 pelo Facebook.

A biblioteca foi criada com o intuito de resolver problemas com projetos e interfaces complexas, independente do serviço que a provem dados, trabalhando completamente como como *Client-side Rendering* (CSR), onde a interface é montada no lado do cliente, ou seja, no navegador.

Segundo Lins (2019) com ReactJS é possível criar componentes reutilizáveis e testáveis, de forma a se alinhar com Martin (2011) que salienta sempre a importância do reaproveitamento de código, e o uso de testes.

2.3.3 JSX

O *JavaScript XML* (JSX) segundo Vu e An (2020) e Lins (2019), é uma extensão do JS onde sua escrita assemelha-se ao HTML ou ao *Extensible Markup Language* (XML), porém também tem a capacidade de ser interpretado como código JS posteriormente virando HTML.

Lins (2019) afirma que não é necessário o uso do JSX para utilizar o ReactJS, porém para projetos iniciados com ele essa é a maneira escolhida na maioria das vezes, também por meio deste que é possível criar componentes customizados e utilizá-los como *tags* HTML.

2.3.4 NextJS

Segundo Jiménez (2021) e Francisco (2020), o NextJS é um *framework* para desenvolvimento de aplicações *web* baseado em ReactJS, e com ele é possível renderizar páginas de diversas maneiras.

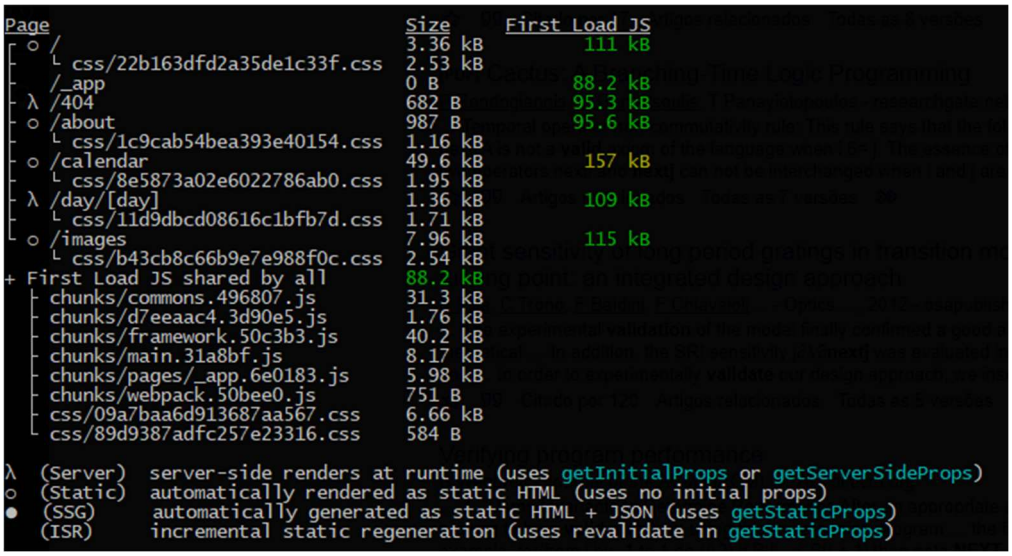
Além de ser usado com o JS, o *framework* também dá suporte ao *TypeScript* (TS), assim como detecção do idioma do usuário e outros benefícios como internacionalização segundo Jiménez (2021).

Jiménez (2021) menciona várias funcionalidades importantes do *framewok* NextJS, sendo elas:

- Otimização de imagens: Utilizando um componente do *framework* é possível, otimizar imagens, para um carregamento mais rápido posteriormente;
- *Server Side Rendering* (SSR): Renderiza a página do lado do servidor para a posterior entrega ao usuário;
- *Static Site Generation* (SSG): Gera página de forma estática;
- *Increment Static Generation* (ISR): É possível gerar páginas estáticas no momento do seu primeiro acesso;
- Rotas para API: É possível criar rotas API para consumo próprio ou de outros serviços.

Na Figura 17 pode-se ver alguns destes tipos de páginas sendo geradas através do *framework*, com sua respectiva legenda mostrando de que forma cada uma foi gerada.

Figura 17 - Exemplo de geração de páginas com o NextJS



Fonte: do Autor.

Segundo Hoang (2020) utilizar SSR é melhor para o SEO do que uma página ReactJS comum que é uma *Single Page Aplication* (SPA).

2.3.5 Backend

Segundo Garcia Silva (2017), uma aplicação *backend* tem a função de servir e processar dados para outras aplicações, no presente estudo, a PWA desenvolvida que consumirá os dados.

Apesar de Lins (2019) colocar o AdonisJS como um *framework* de *backend* mais robusto de outros do JS tendo uma série de funcionalidades configuradas para a facilitação de seu uso, e uma estrutura inspirada em outros *frameworks* como Laravel, o presente projeto utilizará o NextJS mencionado na seção 2.3.4 para melhor compartilhamento de código.

2.3.6 Jest

Segundo Moroz (2019) e Trinh (2020), o Jest é um *framework* para execução de testes automatizados e simulação de dados para estes testes, criado e mantido pelo Facebook assim como o ReactJS segundo Vu e An (2020).

Por conseguinte, é considerado o mais popular dentre os *frameworks* para testes unitários em JS segundo Trinh (2020). Em pesquisa feita por Moroz (2019), Jest foi considerada a ferramenta mais utilizada dentre as bibliotecas de testes da linguagem com 96,5% de seus usuários apontando extrema satisfação em seu uso, pesquisa feita em 2018, com participação de 20.000 desenvolvedores.

Esta ferramenta se torna muito importante para o viés do *clean code* pois Martin (2011) aponta que não há código limpo sem testes.

Neste capítulo foram detalhados os principais conceitos e tecnologias que o presente estudo utiliza como base para o desenvolvimento da aplicação PWA. No próximo capítulo serão apresentados cinco trabalhos que seguem a temática principal destes estudo a fim de realizar uma comparação entre eles.

3 TRABALHOS RELACIONADOS

No capítulo anterior foi realizado um detalhamento dos principais conceitos e ferramentas que são utilizadas como base para o presente estudo. No presente capítulo são detalhados trabalhos semelhantes a este.

O objetivo é comparar as diferentes abordagens, para que desta forma seja estabelecida diferenças e pontos em comum, estimar resultados, verificar abordagens e o melhor método para execução do presente estudo.

3.1 Trabalho 1

No trabalho desenvolvido por Ceconi (2018), denominado “Experiência do Usuário em *Progressive Web Apps*”, teve como principal questão de pesquisa a forma que uma PWA pode aperfeiçoar a experiência de usuário nas múltiplas plataformas da atualidade.

O estudo de Ceconi (2018) também tem como objetivo desenvolver uma aplicação voltada a área de turismo utilizando as metodologias de desenvolvimento de uma PWA como, recursos de *cache* para o funcionamento *offline*, e geolocalização.

Este estudo compõe-se em quatro etapas, primeiramente a autora buscou referências, nas áreas de PWA e *User Experience* (UX), logo após na segunda etapa foram avaliadas as informações coletadas, voltando para o objetivo da aplicação. Na

terceira etapa foi executado o desenvolvimento da aplicação utilizando a técnica PWA, após isso, na quarta etapa é feita a avaliação do projeto.

Ceconi (2018), aponta no Quadro 1, os principais pontos de comparação entre as principais técnicas de desenvolvimento de aplicações *frontend* abordados em seu estudo.

Quadro 1 - Comparação entre as tecnologias web, PWA, híbrida e nativa

Funcionalidade	***Web***	**PWA**	**Híbrido**	**Nativo**
Responsividade	**Sim**	**Sim**	**Sim**	**Sim**
Funciona *offline*	Não	**Sim**	**Sim**	**Sim**
Atualizações rápidas	**Sim**	**Sim**	Não	Não
Indexáveis pelos motores de busca	**Sim**	**Sim**	Não	Não
Notificações	Não	**Sim**	**Sim**	**Sim**
Instalável	Não	**Sim**	**Sim**	**Sim**
Multiplataforma	**Sim**	**Sim**	**Sim**	Não
Presença em lojas virtuais	Não	Não	**Sim**	**Sim**
Acesso ao *hardware*	Não	Parcial	**Sim**	**Sim**

Fonte: Ceconi (2018).

Além desta comparação entre as plataformas citadas, o estudo compara, dois *frameworks*, conhecidos para desenvolvimento de PWAs sendo eles o Ionic e o Polymer, onde ouve o desenvolvimento de duas aplicações, semelhantes utilizando uma base de dados sobre filmes.

Após este desenvolvimento, a autora do estudo utiliza a ferramenta Lighthouse do Google para rodar testes com objetivo de gerar métricas comparativas entre as duas aplicações desenvolvidas.

Os pontos comparados no Quadro 1 vão de encontro aos exemplificados no Quadro 2, e são utilizadas para comparar as aplicações em Ionic e Polymer desenvolvidas por Caconi (2018).

Quadro 2 - Comparação entre os frameworks escolhidos pelo autor do estudo

Métricas	Ionic		Polymer	
	Desktop	*Mobile*	*Desktop*	*Mobile*
Performance	97	48	100	72
PWA	82	91	82	91
Acessibilidade	80	80	100	100
Melhores práticas	100	100	94	94
SEO	89	89	100	100

Fonte: Ceconi (2018).

Ceconi (2018), aborda sobre as técnicas *cache* para aplicação PWA, comparando quais delas seriam mais vantajosas para o usuário, visando a melhor experiência possível, que é um dos objetivos do estudo.

A autora utiliza de uma técnica de 5 camadas, sendo elas: estratégia, escopo, estrutura, esqueleto e superfície para delimitar aspectos da aplicação no que tange o uso e experiência do usuário final, começando pelas questões gerais do negócio, fluxo, requisitos, interação do usuário, navegação, ligando tudo isso a UX com PWA.

A análise final da aplicação contemplou outros aspectos de SEO que se tornam algo a mais no trabalho, todos eles foram executados como o proposto pela autora, apenas foi trocado o *framework* do Polymer para Ionic para a execução do projeto.

No que tange UX, foram trabalhados elementos de interação como, *spinners* e *loaders*¸ animações, carregamentos incrementais das páginas, para redução do *first-load* assim como também o *cache* de algumas requisições, houve o teste com a aplicação Lighthouse com pontuações altas, marcadas em verde pela plataforma, tendo média amarela em acessibilidade.

Com Ceconi (2018), foi possível perceber a importância, e a capacidade tanto de uma PWA quanto do UX, o que alcança uma visão mais profunda para a construção do trabalho proposto no presente estudo.

3.2 Trabalho 2

O estudo de Nedel (2020), denominado “Sistema integrado para atendimento em restaurantes” teve como objetivo desenvolver uma aplicação de cardápio digital, que permita simplificar o atendimento em restaurantes, coletar dados para gestão e para avaliação do sistema. Foi utilizado PWA em toda a aplicação elaborada em forma de uma SPA, tanto para o gestor, para a cozinha e para o cliente do estabelecimento.

O autor explica a importância da digitalização no processo em questão, não apenas para a melhor organização geral dentro do estabelecimento, mas também para a concorrência de mercado e dinamismo da realidade atual.

Nedel (2020) no Quadro 3, compara 3 sistemas, 2 nacionais e 1 estrangeiro, semelhantes, comparando-os ao que foi desenvolvido no estudo proposto, apontando problemas que ocorreram em alguns deles.

Quadro 3 - Comparação entre os sistemas analisados por Nedel

	Goomer	Menu Digital	MENU	Solução proposta
Gestao de cardapio	✔	✔	✔	✔
Utilizado por tablets em cada mesa	✘	✘	✘	✘
Totens de auto atendimento	✔	✘	✘	✘
Site para auto atendimento de clientes	✘	✔	✘	✔
Aplicativo para auto atendimento de clientes	✘	✘	✔	✔
Busca de restaurantes	✘	✘	✔	✘

Fonte: Nedel (2020).

O autor retratou o cenário das principais plataformas *mobile* da atualidade, Android e iOS, em uma visão econômica e de público aos quais utilizam cada plataforma, menciona também alguns sistemas menos conhecidos.

Nedel (2020) em seu estudo explicou o funcionamento de aplicações nativas, e também híbridas, porém, dando maior ênfase ao React Native, Ionic e PWA, onde explicou a base de cada um deles.

Quadro 4 - Comparação de aplicações nativas, webs e PWAs

	Aplicativo Nativo	Website Responsivo	PWA
Funcionamento offline	✔	✘	✔
Notificações push	✔	✘	✔
Instalável	✔	✘	✔
Funcionamento em full screen	✔	✘	✔
Indexável por mecanismos de busca	✘	✔	✔
Funciona em plataformas diferentes	✘	✔	✔
Download não necessário	✘	✔	✔
Updates não necessários	✘	✔	✔

Fonte: Nedel (2020).

No Quadro 4 o autor proporcionou uma comparação entre uma aplicação nativa, uma aplicação *web* padrão e uma PWA, onde nota-se alguns pontos relevantes sobre as tecnologias comparadas.

Nedel (2020), montou uma relação das etapas a serem desenvolvidas para solução do problema proposto, sendo elas: levantamento de informações, estudo das tecnologias a serem aplicadas, levantamento de requisitos, modelagem do sistema, prototipação, testes e avaliação.

A autor utilizou uma abordagem voltada a engenharia de *software*, mostrando os atores da aplicação no caso cliente, gestor e cozinheiro, apresentando também, os requisitos para cada ator envolvido.

Para o sistema em questão foi elaborado o diagrama de casos de uso, que de acordo com o autor ajuda na visualização dos processos envolvidos no projeto, e também para os atores ouve o detalhamento de cada caso de uso, com descrição, fluxo e requisitos associados.

Posteriormente, no estudo foi realizada a prototipação, mostrando as telas e fluxos de cada um dos atores do sistema, assim como o dispositivo alvo deles, delimitando a atuação do sistema de acordo com as telas da aplicação, voltada para cada ator envolvido.

No desenvolvimento de seu estudo, Nedel (2020), mostrou as ferramentas utilizadas na aplicação, sendo elas: JS, TS, Firebase, Preact, Webpack, MeterialUI e Workbox. O autor retratou a importância e vantagem do uso do TS, para projetos de alta complexidade.

No estudo de Nedel (2020) o foco da PWA desenvolvida é propriamente a aplicação do cliente, no desenvolvimento é apresentado a PWA sendo instalada e as telas desenvolvidas.

Nedel (2020) realizou uma pesquisa para avaliar e validar o sistema, para levantar possíveis problemas. As questões foram as seguintes:

- Foi possível executar todas as funcionalidades listadas? Se não qual delas não foi possível?
- O quão fácil e intuitivo é a interface?
- O quão aceitável é o tempo de resposta das funcionalidades?
- O quão útil você considera esse APP/serviço?
- O quão bem-sucedido é o APP em relação ao que ele se propõe?
- O quão agradável é a interface?

Nedel (2020), com o resultado das perguntas acima, obteve uma aceitação considerável da aplicação desenvolvida no estudo, sendo assim a aplicação proposta obteve êxito na maioria dos pontos principais avaliados na pesquisa.

O autor conclui este estudo, mostrando o conhecimento adquirido, tendo mencionado anteriormente não ter conhecimento prévio em PWA e outras tecnologias abordadas, considerou também que PWA pode ser uma abordagem mais economicamente viável para determinados negócios.

3.3 Trabalho 3

Em seu estudo intitulado “O *Progressive Web App* – PWA – como ferramenta para a produção audiovisual”, Trindade (2020) entregou um estudo exploratório sobre PWA, como foco em como uma aplicação PWA pode contribuir para uma aplicação de produção áudio visual através de uma prova de conceito (PoC).

O autor buscou pontos importantes no desenvolvimento de uma aplicação *web* moderna, como *mobile first*, *design* responsivo, computação pervasiva e computação ubíqua, mostrou dados do crescimento da internet no Brasil, assim como de uso dos *smartphones*.

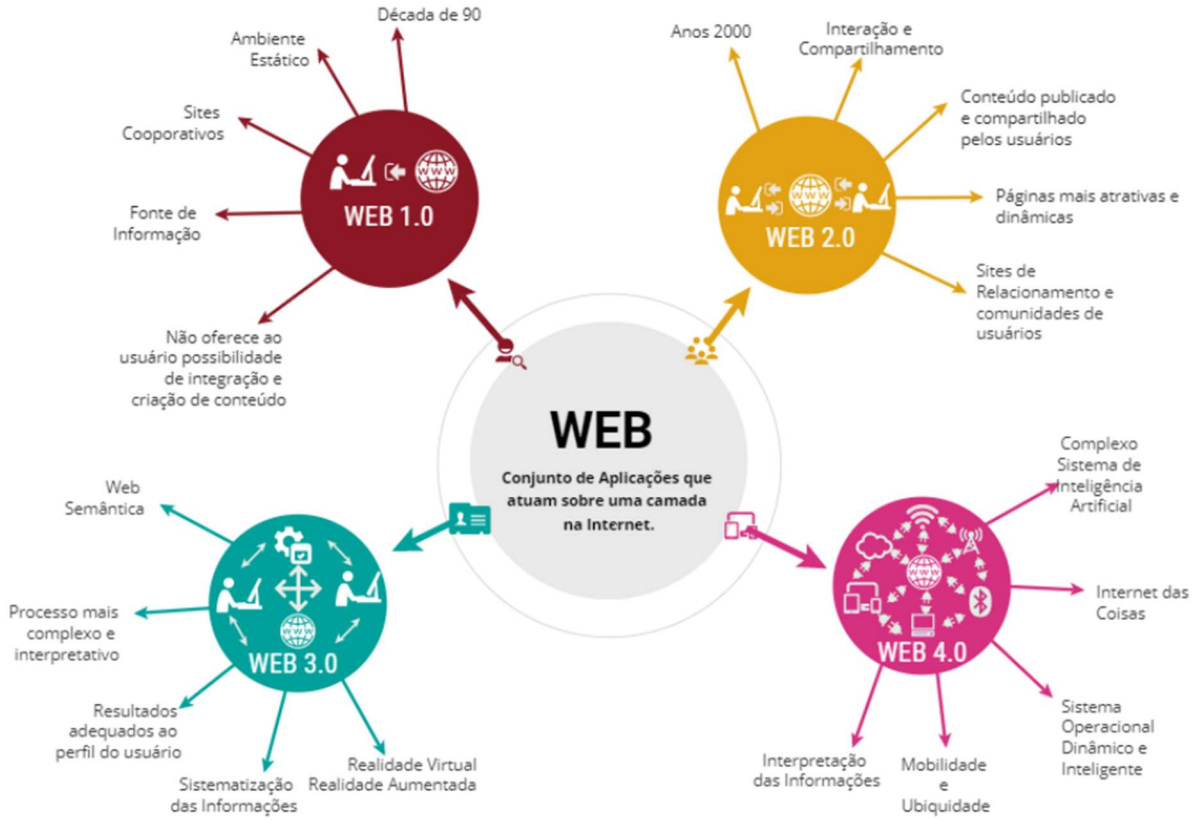
Sendo assim os principais objetivos do estudo de Trindade (2020) foram explorar as funcionalidades e técnicas essenciais para o desenvolvimento de uma PWA, identificar as principais ferramentas em produção audiovisual e com base nisso desenvolver uma PWA agregando estas ferramentas, para o desenvolvimento da aplicação denominada “Pau a Pixel Design Movies”.

O autor aliou estes pontos com a história da *web* desde sua criação e suas 4 etapas também representadas na Figura 18:

- A primeira: a *web* estética;
- A segunda: que buscou maior interação com o usuário final;
- A terceira: conhecida como *web* semântica que buscava uma maior adaptabilidade, e resultados mais precisos a cada usuário;
- A quarta: onde foca em *Internet of Things* (IOT).

Trindade (2020) salientou que a conexão vem aproximando cada vez mais pessoas a objetos, tornando a computação cada vez mais pervasiva.

Figura 18 - Comparação de aplicações nativas, webs e PWAs



Fonte: Trindade (2020).

Segundo Trindade (2020), PWA também seria um evento com o intuito de trazer a interligação de interfaces de comunicação para tornar mais ágil o desenvolvimento de aplicações, o que indica a possibilidade da abordagem PWA tornar-se algo que liga a aplicação ao sistema de forma padronizada e segura.

O motivo da escolha da abordagem PWA segundo o autor foi por conta do baixo custo, e menor tempo gasto em seu desenvolvimento, motivos que outros autores como Alumu (2020) também citam como principais. O autor também mostrou diferenças entre as três abordagens, nativas, híbridas e *web app*, o que reforça a escolha adotada.

O autor descreveu a evolução da *web* como em constante evolução e mudança, e PWA é mais um passo além neste desenvolvimento apontando quatro aspectos distintos, o primeiro deles é, aperfeiçoando a experiência do usuário em diversas plataformas, o segundo, a leveza de um aplicativo *web* sendo eficiente e rápido, o

terceiro, um modelo com padrões específicos, o quarto, utilizando a mesma base de código para várias plataformas.

Neste estudo o autor coloca JSON como sendo uma forma muito mais compacta de transmissão de dados por meio de APIs específicas comparado ao XML, JSON também foi escolhido como formato de transmissão de dados para o atual projeto trafegando-os entre o *backend* e a PWA.

No desenvolvimento da PWA do estudo de Trindade (2020) foram utilizadas tecnologias baseadas em JS para o *frontend,* sendo elas os *frameworks* VueJS e Quasar. Para o desenvolvimento *backend* o autor utilizou a linguagem de programação Java com a ferramenta Spring Boot conectada a um banco relacional SQL Server.

O autor utilizou 3 camadas base para a aplicação desenvolvida sendo elas: camada de apresentação onde contém a interface da aplicação com a PWA, a camada de negócio onde há o *backend* escrito em Java e contém as logicas de negócio, e a camada de persistência relacionada ao banco de dados.

Trindade (2020) implementou em seu protótipo questões características de uma PWA, como o uso *offline, splashscreen,* ícone e instalação, aplicando também conceitos de RWD e *mobile first.* O autor buscou um desenvolvimento forte no quesito *app like* tendo transições de tela semelhantes a uma aplicação nativa.

O autor teve êxito no uso de PWA para o projeto desenvolvido, apontando a possibilidade do uso da tecnologia em outros tipos de aplicações, porém, alega haver poucos recursos bibliográficos por ser uma abordagem de desenvolvimento nova.

3.4 Trabalho 4

No estudo intitulado “Dawnig of Progressive Web Applications (PWA)” Adetunji e Ajaegbu (2020), destacaram a crescente demanda de *software mobile*, onde compararam algumas abordagens, e destacaram a eficiência das PWAs.

O estudo buscou uma análise sistemática com o intuito de comparar as abordagens nativas, híbridas e de PWA, fazendo pesquisas a fim de compreender o conceito PWA de forma a encorajar o uso desta técnica.

Adetunji e Ajaegbu (2020) analisaram dados de pesquisas relacionadas a uso de dispositivos além da análise de referências para as comparações feitas, após a comparação das tecnologias envolvidas, os autores seguiram com a conclusão de seu estudo.

Os autores afirmam que na última década houve um aumento no uso de dispositivos moveis, sendo mais de cinco bilhões o número de pessoas que possuem um dispositivo móvel. Reforçando que é uma crescente significativa no uso deste tipo de aparelho.

Adetunji e Ajaegbu (2020), afirmam que PWA é uma tecnologia emergente no meio acadêmico, onde seus conteúdos tendem a ser novos, pois encontrando-se em ascensão, também é pouco explorada.

Os autores analisaram alguns pontos sobre PWAs comparando-os as outras abordagens, por exemplo, a distribuição ser através da internet via URL e não necessariamente via *stores*, o tamanho e velocidade de instalação ser mais baixa, porém com maior tempo para renderizar ao abrir.

No Quadro 5 os autores do estudo descreverem o ecossistema base do desenvolvimento de aplicações nativas.

Quadro 5 - Comparação do ecossistema das aplicações nativas

Platforms	Virtual Machine (VM)	Programming Language	Integrated Development Environment (IDE)	User Interface	Devices	Application Store
Android (Google)	Dalvik VM	Java	Eclipse, Android Studio, Android SDK	XML files	Heterogenous	Google Paly Store
IOS (Apple)	No	Objective-C or Swift	XCode	Cocoa Touch	Homogenous	Apple iTunes Store
Windows (Microsoft)	Common Language Runtime (CLR)	C# or C++	Visual Studio	XAML files	Homogenous	Windows Phone Market
Blackberry OS (Research in Motion – Rim)	BlackBerry Enterprise Server VM	Java	BlackBerry Plug-in for Eclipse	XML files	Homogenous	BlackBerry Apps World

Fonte: Adetunji e Ajaegbu (2020).

Segundo Adetunji e Ajaegbu (2020), inevitavelmente para desenvolver uma aplicação nativa, para os sistemas citados no Quadro 5, é obrigatório usar seu ecossistema, as principais vantagens de um sistema nativo, são a utilização de sensores e a possibilidade de integração entre APPs facilitada.

Porém os autores apontaram problemas de segurança que podem ser explorados em um APP nativo, a fragmentação das plataformas e a complexidade de teste como principais desvantagens desta abordagem de desenvolvimento.

Na Tabela 1 os autores do estudo compararam os principais pontos de cada uma das 3 abordagens sendo elas: a abordagem nativa, híbrida e PWA.

Tabela 1 - Comparação das 3 principais abordagens do desenvolvimento frontend

FEATURES	NATIVE	HYBRID	PWA
Installable	Yes	Yes	Yes
Offline Capability	Limited	Limited	Yes
Testable Before Installation	No	Yes	Yes
App Market Place Availability	Yes	Yes	Yes
Push Notification	Yes	Yes	Yes
Cross Platform Availability	No	Yes	Yes
Hardware and Platform Access	Yes	Yes	Limited
Background Synchronization	Yes	Yes	Yes
Security Layer	No	No	Yes
Link-Ability	No	No	Yes
Bookmark-Ability	No	No	Yes
Constantly Updated	No	No	Yes
Friction of Distribution	High	High	Low
Desktop Capability	No	No	Yes

Fonte: Adetunji e Ajaegbu (2020).

Os autores concluíram que o desenvolvimento *web* erradicou a fragmentação de outras plataformas, e está resolvendo o problema de redundância de *software* causado ao replicar a mesma interface para diferentes plataformas, podendo ser explorado buscando *softwares* mais eficientes em termos de uso de memória como um todo.

3.5 Trabalho 5

Em seu estudo denominado “Analilysis of deployng a React PWA on Google Play store using Trust Web Activity”, Alemu (2020) analisou o aprimoramento progressivo utilizando uma aplicação ReactJS nos quesitos necessários para tornar a aplicação uma TWA.

O sistema desenvolvido pelo autor denominou-se “Cinkino”, teve como objetivo demonstrar o processo de desenvolvimento de uma PWA o qual é, segundo Alemu (2020), pré-requisito para a implementação de uma TWA, possuindo essa a possibilidade de publicação na Google Play Store.

Alemu (2020) em seus objetivos, analisou e descreveu as ferramentas utilizadas para a implementação do projeto, assim como o fluxo de desenvolvimento de uma PWA e TWA.

O sistema utilizou uma API de cinemas denominada pelo autor como “Finnkino API” para desenvolver uma aplicação que pudesse ter as seguintes funções:

- Procurar filmes disponíveis nos Cinemas Finnkino;
- Filtrar filmes de acordo com localização e data;
- Manter marcadores dos filmes para visualização posterior;
- Ver a localização de teatros próximos;
- Checar as notícias sobre Finnkino.

Segundo o autor, o uso das metodologias PWA e TWA trazem uma maior performance, facilidades no desenvolvimento e na otimização, e menor custo de produção cerca 15 a 30 % mais barato que uma produção nativa, segundo pesquisa do autor, sendo uma boa opção para equipes menores de desenvolvimento.

Outro ponto destacado pelo autor é a popularidade da linguagem de programação JS, onde a linguagem é considerada a mais popular, segundo pesquisa do *Stackoverflow* (2019), o que pode tornar mais simples a procura de desenvolvedores para a metodologia apresentada.

O autor também buscou dados estatísticos de algumas aplicações que utilizaram a metodologia PWA para sua construção onde obtiveram alguns benefícios como, maior conversão, maior tempo de consumo, redução de consumo de dados, ganho de performance, e engajamento, como descrito na Figura 19.

Figura 19 - Casos de sucesso obtidas com o uso de PWA

Fonte: Alemu (2020).

Alemu (2020) optou pela utilização da ferramenta ReactJS para o desenvolvimento da PWA proposta e destacou a ferramenta por ser amplamente utilizada pelos desenvolvedores na *web*, tento versões que executam tanto em *desktops* com Electron e *mobile* com o React Native.

Em seu processo de desenvolvimento, o autor implementou o *Manifest* e SW, buscando apresentar a compatibilidade das *web-apis* utilizadas no estudo com a quantidade relativa de dispositivos compatíveis.

Além disso Alemu (2020) executou tarefas para otimização do *software*, utilizou 3 ferramentas de compressão para redução de código, comparando o desempenho final e escolhendo a com melhor performance.

Utilizando uma técnica chamada *Code Splitting*, o autor do estudo realizou a divisão de código utilizando uma biblioteca interna chamada React Lazy. Esta técnica possibilita a repartição do código da aplicação em partes menores a fim de reduzir o *loading* inicial da aplicação, carregando o restante conforme a utilização do cliente. Em análise Alemu (2020), demonstrou uma redução de *first load* de 162 kb para 145

kb e uma melhora no *ranking* de performance do Lighthouse de 3 pontos, salientando uma melhora mais significativa em sistemas maiores.

Outra técnica imposta pelo autor consiste em utilizar o atributo *srcset* em *tags img* dentro do HTML do projeto, podendo utilizar tamanhos variados de imagens a fim de otimizá-los para diferentes tamanhos de tela como na Figura 20.

Figura 20 - Amostra de uso do atributo srcset nas tags img

Fonte: Alemu (2020).

Alemu (2020) utilizou a ferramenta Workbox para a geração do SW e gerenciamento de *cache*, utilizando majoritariamente a tática de *cache* chamada *Stale While Revalidate* abordada na seção 2.1.2, neste projeto o autor decidiu não adicionar as imagens de conteúdo em *cache*, deixando apenas o *app shell* da aplicação.

Segundo Alemu (2020), no Google I/O do ano de 2019 fora apresentada pela primeira vez o conceito de TWA, onde foi considerada a melhor maneira de exibir conteúdo *web* em *full-screen* em um APP Android.

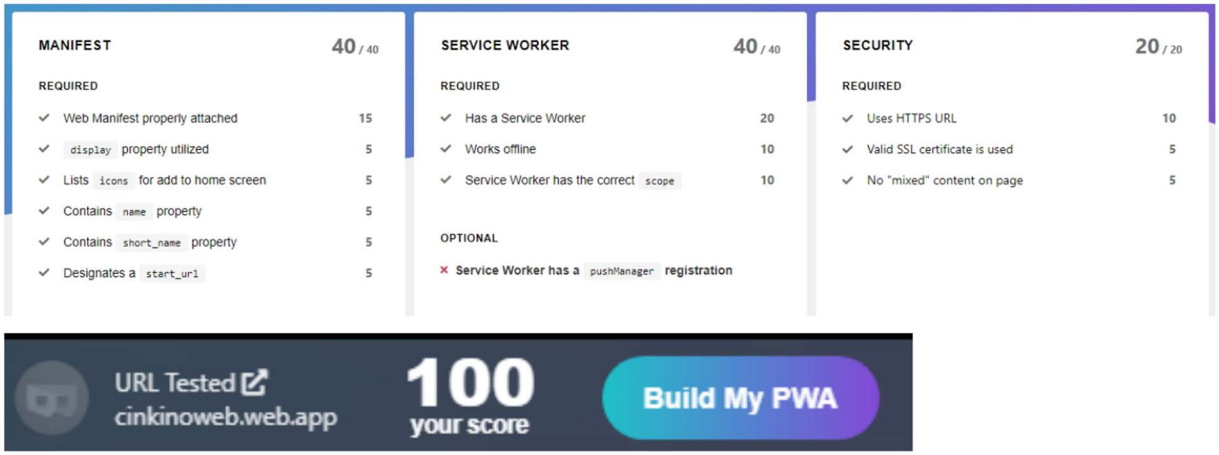
Alemu (2020) mostrou o processo de geração da Android Package (APK) do aplicativo Cinkino para a adição na Play Store processo que só é possível após ter uma PWA válida.

Segundo Alemu (2020), este processo pode ser feito através de 2 ferramentas o Bubble Wrap CLI criada pela Google, ou o PWABuilder que é uma ferramenta criada pela Microsoft que utiliza o Bubble Wrap CLI para a geração da APK, afim de simplificar o processo.

O autor do estudo utilizou o PWABuilder passando a URL da PWA, a ferramenta então executou uma análise da PWA para checar se os pontos obrigatórios, recomendados e opcionais estão válidos gerando uma análise

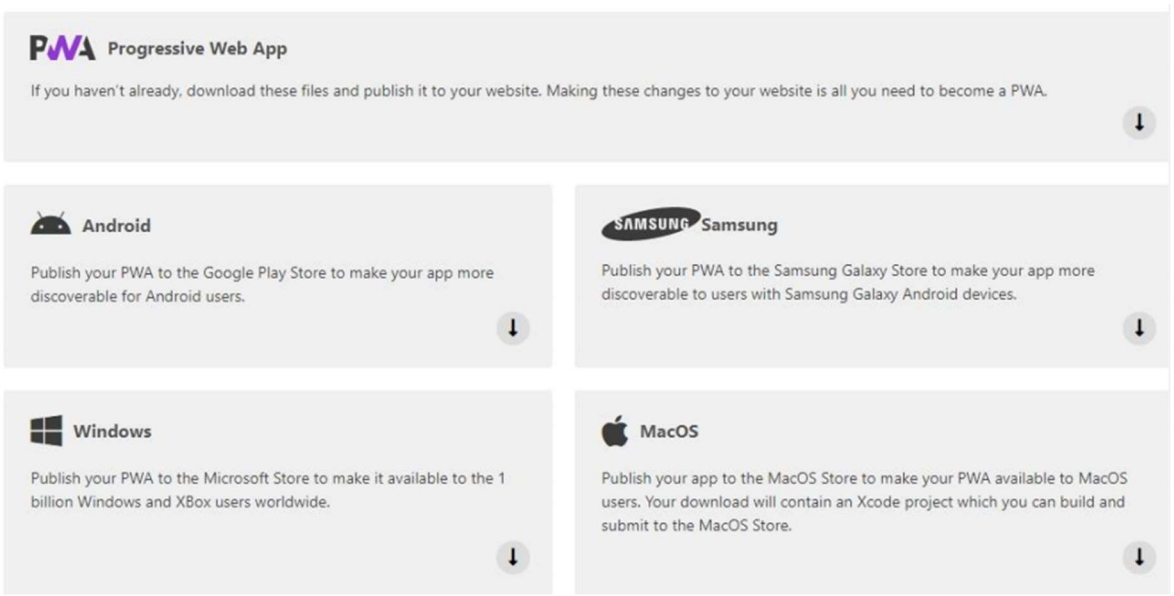
semelhante à da Figura 21. Com esse processo foi possível a geração de um *build* da aplicação para as plataformas descritas na Figura 22.

Figura 21 - Resultado da análise do PWABuilder feita para o APP Cinkino



Fonte: Alemu (2020).

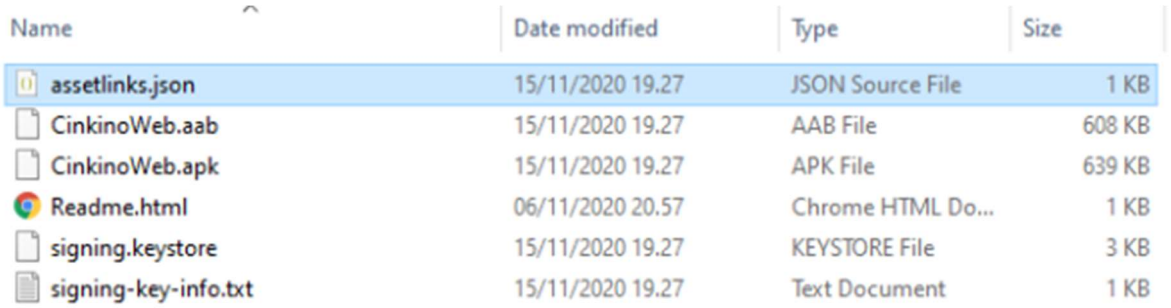
Figura 22 - Opções de build disponíveis em 2020 na aplicação PWABuilder



Fonte: Alemu (2020).

Segundo Alemu (2020) para a TWA funcionar corretamente na loja do Google é necessário a criação de um arquivo de configuração chamado *Digital Asset Links* que é adicionado ao site de maneira pública, para que haja a verificação de propriedade de conteúdo. Ao seguir na opção Android, o autor teve o retorno dos arquivos descritos na Figura 23.

Figura 23 - Arquivos gerados pela ferramenta PWABuilder no build Android



Fonte: Alemu (2020).

Na Figura 23, segundo Alemu (2020) há arquivos necessários para lançar a aplicação na Google Play, por exemplo a APK, para disponibilizar como *download* para os usuários e o *Digital Asset Links* como citado anteriormente. O autor observa que o tamanho da APK resultante é consideravelmente menor que em meios nativos ou híbridos.

Alemu (2020) conclui que há um grande potencial no uso das tecnologias PWA e TWA destacando que, por ter um desenvolvimento mais simples e ser familiar para o ambiente *web* e seus desenvolvedores, esta metodologia pode ser mais economicamente viável para empresas de pequeno porte.

Entretanto o autor observou que o processo de otimização pode ter uma carga de trabalho maior para entrar no que seria o mínimo exigido para o processo do TWA ter sucesso.

Segundo Alemu (2020) o uso do PWABuilder é altamente recomendado por contar com recursos completos que visam trazer mais facilidade no desenvolvimento. A autor lançou o APP Cinkino como versão de teste aberto, e até o término do estudo a Google Play não havia feito o processo de revisão, necessária para a publicação no APP.

3.6 Trabalho 6

No estudo intitulado “Safe Space: Um aplicativo de denúncias e auxílio legislativo para vítimas de violência”, Junior (2020) desenvolveu um aplicativo com o intuito de mapear os casos de violência, principalmente contra minorias, oferecendo um mapa geoprocessado com marcações que indicam o número de acontecimentos e auxiliem no contato a serviços básicos.

O autor relatou um crescente aumento da violência nos últimos anos, destacando o período pandêmico causado pelo COVID-19. Foi notada a necessidade da implementação de uma ferramenta para o reporte de tais atos.

Com esse apontamento o estudo de Junior (2020) teve como objetivos auxiliar através do mapeamento, locais de risco, e de auxílio a vítimas de violência, assim como contribuir para a segurança pública destes locais.

Outro objetivo destacado pelo estudo foi o teste da aplicação em um cenário real, coletando dados para gerar métricas afim de medir a viabilidade do uso da aplicação e do estudo proposto.

Segundo Junior (2020), esta ferramenta também adquiriu a finalidade de informar ao seu usuário das legislações vigentes para os atos denunciados, auxiliando na informação e no auxilio jurídico.

Assim como o presente estudo, Junior (2020) buscou com sua aplicação uma melhoria no convívio social geral, destacando o medo das pessoas ao sair de casa por conta da violência e dois crimes de ódio.

O autor em seu referencial, buscou pelo histórico de violência voltada a mulheres e minorias, e também a como é tratado este tipo de crime, e como os casos são computados, mostrando que o processo de georreferenciamento aplicado em 2020 é em suma através de pesquisa publica, havendo também uma carência de dados neste aspecto.

Junior (2020) compilou em seus trabalhos relacionados estudos voltados a violência das minorias, dados estatísticos e aplicações de denúncia, além disso mostrou sobre movimentos contra a violência.

Em um destes trabalhos apresentou o APP denominado “Isis” que também é voltado a criação de denúncias, porem mostrando um recurso específico deste sistema, uma tela *fake* para que um marido não perceba que é uma aplicação de denúncias.

Em outro destes estudos Junior (2020) destacou a importância de uma categorização mais precisa para um levantamento mais exato das estatísticas dos crimes de ódio no Brasil.

O estudo mostra as tecnologias utilizadas e o seu foco, sendo elas: Python, Flask, PostgresSQL, e React Native. Junior (2020) teve foco na aplicação mobile através do React Native e no seu funcionamento na plataforma Android.

Para a amostragem da aplicação foi exibido o fluxo da utilização do aplicativo, assim como o modelo relacional do banco de dados, requisitos funcionais e não funcionais.

A marcação dos locais de risco teve seu registro por bairro, tendo sinais variados de acordo com o número de denúncias feitas em determinado local. Além disso ofereceu um fluxo baseado no imediatismo da ocorrência podendo submeter uma ligação para polícia, ambulância ou contatos de emergência de forma rápida.

Para o auxílio estatístico o APP ofereceu um cadastro de usuário contendo dados como religião, orientação sexual e gênero, dados que poderiam auxiliar nos registros estatísticos específicos, dando maior precisão nos dados coletados. A aplicação também contou com um registro de denúncias objetivo, onde buscava saber o que aconteceu, onde aconteceu e quando aconteceu.

O aplicativo proposto por Junior (2020) ofereceu uma tela para visualizar as denúncias feitas, e por meio desta exibir as legislações vigentes para a categoria de transgressão registrada.

Junior (2020) realizou diversas pesquisas para validar e avaliar a aplicação por ele proposta, sendo estas perguntas referentes principalmente a violências sofrida e a interface da aplicação.

Em um conjunto de 100 pessoas, quando questionadas sobre terem sofrido violência 29% responderam que sim, porém se mostraram extremamente positivas quando questionadas sobre o auxílio jurídico, tendo 72% das pessoas o interesse de ter algum tipo de ajuda.

Já para análise da aplicação foram selecionadas 8 pessoas para avaliarem as funcionalidades e telas da aplicação. A tela para os acessos de emergência apesar de fácil de utilizar, Junior (2020) conclui que seria mais relevante se esta opção estivesse na tela de bloqueio.

Ambas as telas desenvolvidas na aplicação foram julgadas fáceis de utilizar e importantes pelos entrevistados, tendo também sugerido uma adição de informações na tela de legislações e um campo para detalhar a denúncia feita.

Apesar da aprovação dos usuários, foi visto como importante alguma integração com sistemas governamentais para enriquecer o mapa de georreferenciamento, o que proporcionaria um maior índice de compartilhamento do APP, e consequentemente o seu uso. Além disso 85% dos entrevistados baixariam o APP caso sofressem algum crime de ódio.

O conhecimento jurídico aumentou consideravelmente dentre os entrevistados, fornecendo também, uma maior sensação de segurança, salientando que essa sensação seria maior ainda caso o conjunto de informações fossem maiores.

Junior (2020) conclui que a informação jurídica e o auxílio através da ferramenta são de extrema importância, porem a funcionalidade de chat foi descartada pois foi visto que seria mais maléfico do que benéfico ao usuário com trauma. O autor através das pesquisas realizadas teve uma ideia bem formada dos próximos passos para a continuação da ferramenta e de melhorias para ela.

3.7 Comparativo entre os trabalhos relacionados

O Quadro 6 mostra a comparação entre os cinco trabalhos relacionados. Os principais pontos de comparação são os seguintes: os objetivos, a estrutura de cada estudo, a metodologia utilizada, e como foi efetuada a validação de cada um deles.

Quadro 6 - Comparativo dos trabalhos relacionados

Autores	Ceconi (2018)	Nedel (2020)	Trindade (2020)	Adetunji e Ajaegbu (2020)	Alemu (2020)	Junior (2020)
Objetivos	Desenvolver uma aplicação voltada a área de turismo utilizando as metodologias de desenvolvimento de uma PWA como, recursos de *cache* para o funcionamento *offline*, e geolocalização.	Desenvolver uma aplicação de cardápio digital, que permita simplificar o atendimento em restaurantes, coletar dados para gestão e para avaliação do sistema. Tudo isso utilizando PWA em toda a aplicação SPA, tanto para o gestor, a cozinha, quanto para	Explorar as funcionalidades e técnicas essenciais para o desenvolvimento de uma PWA, identificar as principais ferramentas em produção audiovisual e com base nisso desenvolver uma PWA agregando estas ferramentas, para o desenvolvimento da aplicação	Visa uma análise sistemática com o intuito de comparar as abordagens nativas, híbridas e de PWA, fazendo pesquisas a fim de compreender o conceito PWA de forma a encorajar o uso desta técnica.	Demonstrar o processo de desenvolvimento de uma PWA o qual é pré-requisito para a implementação de uma TWA, possuindo essa a possibilidade de publicação na Google Play Store.	Através de uma ferramenta de denúncia, visa auxiliar vítimas de crimes de ódio e contra minoria, assim contribuindo no convívio social e segurança pública. Visa também validar essa ferramenta em um cenário real, e analisar seus resultados.

		o cliente do estabelecimento.	denominada “Pau a Pixel Design Movies”.			
Estrutura	Quatro Etapas: busca por referências, avaliação das informações coletadas, desenvolvimento, avaliação sobre o resultado do projeto.	Seis Etapas: levantamento de informações, estudo das tecnologias aplicadas, levantamento de requisitos, modelagem do sistema, prototipação, testes, avaliação.	Seis Passos: pesquisa exploratória sobre o tema, exploração de aplicações similares, definição do objeto/problema de pesquisa, levantamento de requisitos, elaboração do projeto de *Software*, implementação do *Software*.	Três Etapas: análise literária, abordagens de desenvolvimento comparadas, conclusão.	Quatro Etapas: introdução, ferramentas, implementação, referencial.	Sete Etapas: Introdução, fundamentação teórica, trabalhos relacionado, materiais e métodos, desenvolvimento, análise de resultados e conclusão.
Métodos utilizados	*User experience*, busca de referências, avaliação de informações, implementação de PWA, estudo comparativo,	Automação de processos, estudo comparativo, evolução de mercado, modelagem de procediment	Computação pervasiva e ubíqua, *design* de interface, evolução histórica da *web*, estudo evolutivo, estudo	Análise sistemática, abordagem comparativa, análise qualitativa.	Aprimoramentos de *Software*, técnicas modernas de desenvolvimento *web*, publicação de APPs na	*User experience*, busca de referências, avaliação de informações, análise de informações, modelagem de

	aprimoramento com camadas, implementações de técnicas de SEO.	os e fluxos, delimitação de projeto, pesquisas com usuários.	comparativo, lógica em camadas, estudo de ferramentas.		Google Play, utilização de APIs públicas, análise de custos, análise de popularidade das ferramentas utilizadas, casos de sucesso, exemplos reais, fluxo de desenvolvimento, *code splitting*, *lazy load*, *data source set*, uso de abstrações.	procediment o e fluxos, delimitação de projeto, pesquisas com usuários, uso de dados geográficos.
Validação	Lighthouse.	Pesquisas de satisfação com os clientes.	Utilização PoC para avaliar a viabilidade da técnica PWA, para o desenvolvimento de uma ferramenta na área áudio visual.	Comparação entre plataformas.	Light House, PWABulder, Google Play Store.	Pesquisas com os usuários.

Fonte: Autor, 2021.

Com a análise comparativa realizada nos seis trabalhos escolhidos é possível perceber pontos importantes em comum, e também seus pontos de destaque, sendo importante salientar alguns destes pontos.

Em cinco trabalhos analisados pode-se ver alguma maneira de comparar PWA a outros meios de desenvolvimento, o que faz com que seja possível ser bem preciso sobre o que uma PWA pode ou não fazer ou até mesmo ser superior ou inferior que outro meio.

Trindade (2020) salienta bastante a história da *Internet* desde sua criação com páginas estáticas, como mostrado, muita coisa evoluiu assim com o público que utiliza a *web* que segundo Alemu (2020) torna-se cada vez maior e dinâmico, e recentemente mais móvel também. Os estudos aqui comparados também mostraram a importância da experiência do usuário que está cada dia mais exigente.

Segundo os estudos analisados, evolução, experiência de usuário, mobilidade e dinamismo é o que levou o surgimento da metodologia de desenvolvimento PWA.

Outro ponto em comum é o quanto essa abordagem pode reduzir custos de desenvolvimento e ampliar o retorno e engajamento final da aplicação o que fica mais evidente nos estudos de Alemu (2020).

É possível ver principalmente nos estudos de Alemu (2020), Ceconi (2018) e Nedel (2020) o processo de desenvolvimento de uma PWA, e bons exemplos de aplicações que podem seguir esta linha de desenvolvimento, assim como as formas que elas podem ser moldadas.

Além disso, foi possível perceber o quanto esta técnica é emergente no meio acadêmico por contarem com artigos recentes em sua maioria, e o quanto PWA pode evoluir respeitando um padrão, mantendo a segurança, melhorando o desenvolvimento e resultado final das aplicações.

Já no que tange a violência Junior (2020) apresentou vários dados estatísticos que comprovam o aumento da violência contra a mulher e minorias o que traz a importância de se desenvolver aplicações que visam justamente auxiliar estas vítimas.

Neste capítulo foram apresentados cinco estudos semelhantes, afim de realizar uma análise comparativa entre eles. O próximo capitulo abordará os procedimentos metodológicos, mostrando as tecnologias e a construção da pesquisa realizadas neste trabalho.

4 PROCEDIMENTOS METODOLÓGICOS

Este capítulo aborda os procedimentos metodológicos, métodos de pesquisa e abordagem, utilizados no desenvolvimento deste trabalho, assim como os objetivos de pesquisa e seus procedimentos técnicos.

4.1 Pesquisa enquanto aos métodos científicos

O presente estudo utiliza o método dedutivo, pois encontra-se como o mais adequado para dados pré-coletados com o intuito de gerar pressupostos onde visam trazer maior exatidão no resultado final.

O método dedutivo é segundo Lakatos e Marconi (2003) o método mais adequado para as comparações e medições de cunho matemático, onde no presente estudo haverão dados matemáticos provindos de algumas ferramentas como Google Lighthouse e PWABuilder.

O presente estudo também utilizará o método dialético visto que as tecnologias descritas neste estudo estão em constante evolução, podendo elas mudarem de estudo para outro ou mesmo em um período de tempo específico. O método dialético é empregado neste estudo principalmente nos pontos que podem divergir entre autores, buscando a melhor explicação ou exemplificação.

4.2 Pesquisa enquanto ao modo de abordagem

O presente trabalho utiliza tanto a abordagem qualitativa quanto quantitativa, empregando uma análise qualitativa em pontos subjetivos de comparação, onde tendem a ter margem para interpretação, como as comparações nas abordagens de desenvolvimentos citados no capítulo 3 ou mesmo na comparação entre plataformas.

Além disso o estudo também conta com métricas quantitativas, tais como desempenho da aplicação, tamanho da aplicação. Estas métricas estarão contidas principalmente nas interações da aplicação com as ferramentas: Google Lighthouse e PWABuilder onde trazem medidas quantificáveis.

4.3 Pesquisa enquanto aos fins da pesquisa

O estudo emprega uma pesquisa de caráter exploratório e descritivo, pois tem o intuído de explorar a abordagem de desenvolvimento PWA, afim que agregar conhecimento.

Com isso o estudo buscou comparar trabalhos relacionados para explorar os procedimentos padrões de desenvolvimento desta metodologia, além de buscar comparações entre outras metodologias de desenvolvimento para agregar aos fins de pesquisa selecionados.

Outro ponto explorado são comparações entre plataformas no desenvolvimento deste estudo, desenvolvendo a aplicação, comparando, aplicando e descrevendo estes conceitos.

4.4 Pesquisa enquanto aos procedimentos técnicos

As metodologias utilizadas neste trabalho são a pesquisa experimental, pesquisa documental e pesquisa bibliográfica.

A pesquisa bibliográfica apesar de ter uma boa abrangência no que tange as ferramentas mais comuns no desenvolvimento de PWAs, ainda assim não trazem toda a informação necessária, sendo preciso seguir também outros métodos.

O presente estudo utiliza também o método de pesquisa experimental pois muitos processos foram aplicados de forma a conduzir testes, sendo que estes geraram resultados descritos no estudo.

Outro ponto no presente estudo são os diversos ferramentais utilizados no desenvolvimento, pois muitos deles não são utilizados diretamente em estudos anteriores, sendo necessária uma maior pesquisa em tais ferramentas e metodologias para a sua aplicação neste trabalho, sendo necessária o uso da pesquisa documental.

No presente capítulo foram apresentadas as metodologias utilizadas neste estudo, no capítulo seguinte é detalhado o procedimento de desenvolvimento do projeto proposto.

5 DESENVOLVIMENTO

No capítulo anterior foram abordados os métodos empregados na pesquisa deste estudo, no presente capítulo serão descritos os procedimentos de desenvolvimento do projeto proposto, assim como a aplicação das tecnologias descritas neste estudo.

5.1 Ambientes de teste e desenvolvimento

Como o projeto objetiva ser multiplataforma, é importante haver um ambiente de testes onde seja possível uma gama ampla de sistemas e aparelhos, principalmente com o objetivo de testar o projeto e verificar as particularidades entre plataformas e *browsers* caso haja alguma.

Com isso definido, tem-se disponíveis 3 plataformas onde serão utilizadas para os testes, estes ambientes são descritos nos Quadros 7, 8 e 9.

Quadro 7 - Descrição da plataforma de testes 1

Sistema:	Windows 10
Ficha Técnica:	**Ambiente:** Máquina Real **Processador**: AMD Rysen 5 4650g **Memória Ram**: 16 GB **Placa de Vídeo:** Veja 7 (gráficos integrados) **Câmera:** Não
Browsers:	Chrome, Firefox, Brave, Edge

Fonte: Autor, 2021.

Quadro 8 - Descrição da plataforma de testes 2

Sistema:	Android 11 (Android ONE)
Ficha Técnica:	**Ambiente:** Máquina Real **Processador**: Qualcomm Snapdragon 665 (Kryo 260 Octacore) **Memória Ram**: 4 GB **Placa de Vídeo:** Adreno 610 **Câmera:** Sim
Browsers:	Chrome, Brave

Fonte: Autor, 2021.

Quadro 9 - Descrição da plataforma de testes 3

Sistema:	iPhone 11 pro – Browser Stack
Ficha Técnica:	**Ambiente:** Simulador (Browser Stack)
Browsers:	Safari

Fonte: Autor, 2021.

Para a aplicação proposta, os principais testes foram executados nas máquinas descritas nos Quadros 7, 8 e 9, sendo que em ambiente *desktop* os principais testes foram feitos utilizando Windows 10 e os testes *mobile* utilizando as plataformas Android e iOS.

5.2 Definição da aplicação proposta

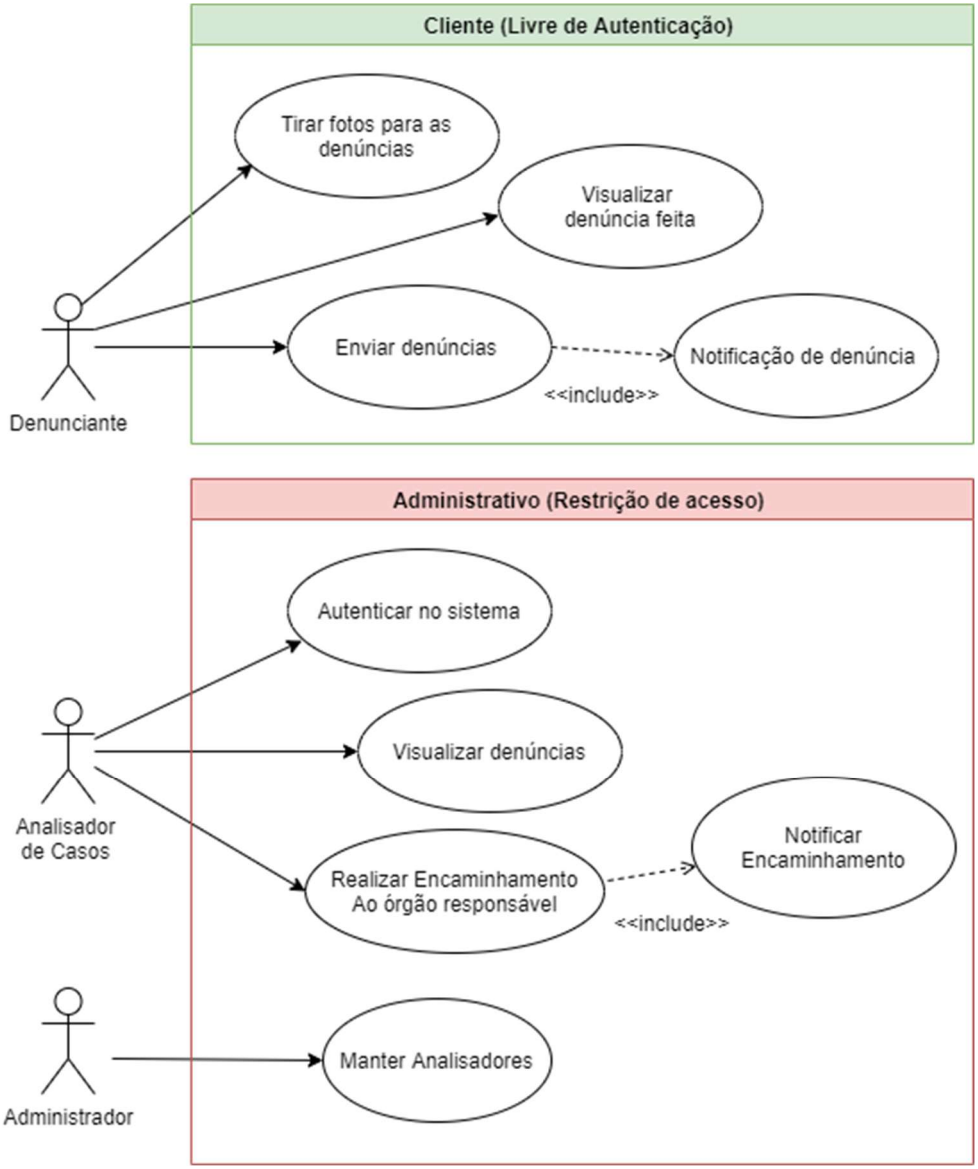
Esta seção tem o intuito de utilizar metodologias técnicas de *software* para definir a proposta da aplicação, afim de delimitar e examinar tanto em funcionalidades, quanto em fluxos de operação. As principais abordagens para essa definição serão os casos de uso e levantamento de requisitos.

5.2.1 Casos de uso

Segundo Cockburn (2007), um caso de uso tem a finalidade de mostrar o comportamento do sistema apartir de uma ação feita por uma parte interresada. No projeto em questão posui 3 atores sendo eles o denunciante, o analisador de casos e o administrador, onde suas ações são descritas na Figura 24.

A aplicação proposta tem como foco o denunciante, onde o mesmo pode em sua denúncia, utilizar de recursos mais avançados do *browser*, que é justamente um dos objetivos deste estudo.

Figura 24 - Representação dos casos de uso



Fonte: Autor, 2021.

5.2.2 Requisitos

Para a aplicação proposta são definidos alguns requisitos funcionais e não funcionais, para delimitar e indicar as funcionalidades a serem desenvolvidas afim de segmentar, detalhar e priorizar a implementação. Nos Quadros 10 e 11 é possível observar os requisitos da aplicação proposta.

Quadro 10 - Requisitos não funcionais

Descrição do requisito	Prioridade
1 - Ser uma PWA Aplicação deverá ser desenvolvida como uma PWA, seguindo os padrões definidos da metodologia.	Alta
2 - Utilizar *Javascript* Aplicação deverá utilizar *Javascript* como linguagem de programação tanto no *frontend* quanto no *backend*.	Alta
3 – Utilizar NextJS Utilizar NextJS como *framework* para o *frontend* e também para o *backend*	Alta
4 – Ser uma TWA A aplicação deverá dispor de todos os padrões necessários para que entre na Google Play Store como uma TWA.	Alta
5 – Utilizar JWT para autenticação O método de autenticação deverá ser feito utilizando *Json Web Token.*	Alta
6 – Utilizar banco de dados PostgresSQL Aplicação deverá utilizar o banco de dados PostgresSQL,	Alta
7 – Ser testada em no mínimo 2 plataformas A aplicação deve ser testada em no mínimo 2 plataformas uma *mobile* e outra *desktop*.	Alta
8 – Ser testada em no mínimo 2 navegadores A aplicação deve ser testada em no mínimo 2 navegadores distintos.	Alta
9 – Utilizar Amazon S3 como *file storage* O sistema deverá utilizar a ferramenta S3 para o envio de arquivos para a nuvem.	Alta
10 – Persistir aplicação em *cache* por pelo menos 1 ano Aplicação deverá manter o essencial em cache no aparelho do cliente por no mínimo 1 ano.	Alta
11 – Utilizar PrismaJS como ORM O sistema deverá utilizar a ORM PrismaJS para a criação das migrações e esquema de dados do sistema.	Alta
12 – Utilizar compressor de imagens antes do envio	Alta

O sistema deverá reduzir as imagens antes de seu envio ao sistema, reduzindo o consumo de espaço no S3, e diminuindo o tempo de envio da denúncia.	
13 – A aplicação deverá ser *App-Like*	Alta
A aplicação proposta deverá se assemelhar a um *app mobile.*	
14 – Utilizar API terceiras	Alta
O sistema deverá utilizar APIs de terceiros onde possa facilitar a experiência do usuário, como tradução da geolocalização para dados como cidade e estado, e a busca de cidades e estados.	
15 – O sistema deverá salvar a denúncia também no dispositivo	Alta
Como o sistema não requer autenticação para o envio de denúncias, as denúncias além de serem salvas no sistema, também deverão ser salvas no dispositivo, para que o usuário possa resgata-la quando achar necessário.	

Fonte: Autor, 2021.

Quadro 11 - Requisitos funcionais

Descrição do requisito	Prioridade
1 – Manter denúncias	Alta
O sistema deverá manter o cadastro de denúncias.	
2 – Permitir o denunciante enviar ou tirar até 4 fotos	Alta
O sistema deverá permitir o envio de até 4 fotos por denúncia.	
3 – Permitir que o denunciante possa tirar Fotos	Alta
Ao estar em uma plataforma móvel o denunciante deverá poder tira fotos a partir de sua câmera.	
4 – Permitir que o denunciante acompanhe o andamento de sua denúncia	Alta
O sistema deverá permitir ao denunciante acompanhar o andamento e os dados de sua denúncia.	
5 – Permitir visualizar denúncias feitas no dispositivo através de uma listagem	Alta
O sistema deverá permitir a visualização de uma lista das denúncias feitas através do dispositivo.	
6 – Permitir o uso da geolocalização na criação da denúncia	Alta
O sistema deverá permitir a utilização da geolocalização para o preenchimento da localização agilizando o processo de preenchimento.	
7 – Permitir aos usuários a visualização dos dados da denúncia	Alta
O sistema deverá fornecer dados completos sobre as denúncias realizadas.	

8 – Permitir aos usuários o compartilhamento da denúncia	Alta
O sistema deverá permitir o compartilhamento da denúncia através do *share api* para outros utilizadores da plataforma poderem visualizar e acompanhar a denúncia	
9- Permitir ao denunciante o envio de imagens caso esteja em uma plataforma *desktop*	Alta
Caso o denunciante não tiver uma câmera ou estiver em uma plataforma *desktop* o mesmo deverá poder buscar uma foto em seu dispositivo para o envio	
11 – Permitir o salvamento da denúncia por um dispositivo terceiro	Alta
Ao acessar a denúncia o sistema deverá verificar se a denúncia está salva no dispositivo, se a mesma não estiver, deverá dar a opção de salvar através de um botão na tela.	

Fonte: Autor, 2021.

5.3 Evolução da aplicação proposta

Esta seção tem o intuito demonstrar o desenvolvimento da aplicação proposta seguindo os pontos apresentados no capítulo 2, mostrando os comportamentos nos Casos de Uso, e as funcionalidades delimitadas nos Requisitos.

5.3.1 Configurando a PWA

Assim como na seção 2, para que uma PWA possa funcionar da forma correta são necessárias algumas configurações, segundo Tandel e Jamadar (2018) e Ceconi (2018) uma delas é o arquivo *manifest.json*, onde contém informações básicas sobre o aplicativo e também, como o mesmo deve-se comportar.

Figura 25 - Representação do manifesto da aplicação proposta

```
{
  "name": "Reporti - Denúncias Online",
  "short_name": "Reporti",
  "categories": ["Denúncias", "Alertas"],
  "description": "Sistema de denúncias Online",
  "display": "standalone",
  "orientation": "portrait",
  "theme_color": "#003B36",
  "background_color": "#003B36",
  "start_url": "/",
  "icons": [
    {
      "src": "/icons/icon48.png",
      "sizes": "48x48",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "/icons/icon96.png",
      "sizes": "96x96",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "/icons/icon192.png",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "/icons/icon512.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }
  ],
```

Fonte: Autor, 2021.

Na Figura 25 é possível ver um conjunto de informações estruturadas em formato JSON, estas informações tem como objetivo, descrever a aplicação ao navegador segundo Tandel e Jamadar (2018).

Segundo Ceconi (2018) com estas informações é possível determinar as configurações básicas da aplicação, e podem ajudar no descobrimento da aplicação através de plataformas de busca como o Google.

Na Figura 25 tem-se algumas informações da aplicação em desenvolvimento, como o nome, categorias, descrição, cores tema, modo de exibição e seus ícones, que serão utilizados pelo navegador para dar identidade a aplicação assim que a mesma é lida ou instalada pelo navegador.

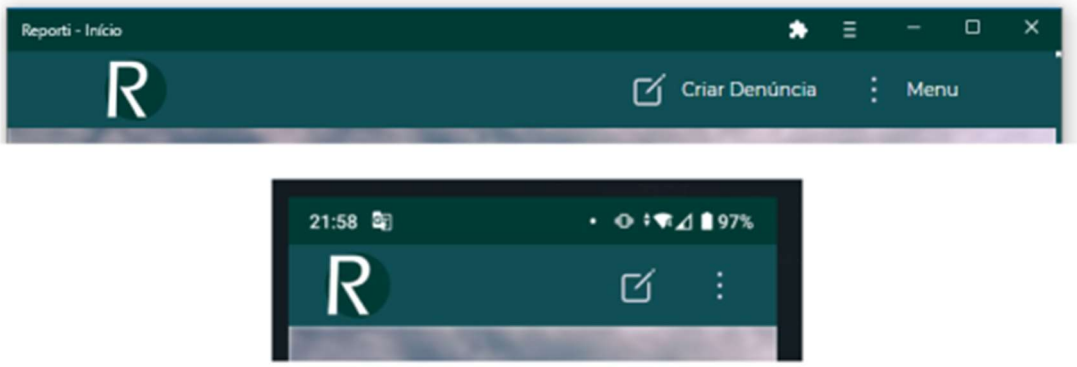
Na Figura 26 pode-se observar a representação do ícone na área de trabalho, tanto no Android à esquerda quanto no Windows à direita, onde é possível notar o uso dos ícones e nomes configurados no manifesto.

Figura 26 - Representação do ícone da aplicação

Fonte: Autor, 2021.

Na Figura 27 quando a aplicação está aberta, nota-se a utilização do *display* e do *theme color*, tanto na aplicação no Windows representada na parte de cima da Figura 27 quanto no Android parte de baixo da Figura 27.

Figura 27 - Representação do cabeçalho do sistema



Fonte: Autor, 2021.

Na Figura 28 tem-se um atributo de tipo *array* presente no manifesto chamado *shortcuts,* cada item dentro dessa array deve conter: *name*, *url*, *description* e *icons*. Este recurso ajuda a elencar um ícone de acesso rápido para alguma página específica do sistema.

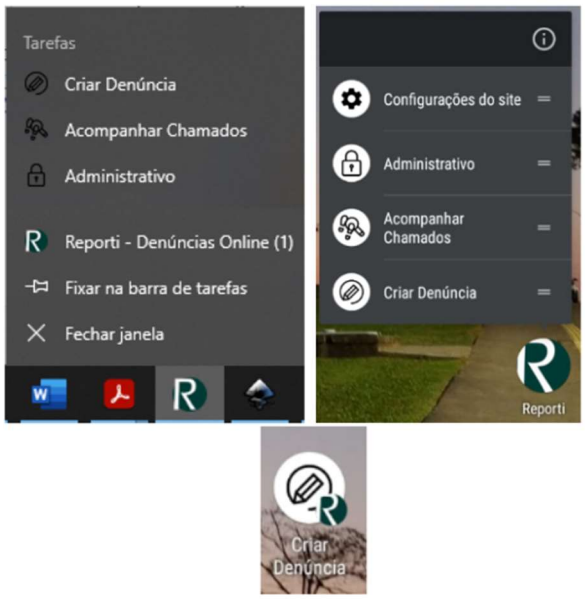
Figura 28 - Representação do shortcuts no manifesto

```
"shortcuts": [
  {
    "name": "Criar Denúncia",
    "url": "/criar-denuncia",
    "description": "Crie sua denúncia Online",
    "icons": [
      {
        "src": "/icons/crie192.png",
        "type": "image/png",
        "sizes": "192x192"
      },
      {
        "src": "/icons/crie128.png",
        "type": "image/png",
        "sizes": "128x128"
      }
    ]
  },
  {
    "name": "Acompanhar Chamados",
    "url": "/acompanhar-chamados",
    "description": "Acompanha os Chamados feito por Você",
    "icons": [
      {
        "src": "/icons/follow192.png",
        "type": "image/png",
        "sizes": "192x192"
      },
      {
        "src": "/icons/follow128.png",
        "type": "image/png",
        "sizes": "128x128"
      }
    ]
  },
  {...
  }
]
```

Fonte: Autor, 2021.

Na Figura 29 pode-se ver os *shortcuts* funcionando no sistema Windows a esquerda quando pressionado o botão direito no ícone da aplicação na barra de navegação do sistema, já no Android a direta é possível acessá-los segurando o ícone pressionado. O sistema Android permite a adição de um *shortcut* como um ícone na tela principal, como mostrado na parte inferior da Figura 29.

Figura 29 - Shortcuts nos sistemas Android e Windows

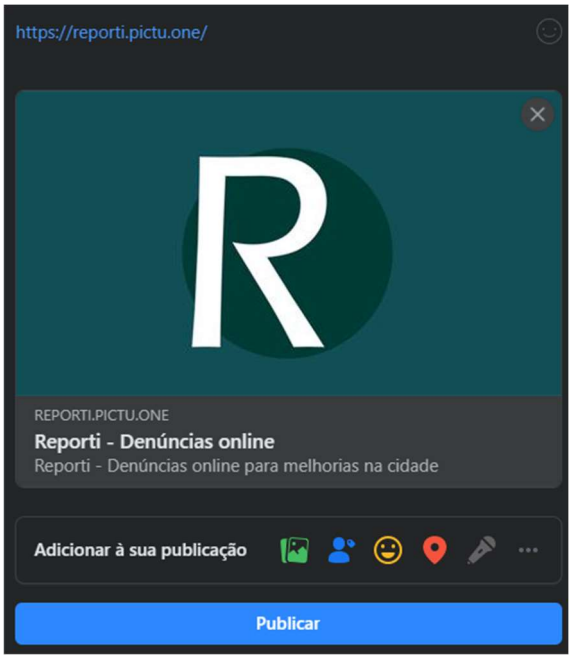


Fonte: Autor, 2021.

5.3.2 Metadados, SEO e rastreabilidade

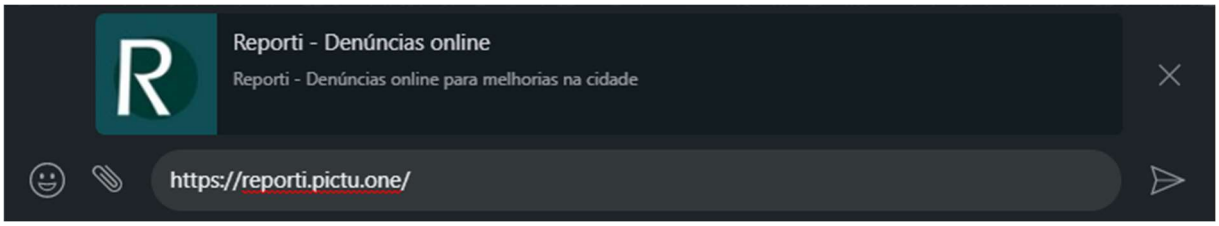
Uma das bases de uma PWA citados por Biørn-Hansen e Majchrzak (2017), é ser compartilhável, descobrível e envolvente (seção 2.1.6). Desta forma, assim como no *manifest.json* o HTML também contém metadados que, segundo o autor são descritos como *metatags*, que são *tags* adicionadas na página afim de descrever o objetivo da aplicação, assim como representar o conteúdo da aplicação em cada uma das páginas, este conjunto de informações são lidos pelos buscadores, redes sociais e outras aplicações, gerando *links* mais auto explicativos e melhorando o ranqueamento nos buscadores.

Figura 30 - Links com metatags descritivas compartilhando pelo Facebook



Fonte: Autor, 2021.

Figura 31 - Links com metatags descritivas compartilhando pelo WhatsApp



Fonte: Autor, 2021.

Nas Figuras 30 e 31 tem-se a resposta da aplicação no compartilhamento do Facebook e no envio de mensagens no WhatsApp, é notada a presença de uma imagem representando a página em questão, o nome da página, e uma breve descrição, dados que podem variar de acordo com o conteúdo da página, sendo assim comportam-se como uma previa do conteúdo a ser mostrado ao acessar o *link*, os mesmos dados que apareceriam no resultado de busca são exemplificados na Figura 10 (pág. 27).

Figura 32 - Componente voltado aos metadados das páginas

```
<meta charSet="utf-8" />
<meta httpEquiv="X-UA-Compatible" content="IE=edge" />
<meta
  name="google-site-verification"
  content="imEmJjnY8LRB_gCyQpOHG1E6dLgt7_SySHboWMsERho"
/>
<meta name="description" content={description}></meta>
<meta
  name="viewport"
  content="width=device-width,initial-scale=1,minimum-scale=1,maximum-scale=5"
/>
<meta name="theme-color" content="#003B36"></meta>

<link rel="manifest" href="/manifest.json?v=5" />
<link rel="icon" href="/favicon.ico?v=5" />

{/* Apple Tags*/}
<link rel="apple-touch-icon" href="/icons/icon192.png?v2" />
<meta name="apple-mobile-web-app-capable" content="yes"></meta>
<meta name="apple-mobile-web-app-status-bar-style" content="black"></meta>

{/* Open Grafh Tags */}
<meta name="og:type" property="og:type" content="website" />
<meta name="og:title" property="og:title" content={title} />
<meta
  name="og:description"
  property="og:description"
  content={description}
/>
<meta name="og:site_name" property="og:site_name" content={siteName} />
<meta name="og:url" property="og:url" content="https://reporti.pictu.one/" />
<meta name="og:image" property="og:image" content={imagemUrl} />

{/* Twitter Tags */}
<meta name="twitter:card" content="summary" />
<meta name="twitter:title" content={title} />
<meta name="twitter:description" content={description} />
<meta name="twitter:site" content={siteName} />
<meta name="twitter:creator" content="Guilherme Maccali" />
<meta name="twitter:image" content={imagemUrl} />
```

Fonte: Autor, 2021.

Na Figura 32 pode-se observar o uso de algumas *tag links* que importam o arquivo de manifesto citado anteriormente, o que é obrigatório para o pleno funcionamento da aplicação proposta.

Para possibilitar uma melhor integração com estas *metatags* fora criado um componente (Figura 32), que visa receber os dados base da página e aplicar os dados nestas *metatags*, facilitando o processo descrito nessa seção.

Segundo Stivala e Pellegrino (2020), estas *metatags* possibilitam um melhor engajamento através das buscas, podendo ter um resultado muito satisfatório também em redes sociais, deixando o *link* mais atraente e passando maior confiança e segurança aos usuários de redes sociais.

Na Tabela 2 Stivala e Pellegrino (2020) descrevem a função das principais *metatags* para criação dos *previews* dos *links* compartilhados, sendo elas as responsáveis pelo título, imagem e descrição que serão mostradas.

Tabela 2 - Descrição das metatags para criação de link previews

Open Graph	Twitter Cards	Description
`og:title`	`twitter:title`	The title of the article without any branding
`og:description`	`twitter:description`	A brief description of the content.
`og:image`	`twitter:image`	The URL of the image that appears in the preview.
`og:url`	-	The canonical URL for the page, without session variables or user identifying parameters. This URL is used to aggregate likes and shares.

Fonte: Stivala e Pellegrino, 2020.

5.3.3 Telas da aplicação

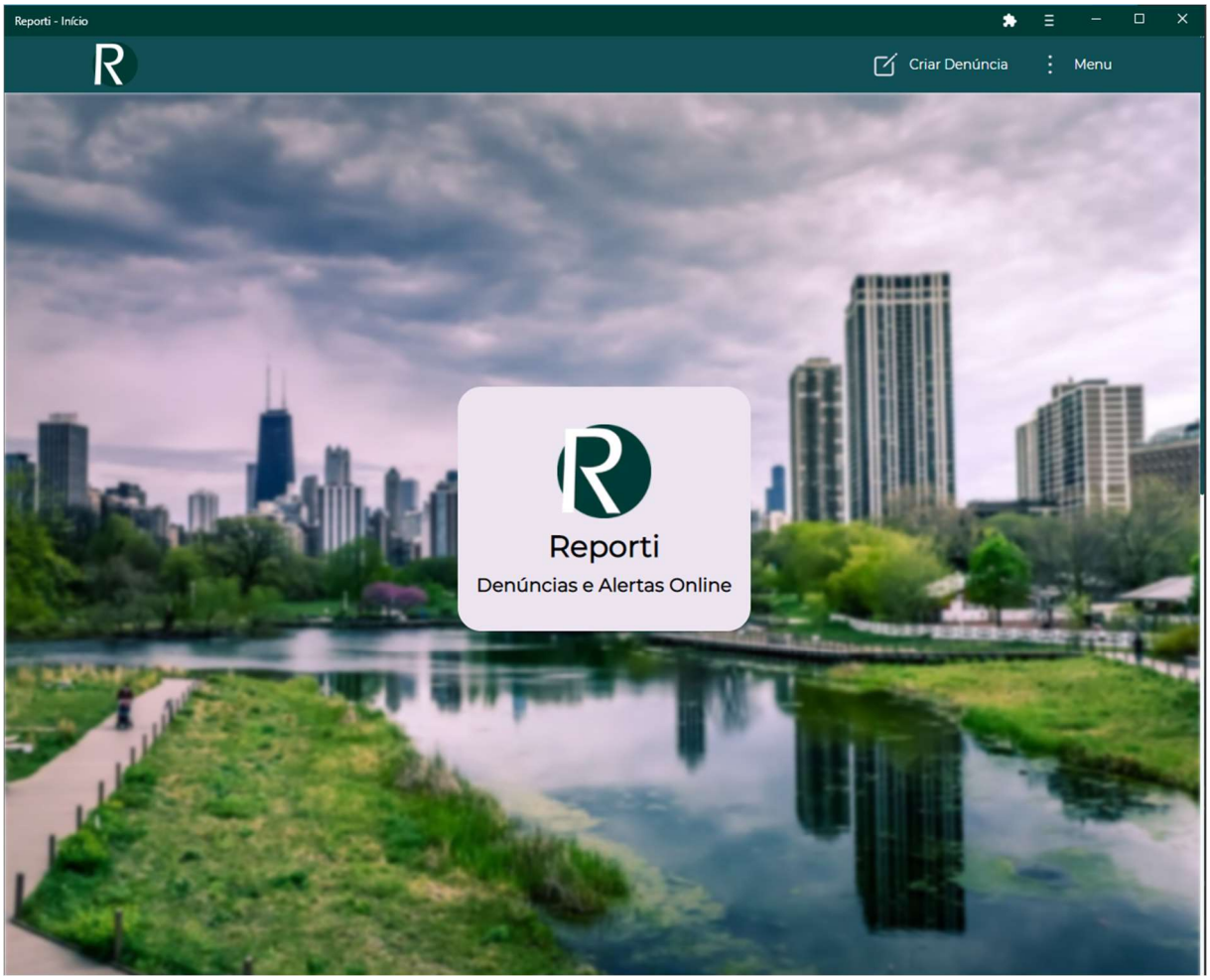
Nas seções anteriores foram mostrados os casos de uso, requisitos e as configurações para que a aplicação possa desempenhar sua função da forma proposta. Nesta seção serão mostradas as telas da aplicação assim como a explicação do seu funcionamento.

Segundo França (2015) a responsividade é imprescindível nos dias atuais e segundo Ceconi (2018) a responsividade é um ponto importante para sustentar uma das bases de uma PWA que é chamado de *app-like*. Ao longo das representações das telas do sistema, levando em consideração a importância desse ponto, serão mostradas as telas tanto para *desktop* quanto para *mobile*.

5.3.3.1 Comportamento e responsividade

Ao clicar em um dos ícones descritos na Figura 26 ou acessado o endereço disponível na *web* para a aplicação, a mesma é executada, trazendo a sua página inicial, representada nas Figuras 33 e 34.

Figura 33 - Página inicial da aplicação no desktop

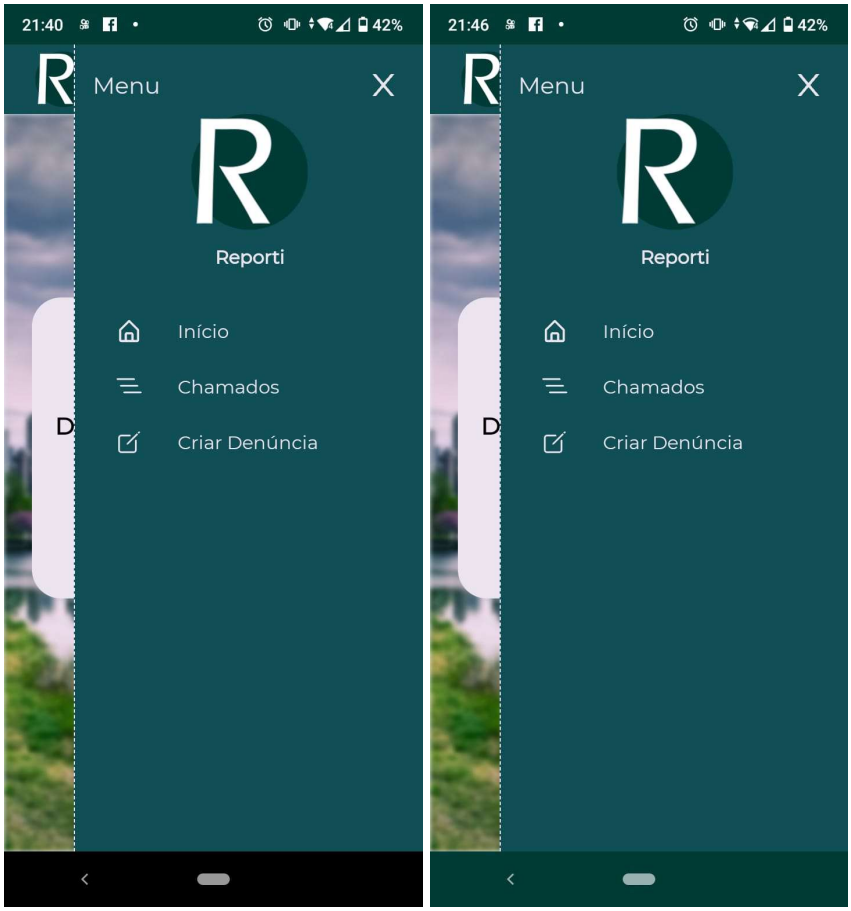


Fonte: Autor, 2021.

Figura 34 - Página inicial da aplicação no mobile

Fonte: Autor, 2021.

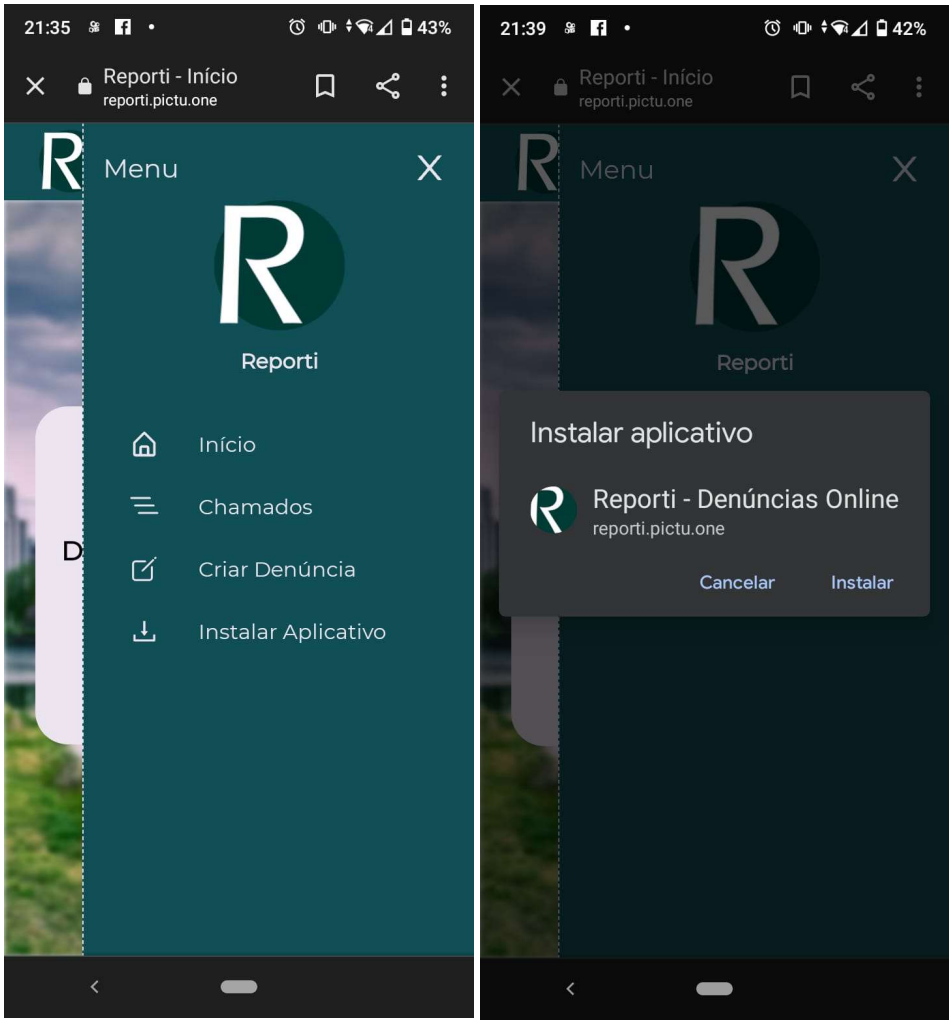
Figura 35 - Menu da aplicação mobile



Fonte: Autor, 2021.

Na Figura 35 pode-se ver a aplicação instalada via APK a direita e a PWA a esquerda, com isso nota-se que não há grandes diferenças além da cor do menu inferior, dada com o intuito de diferenciar as 2 versões.

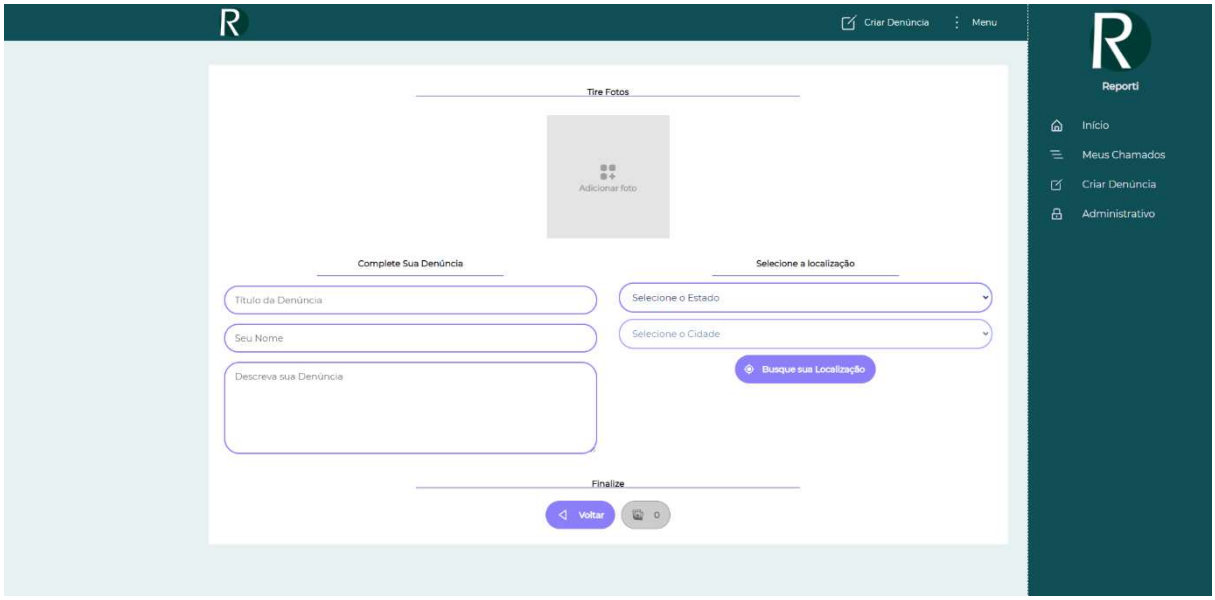
Figura 36 - Aplicação no navegador



Fonte: Autor, 2021.

A Figura 36 mostra a presença do item “Instalar Aplicativo” no menu, que aparece quando a aplicação não encontra-se instalada no dispositivo, possibilitando o usuário de instalá-la caso ache necessário, tendo assim uma experiência mais próxima de um APP nativo como visto na Figura 35.

Figura 37 - Menu da aplicação em telas grandes



Fonte: Autor, 2021.

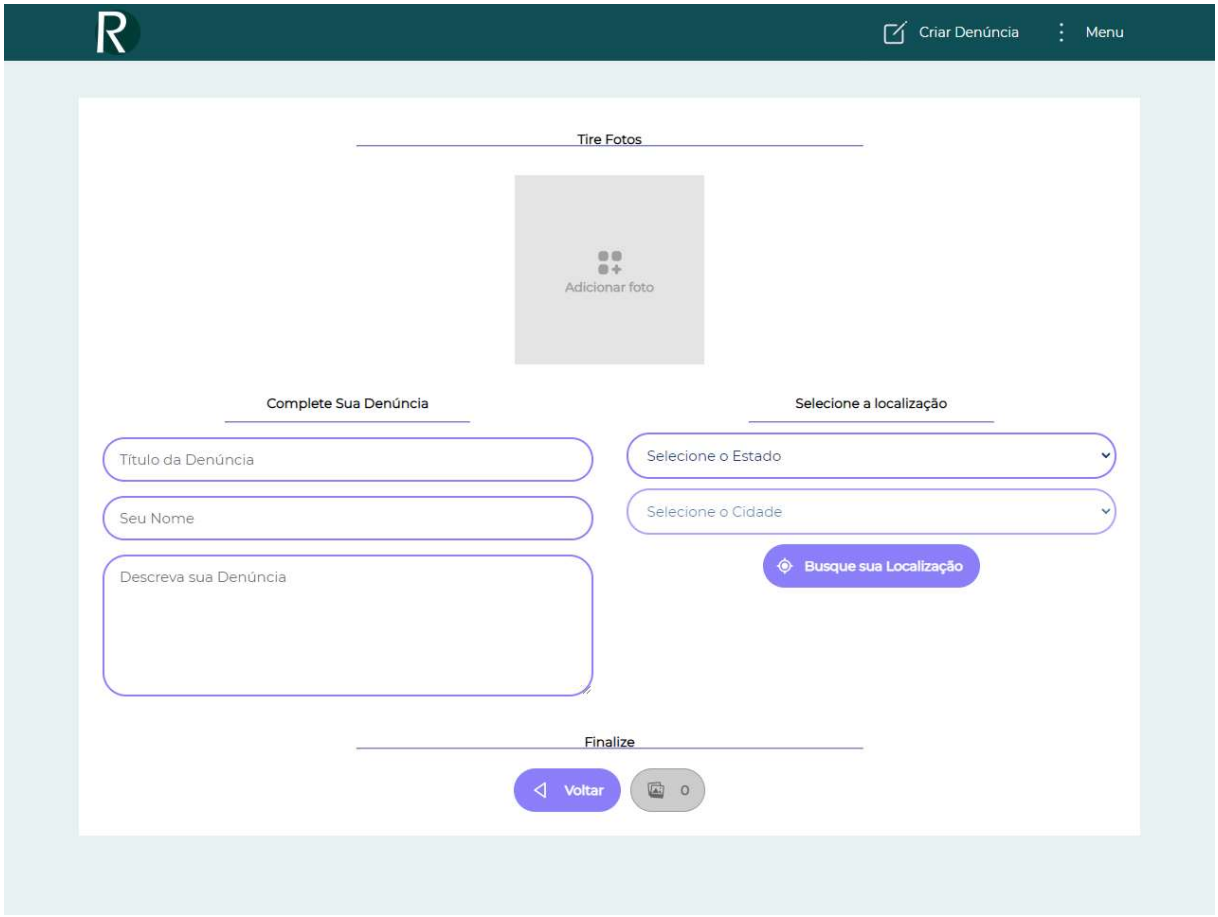
Pode-se ver nas Figuras 34, 35, 36 e 37 o comportamento do menu da aplicação, onde tem-se um botão que dá destino para criação da denúncia e outro do menu, que abre a barra com o menu completo, na versão *mobile* pode-se notar a presença de ícones para o acesso a criação de denúncia, e outro para o menu, já na versão *desktop* estes botões mantêm o texto que os representa.

Outro comportamento adotado para a aplicação proposta é manter o menu recolhido para telas pequenas e mantê-lo aberto apenas em telas maiores, tendo assim um comportamento voltado a estratégia *app-like*.

5.3.3.2 Criação da denúncia

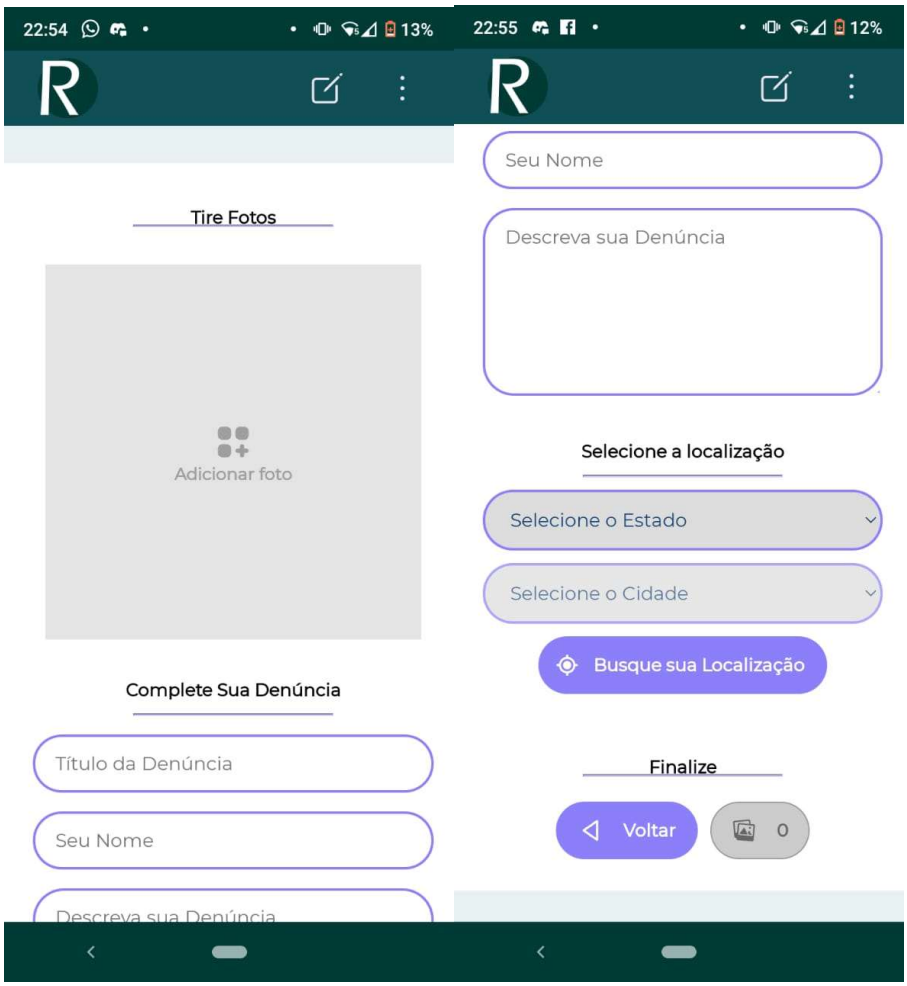
Na Figura 38 e 39 podemos ver a tela de criar denúncia na versão *desktop* e no *mobile* respectivamente, esta tela é a que permite o denunciante cadastrar a denúncia que, posteriormente, será analisada pela autoridade correspondente.

Figura 38 - Criação da denúncia no desktop



Fonte: Autor, 2021.

Figura 39 - Criação da denúncia no mobile



Fonte: Autor, 2021.

Neste formulário o denunciante tem a possibilidade de enviar imagens caso esteja em uma plataforma *desktop,* ou tirar fotos caso esteja em uma plataforma *mobile*, para isso, ao clicar em “adicionar foto” tem-se um novo quadro que traz a possibilidade de clicar no ícone da câmera e completar esta ação, podendo o denunciante adicionar ou remover mais fotos até um limite de 4, descrito nos requisitos.

Posteriormente na segunda seção denominada “Complete sua Denúncia” é possível a adição de algumas informações como: título da denúncia, nome do denunciante, e uma descrição da mesma.

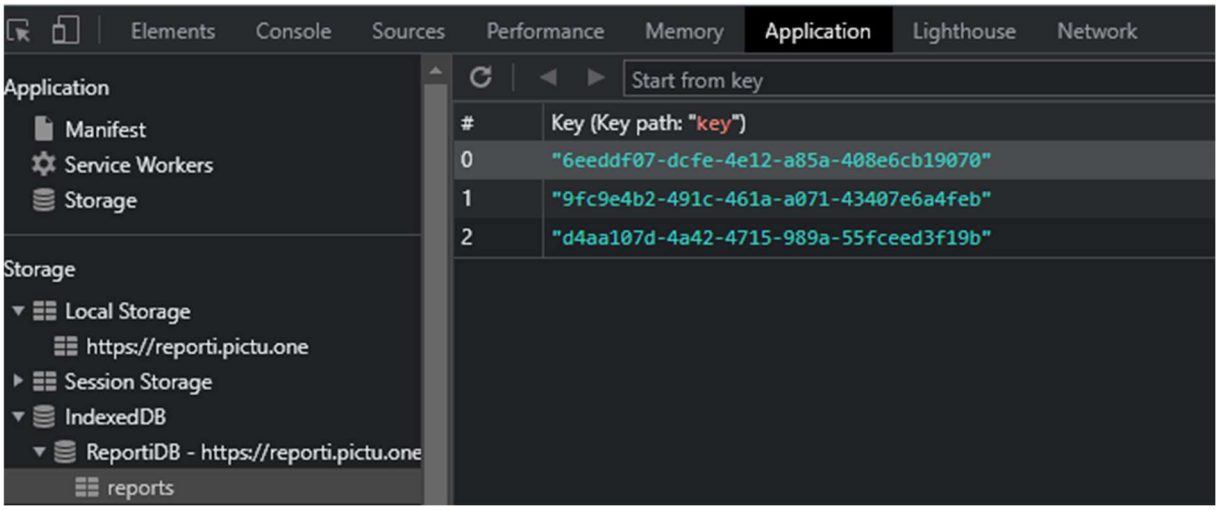
Na terceira seção intitulada como “Selecione a Localização” é possível selecionar a localização através do botão “Busque sua Localização”, ou através dos *selects* de estado e cidade.

Ao buscar pelo botão, o mesmo pedirá acesso a localização do dispositivo, com isso tem-se as coordenadas para a busca do local, que é feita em uma API terceira, ao identificar a cidade e estado desta coordenada, o sistema completa automaticamente os *selects* de cidade e estado previamente populados através de buscas na API do Instituto Brasileiro de Geografia e Estatística (IBGE).

Após o completo preenchimento dos campos obrigatórios e a seleção de ao menos uma imagem, pode-se completar o envio do formulário pressionando o botão de “Enviar”.

Na submissão do formulário, primeiramente são enviados os arquivos que passam por um processo de redução de tamanho para *backend* da aplicação, onde os arquivos são direcionados para o S3 da Amazon que retorna o *link* de cada imagem salva, montando um *array* de *links* que será adicionado na postagem da denúncia.

Figura 40 - Representação de dados na tabela do IndexedDB

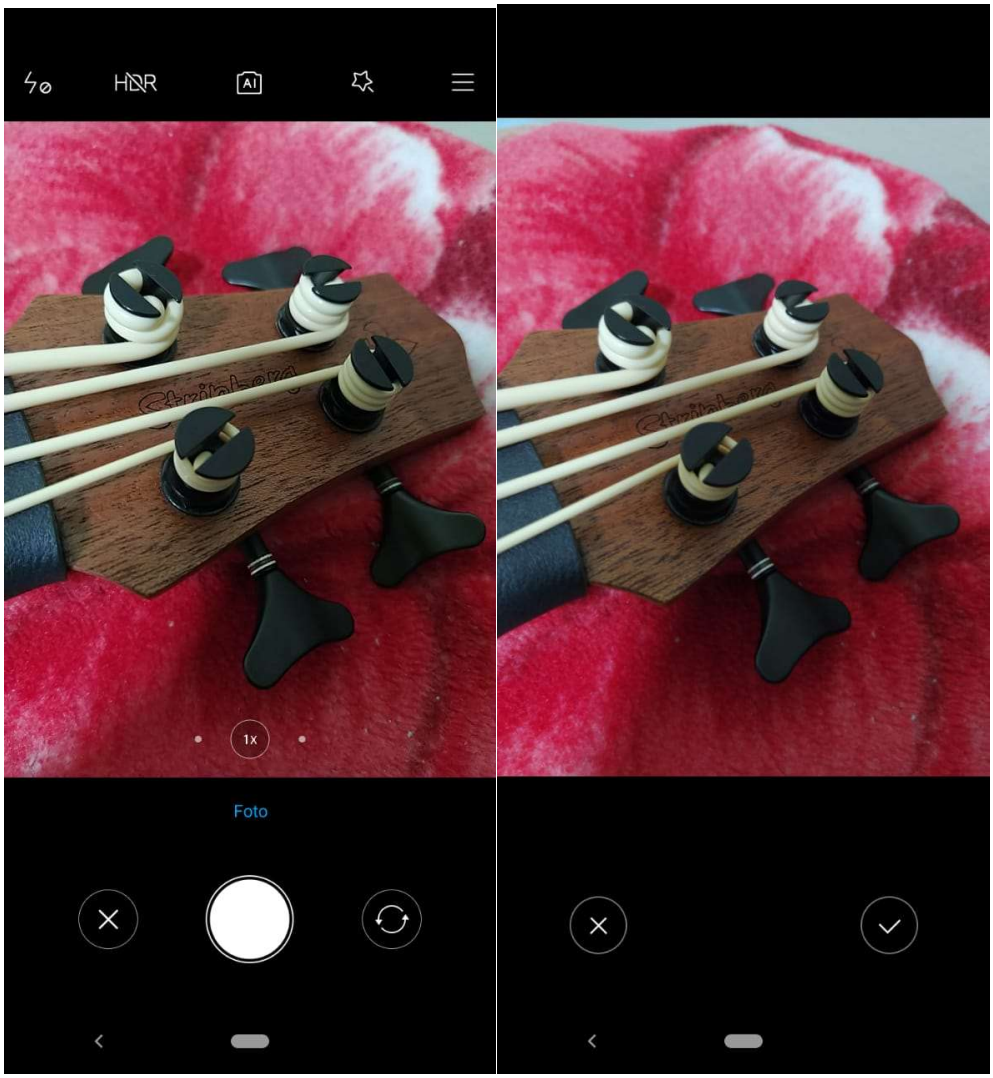


Fonte: Autor, 2021.

Com os *links* coletados e adicionados na denúncia, a mesma é enviada e armazenada pelo *backend* que retorna um objeto com os dados cadastrados. A PWA ao receber um retorno positivo, grava estes dados no *IndexedDB* do navegador (Figura 40), salvando o seu identificador para a posterior busca em “Meus chamados”.

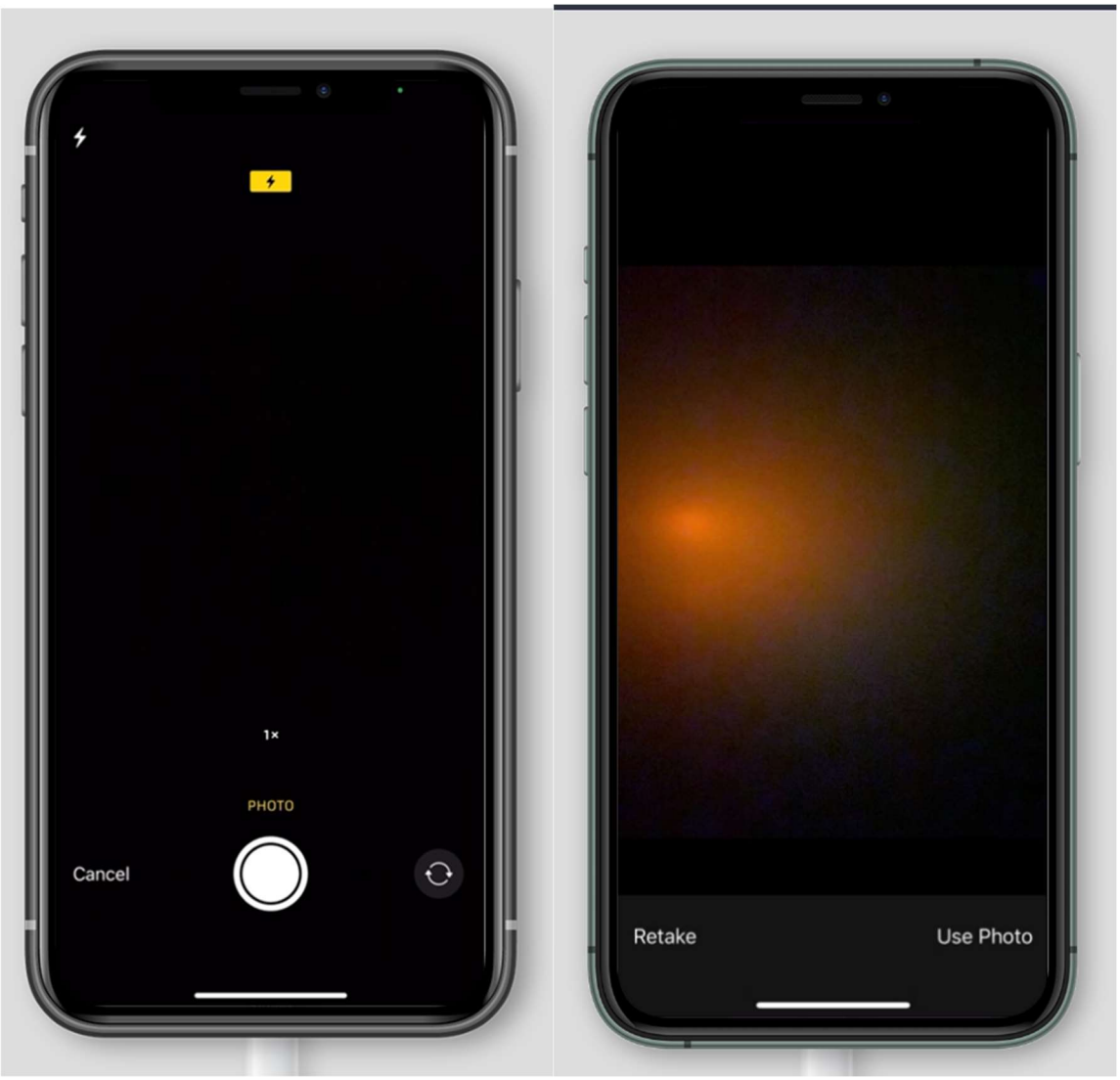
Nas Figuras 41, 42 e 43 tem-se o resultado da ação de clique ou toque no botão da câmera, que irá abrir preferencialmente como representado nas Figuras 41 e 42 se o dispositivo estiver com a câmera ativada, caso o mesmo não possuir câmeras ou as câmeras estiverem desabilitadas, a aplicação mostrará o explorador de imagens, conforme a Figura 43.

Figura 41 - Envio de foto via câmera



Fonte: Autor, 2021.

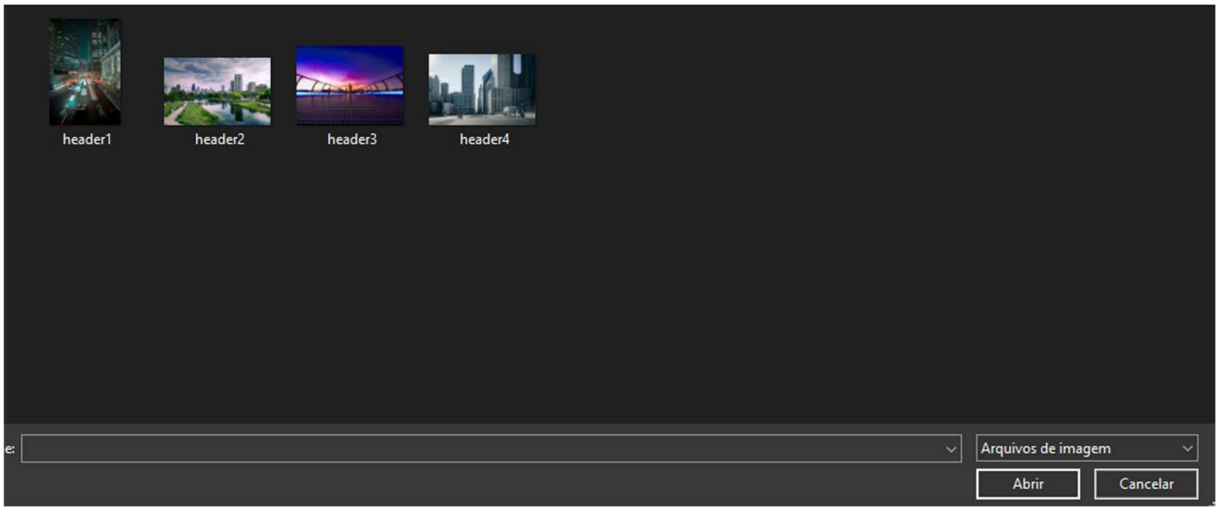
Figura 42 - Envio de foto via câmera (iPhone)



Fonte: Autor, 2021.

Nas Figuras 41 e 42 ao lado esquerdo, pode-se ver o aplicativo utilizando o APP nativo da câmera disponível no celular para tirar uma foto, já ao lado direito é possível ver que após o usuário tirar a foto o aplicativo nativo oferece um *preview* da foto tirada, onde usuário poderá descarta-la ou confirma-la para o envio da mesma para a aplicação. As plataformas Android e iOS em que foram feitos os testes tiveram um comportamento semelhante.

Figura 43 - Envio de imagem sem câmera

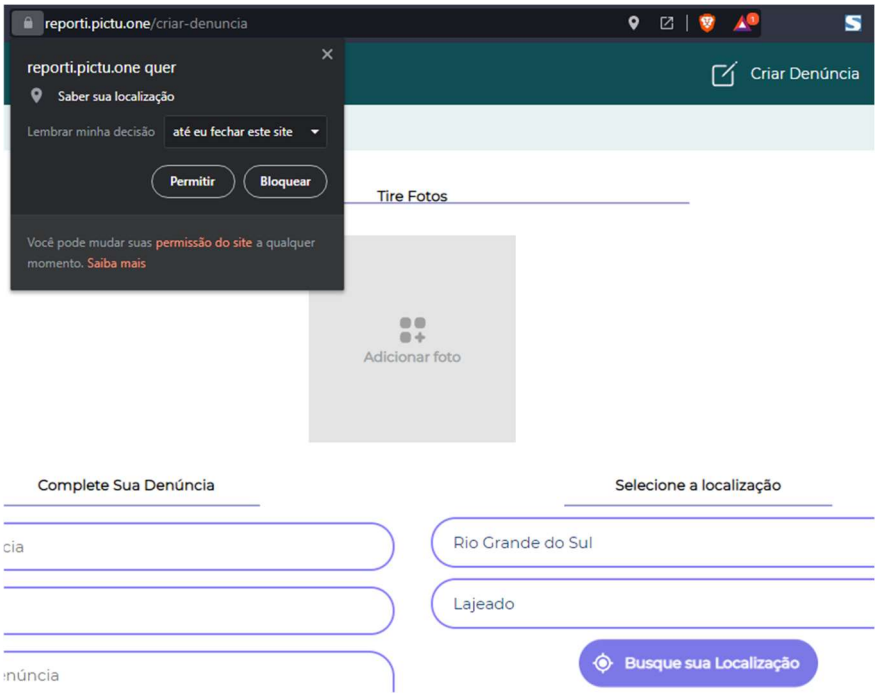


Fonte: Autor, 2021.

Como mencionado por Tandel e Jamadar (2018) e Ceconi (2018), é uma boa prática trazer ao usuário uma segunda opção de preenchimento de dados caso não seja concedido acesso ao recurso nativo. Os principais casos para representar este aspecto na aplicação proposta é o uso da câmera e geolocalização, pois estes meios alternativos de fluxo também fazem parte do desenvolvimento progressivo segundo Ceconi (2018).

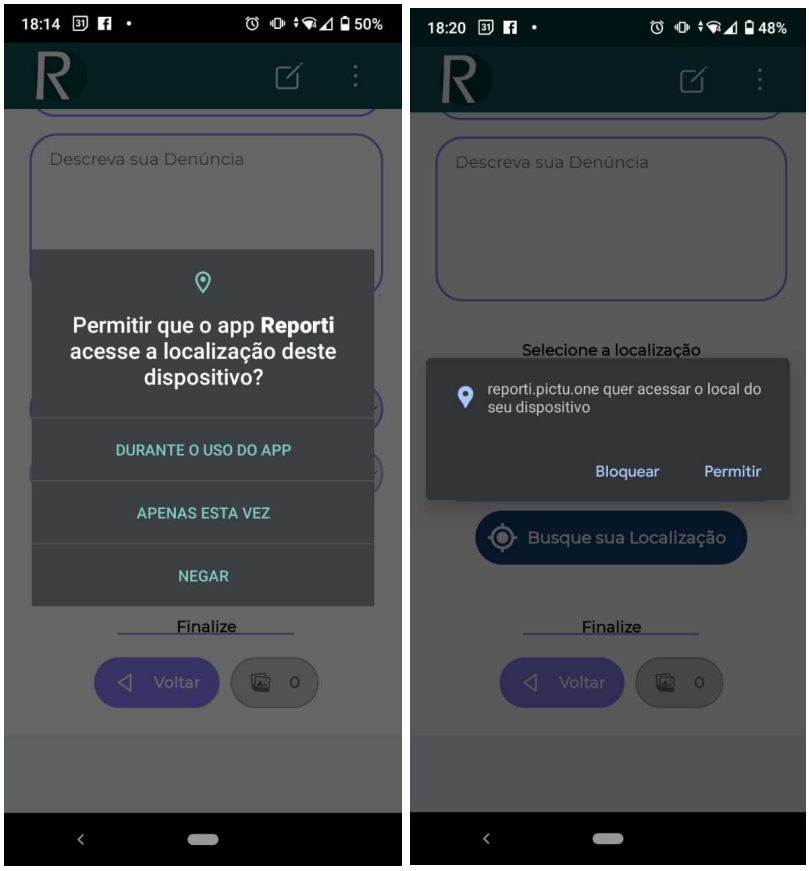
Para a utilização dos componentes de *hardware* do dispositivo é necessário que o dispositivo tenha o recurso a ser solicitado, e que o usuário conceda a permissão de uso deste recurso, como pode-se ver nas Figuras 44 e 45.

Figura 44 - Pedido de permissão via browser no desktop



Fonte: Autor, 2021.

Figura 45 - Pedido de permissão via APP e PWA

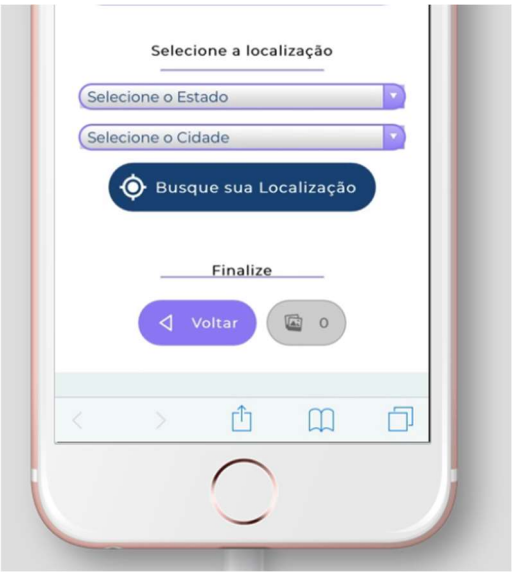


Fonte: Autor, 2021.

As Figuras 44 e 45 representam o sistema pedindo a permissão do usuário para o acesso a localização do dispositivo, após concedida ou não a resposta fica gravada, sendo lembrada nas próximas seções e podendo ser alterada por meio de alguns comandos no navegador.

Na Figura 46 pode-se ver um exemplo de problemas que podem ocorrer entre *browsers* mencionados por Sheppard (2017), a Figura 46 mostra os *selects* de localização no navegador Safari do iOS, anomalia que pode ocorrer pela maneira em que o *browser* interpreta as *tags* HTML ou CSS da aplicação.

Figura 46 - Anomalias entre browsers



Fonte: Autor, 2021.

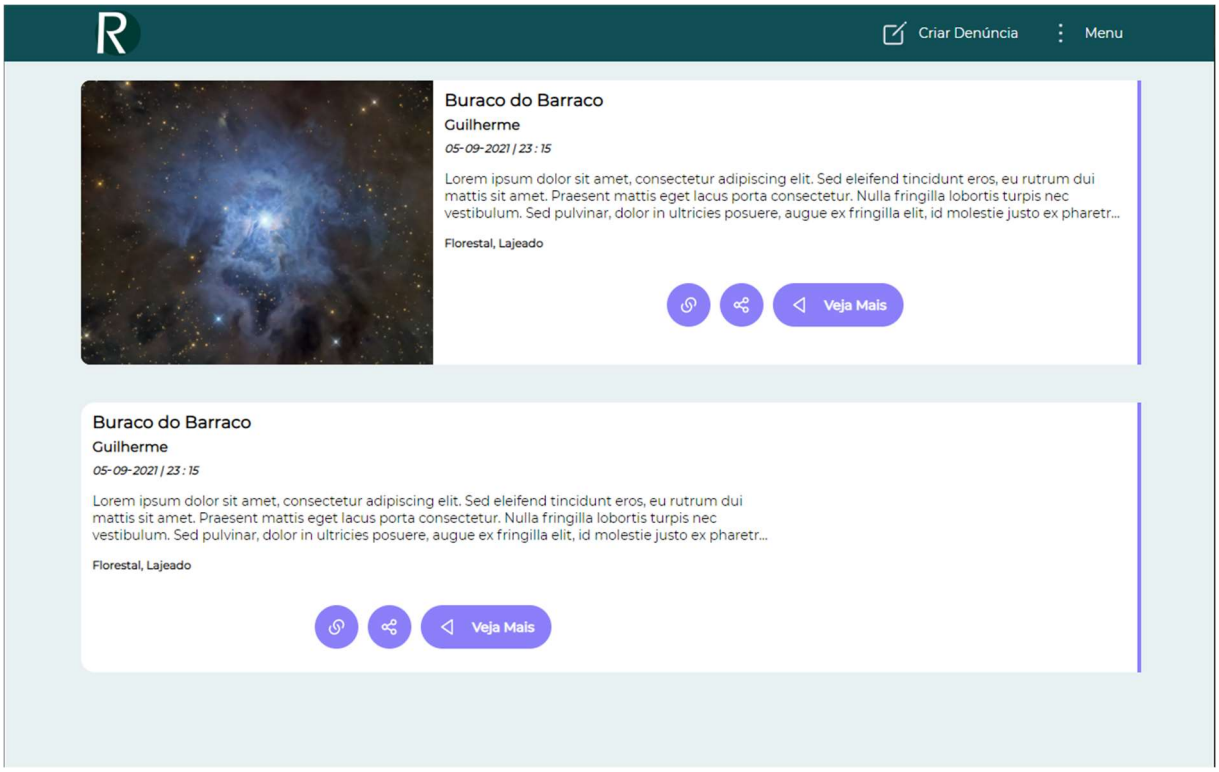
5.3.3.3 Listagem das denúncias feitas

Nas Figuras 47 e 48 tem-se a página de listagem das denúncias feitas nos dispositivos em questão, onde cada denúncia traz suas informações básicas como: imagem, nome do denunciante, data da denúncia, as 3 primeiras linhas da descrição, o local e 3 botões.

O primeiro botão copia o *link* para a área de transferência podendo ser colado em outro local desejado, o segundo abre a *share api* uma API nativa do navegador

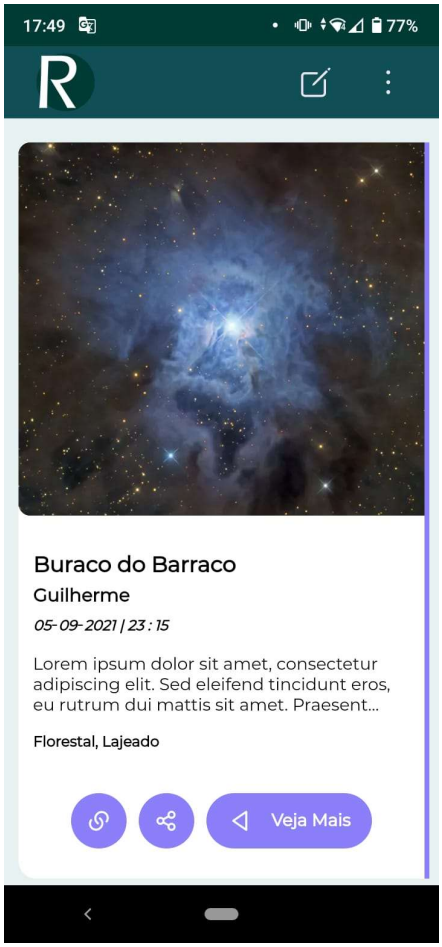
que fornece opções de compartilhamento avançadas, e o terceiro abre uma modal com as informações completas da denúncia.

Figura 47 - Visualização das denúncias feitas pelo usuário no desktop



Fonte: Autor, 2021.

Figura 48 - Visualização das denúncias feitas pelo usuário no mobile

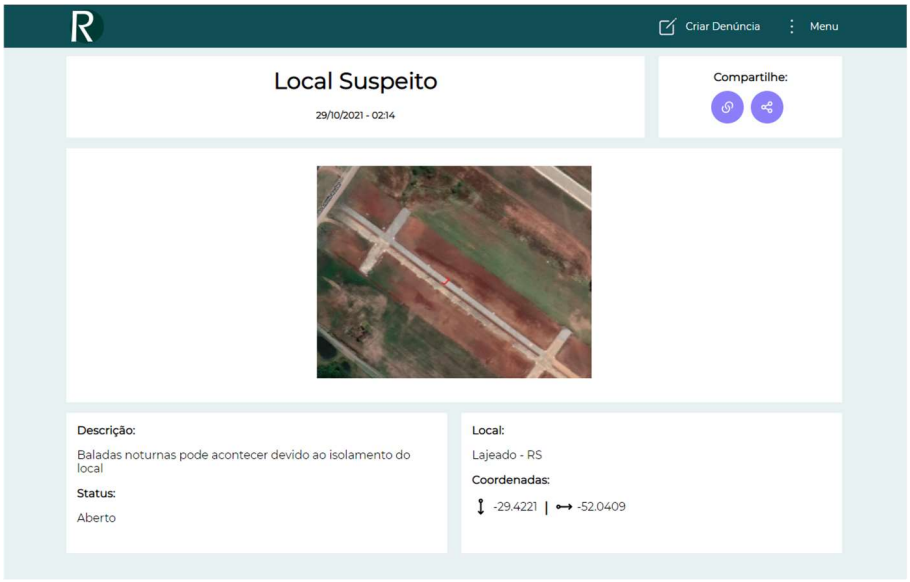


Fonte: Autor, 2021.

5.3.3.4 Página da denúncia

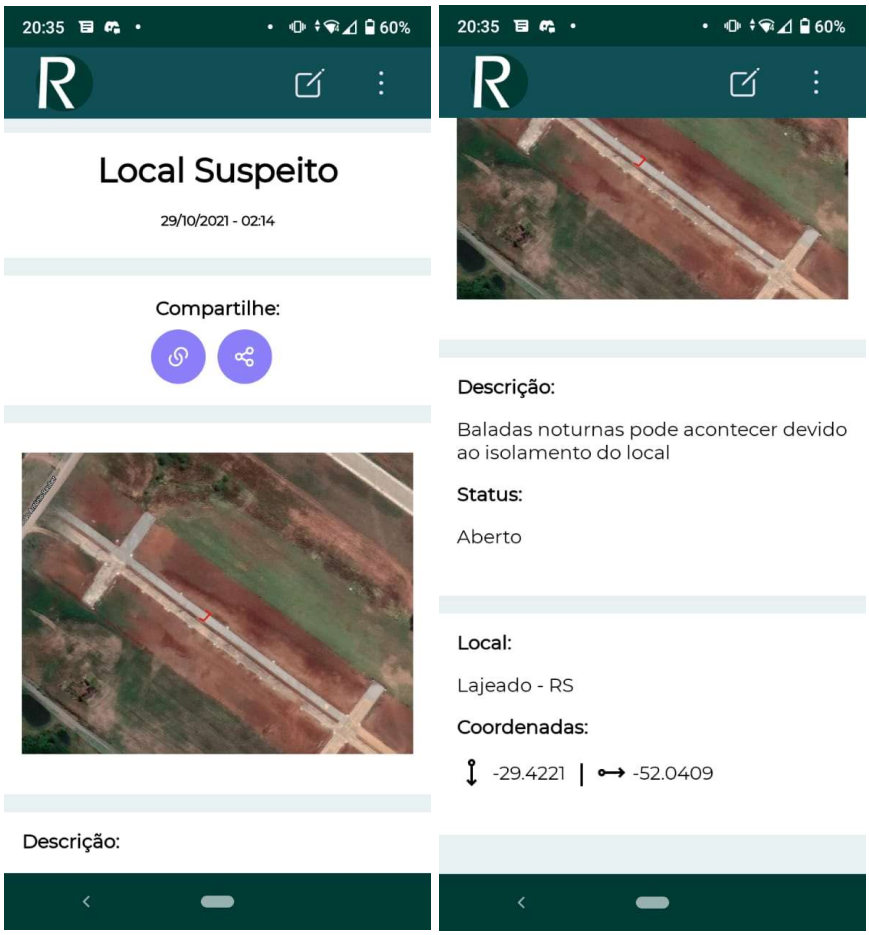
Após a criação da denúncia, a aplicação proposta conta com uma página compartilhável para o acompanhamento da denúncia onde fornece todos os dados cadastrados nela assim como o andamento da denúncia.

Figura 49 - Página da denúncia no desktop



Fonte: Autor, 2021.

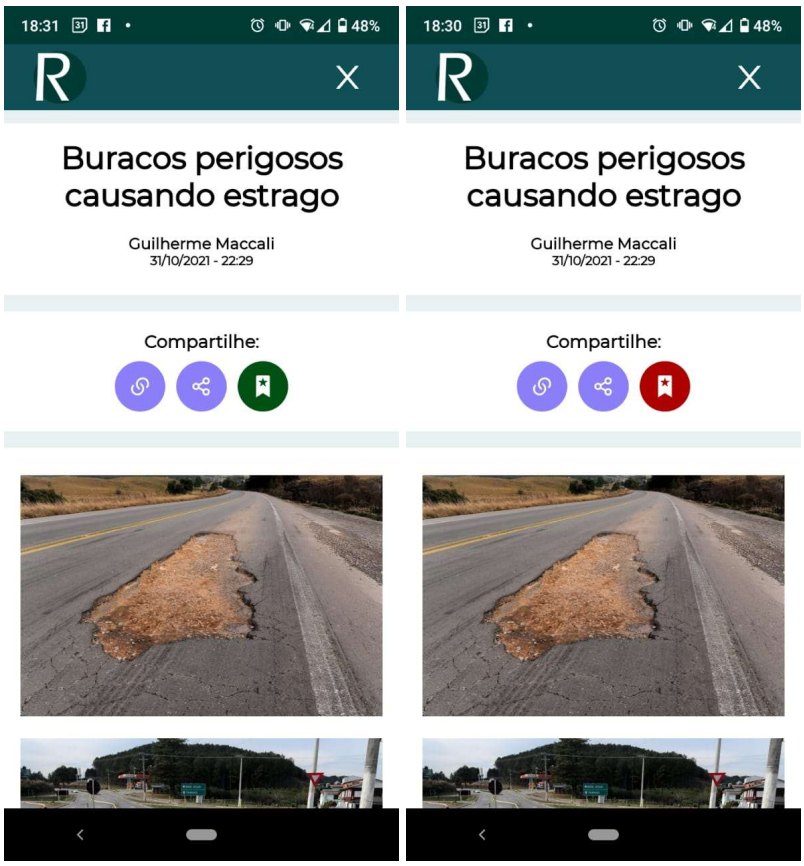
Figura 50 - Página da denúncia no Mobile



Fonte: Autor, 2021.

As Figuras 49, 50, 51 e 52 mostram a página com os dados da denúncia no *desktop* e no *mobile* respectivamente, onde no primeiro bloco é mostrado o título dado a denúncia, o nome do denunciante, assim como a data e hora que foi cadastrada.

Figura 51 - Modal da página da denúncia



Fonte: Autor, 2021.

A Figura 51 também demonstra a página sendo aberta como um modal, semelhante a um APP nativo sendo acessada desta forma através dos botões “saiba mais” representados nas Figuras 47 e 48.

Ainda na Figura 51 pode-se ver que ao marcar a denúncia o último botão passa de verde para vermelho, assim marcado, o usuário poderá acompanhar a denúncia através do menu de chamados, ou mesmo remover a denúncia do menu de chamados.

A Figura 52 mostra a página da denúncia aberta no iOS, não sendo notada diferenças perceptiveis no *layout* comparada as outras plataformas testadas neste estudo.

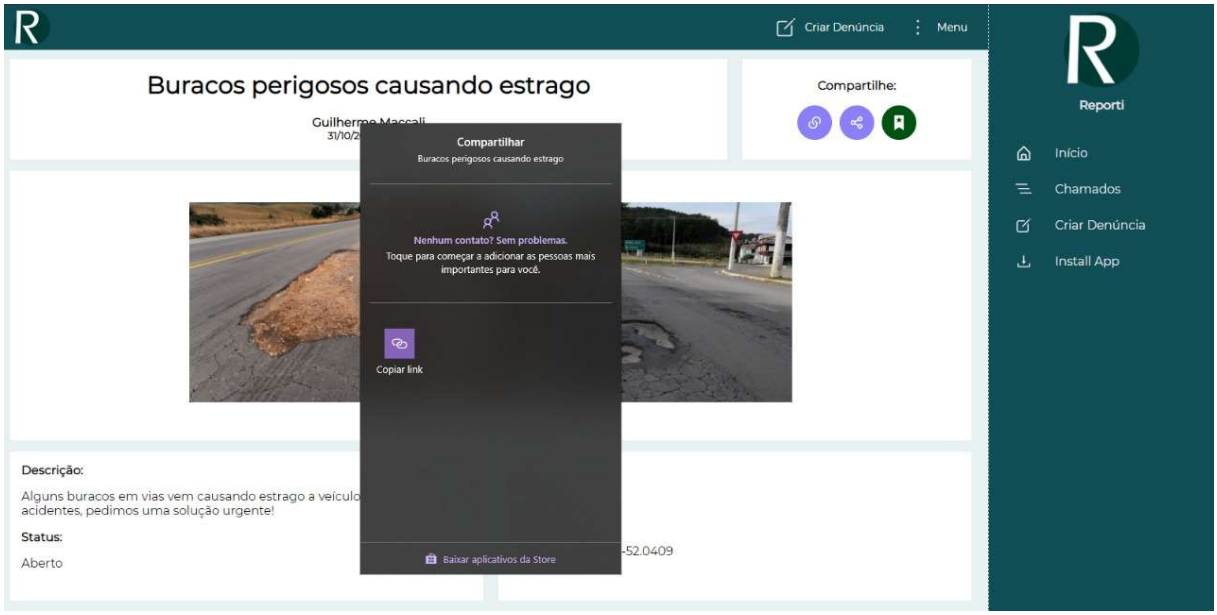
Figura 52 - Página da denúncia no simulador do iPhone

Fonte: Autor, 2021.

No segundo bloco da página de denúncia tem-se a sessão de compartilhamento e interação, onde o primeiro botão copia o *link* para a área de transferência, o segundo botão abre o compartilhamento via *share api* Figuras 53 e 55 e o terceiro botão salva a denuncia localmente via *IndexedDB*.

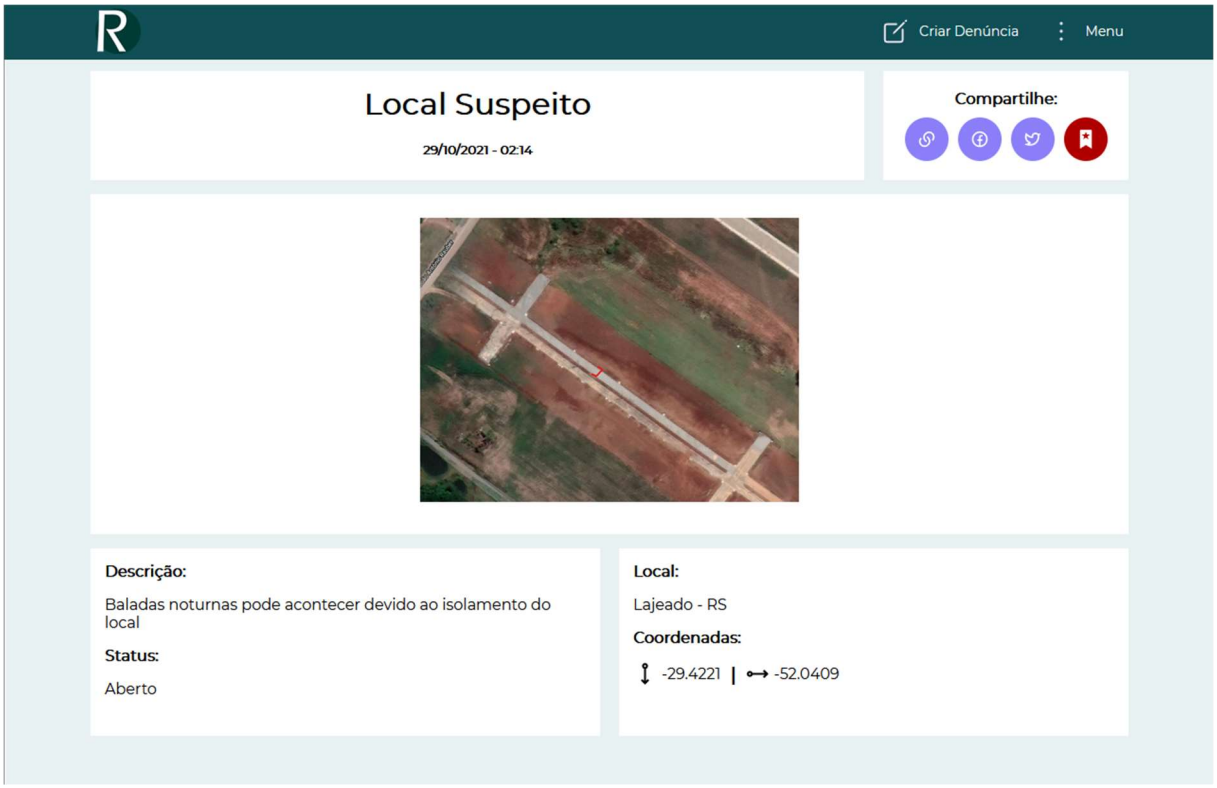
Como o navegador Firefox não conta com a *share api* até o momento deste estudo, o mesmo recebeu um compartilhamento próprio para as principais redes sociais (Facebook e Twitter) representado na Figura 54.

Figura 53 - Share API da página no desktop



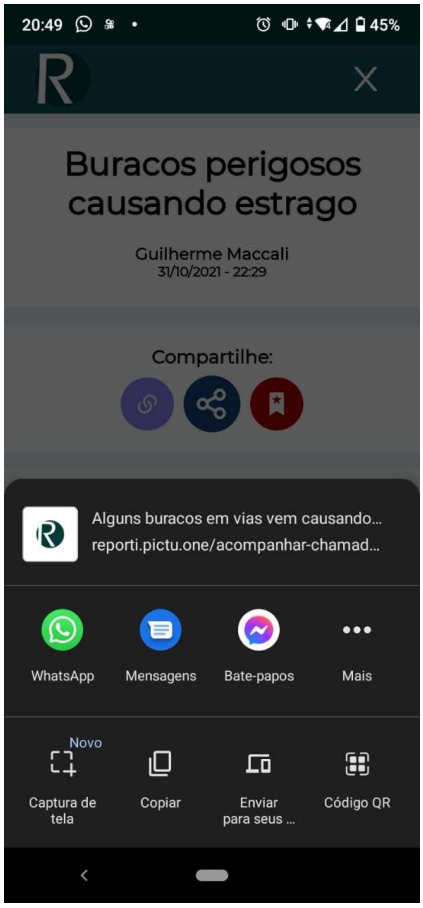
Fonte: Autor, 2021.

Figura 54 - Share API da página no desktop (Firefox)



Fonte: Autor, 2021.

Figura 55 - Share API da página no mobile



Fonte: Autor, 2021.

Ao topo do modal que é mostrado ao pressionar o botão pode-se ver na Figura 55, um breve texto acompanhado do *link* da página, e logo abaixo 2 botões, um para copiar, que copiará o texto com o *link* e outro para o compartilhamento dessa informação via recurso de proximidade.

Pode haver uma diferença na segunda sessão do modal, que pode mostrar recomendações de contatos existentes no dispositivo, que possuam a possibilidade de receber o *link* compartilhável.

A terceira seção do modal mostra os APPs mais utilizados para o compartilhamento no dispositivo, e a quarta seção do modal com os demais aplicativos. Nota-se que o recurso visa a rapidez no compartilhamento pois tenta prever onde o usuário irá compartilhar tal informação, assim também pode ser notado que seu desenvolvimento vem sendo aprimorado, tendo o recurso visualmente

evoluído durante o desenvolvimento da aplicação proposta, trazendo novas opções ao usuário.

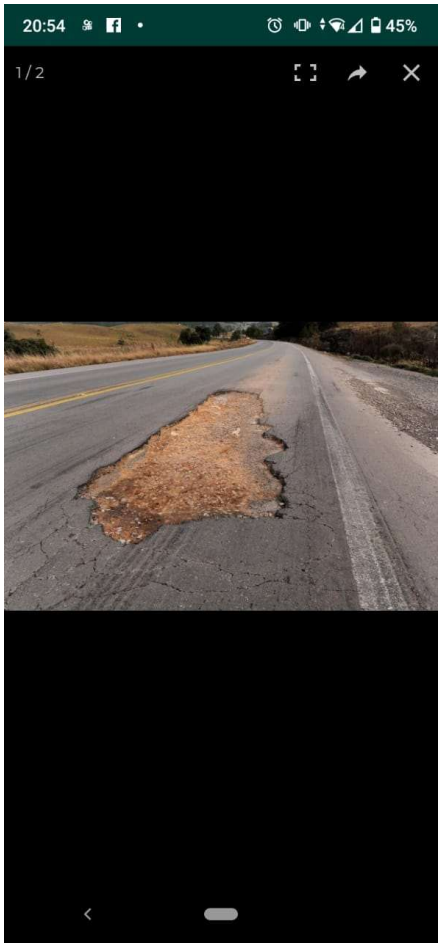
A Figura 56 representa a *share api* no dispositivo iOS, tendo muitas semelhanças com os representados na Figura 55, mudando apenas poucos elementos.

Figura 56 - Share API da página no simulador do iPhone

Fonte: Autor, 2021.

O terceiro bloco da página de denúncias mostra as imagens que foram inseridas na denúncia onde pode-se vê-las uma abaixo da outra no *mobile* (Figura 51) e em formato de *grid* do *desktop* (Figura 49), ao clicar ou pressionar uma das imagens pode-se ver a abertura de uma galeria (Figura 57), onde preenche a página e mostra a imagem ampliada, facilitando a visualização das imagens, podendo também passar de uma imagem para a outra a partir desta visualização.

Figura 57 - Galeria para melhor visualização das imagens



Fonte: Autor, 2021.

O quarto bloco da página de denúncia mostra a descrição feita pelo denunciante, assim como o status da denúncia para o seu acompanhamento, já o quinto bloco mostra o local identificado ou mencionado pelo denunciante, acompanhado das coordenadas caso elas tenham sido inseridas pelo usuário no momento do cadastro.

5.3.4 Configuração e geração da TWA

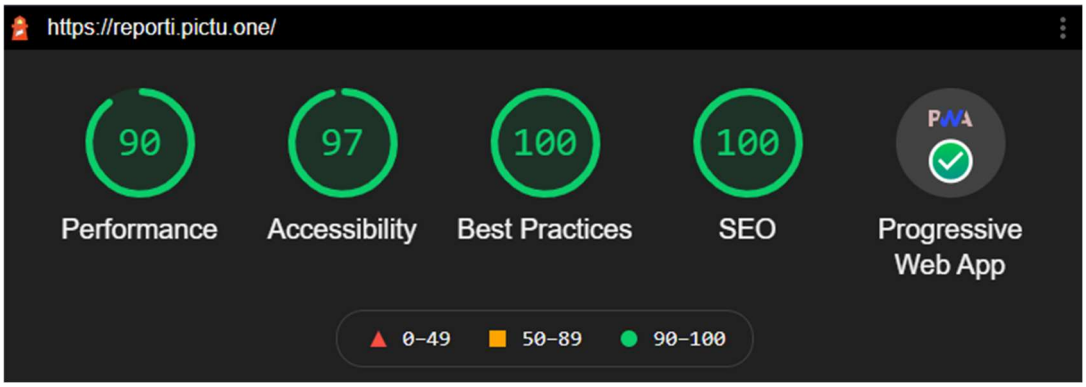
Segundo Alemu (2020), uma TWA pode ajudar os usuários a terem uma experiência ainda melhor, tornando o aplicativo ainda mais nativo, sendo possível por meio desta, a adição na loja de aplicativos da Google.

Segundo Alemu (2020), uma TWA possui algumas vantagens significativas perante a um APP nativo, sendo uma delas o *deploy*, pois não há necessidade de lançar uma nova APK na Google Play a cada atualização.

Uma TWA também resulta em APKs muito leves, beirando 2 MB, o que é imprescindível para dispositivos com pouca memória, e para uma instalação rápida. Importante salientar que uma TWA executa utilizando o navegador padrão do dispositivo estando suscetível as limitações ou regras do navegador em questão, tanto em armazenamento interno quanto acesso as APIs nativas no dispositivo.

Para que uma TWA possa ser implementada é preciso que a PWA esteja funcional. Por isso é necessário utilizar ferramentas como PWABuilder ou o Lighthouse mencionado na seção 2.3.1 para esta avaliação. Na Figura 58 pode-se ver a avaliação da aplicação proposta perante os quesitos avaliados pela ferramenta, sendo que para o presente estudo o quesito “Progressive Web Apps” é o mais relevante.

Figura 58 - Resultado da avaliação feita através do Lighthouse

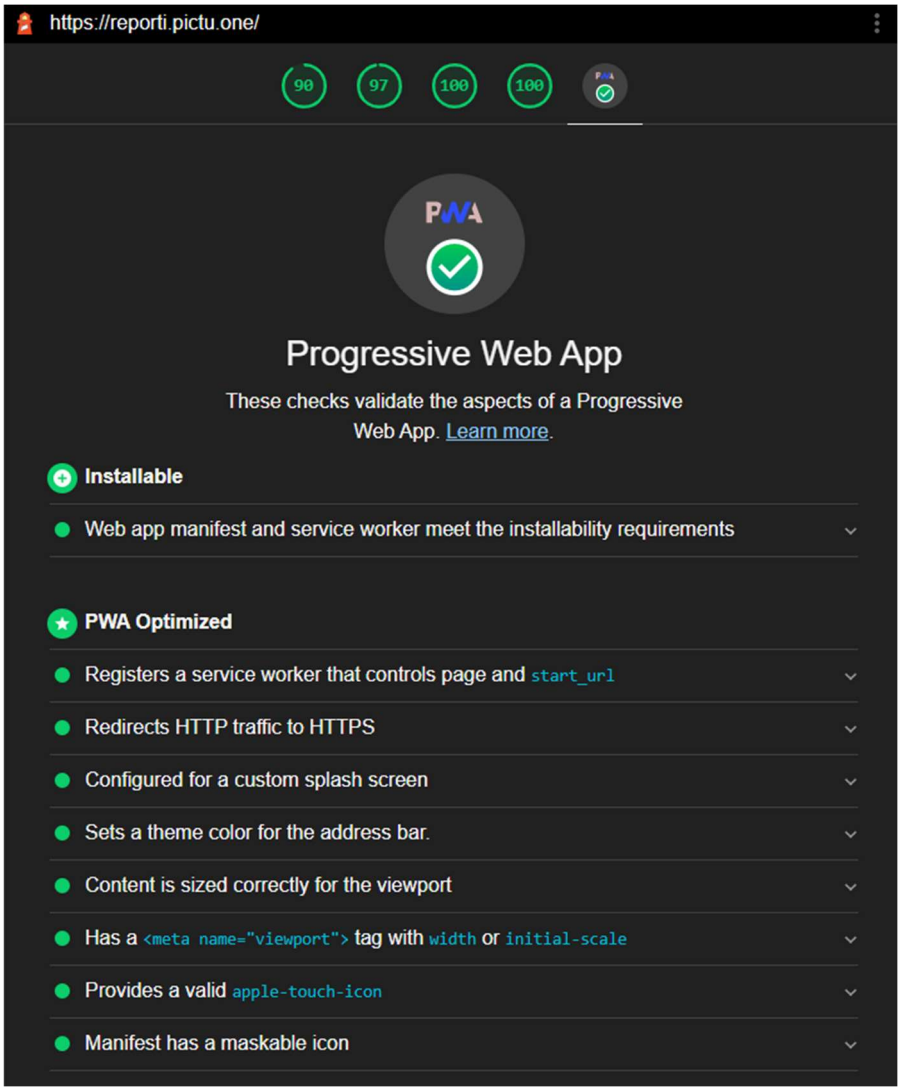


Fonte: Autor, 2021.

Por outro lado, todos estes itens são importantes para o desenvolvimento de aplicações *web* melhores, segundo Ceconi (2018), e exemplificado por Alemu (2020) com a otimização da aplicação descrita em seu trabalho.

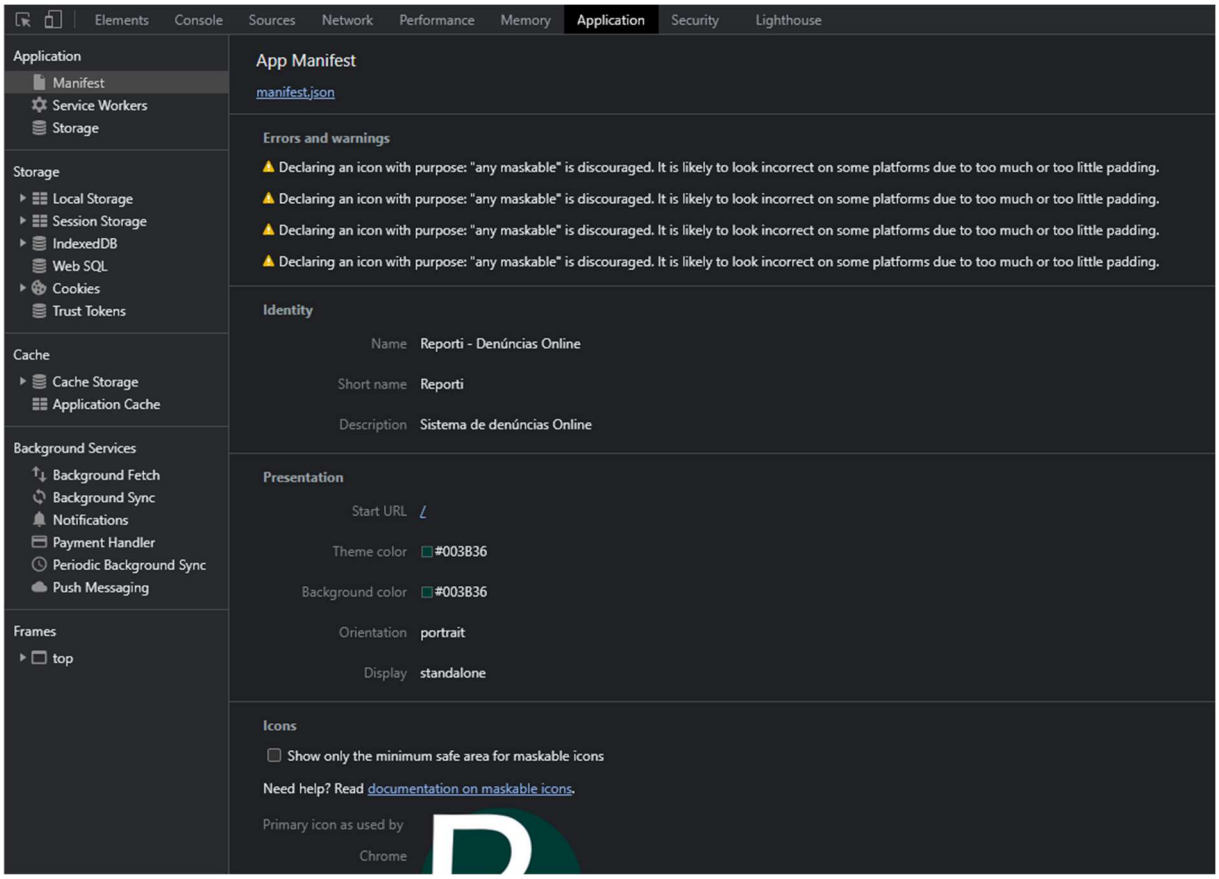
Na Figura 59 pode-se ver que a aplicação proposta atende aos requisitos de uma PWA, segundo a aplicação do Google Lighthouse. Pode-se notar alguns destes itens descritos nas Figuras 25 (pag. 72) e 32 (pag. 77), onde mostra a escrita do manifesto e das *meta tags*.

Figura 59 - Resultado da avaliação no quesito PWA pelo Lighthouse



Fonte: Autor, 2021.

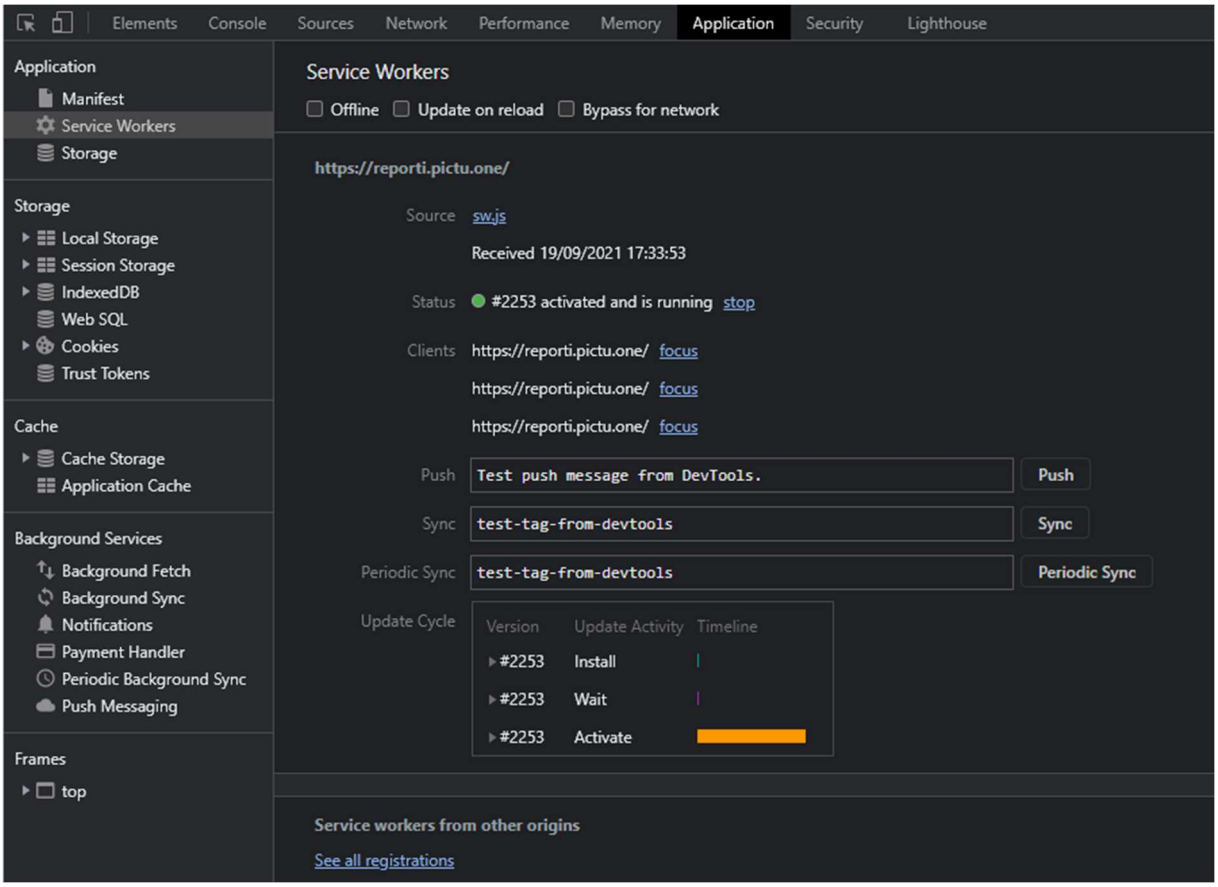
Figura 60 - Representação do manifesto no Chrome Dev Tools



Fonte: Autor, 2021.

Na Figura 60 pode-se ver a representação do manifesto da aplicação assim como todas as informações descritas no arquivo *manifest.json* onde o navegador mostra todas as atribuições contidas no arquivo, assim como erros e advertências que podem ocorrer, servindo também de auxílio para o seu desenvolvimento.

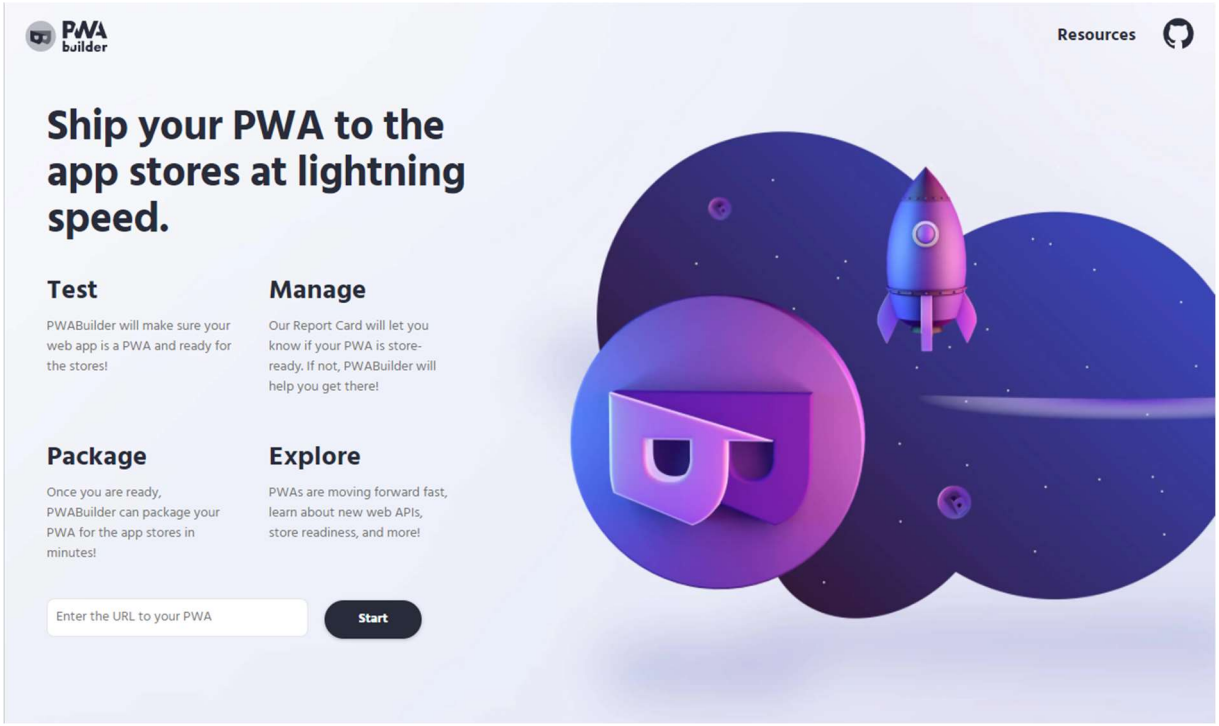
Figura 61 - Representação do Service Worker no Chrome Dev Tools



Fonte: Autor, 2021.

Na Figura 61 pode-se analisar o SW que está em execução na aplicação, mostrando alguns dados como, status, data da inserção, assim como o ciclo de vida.

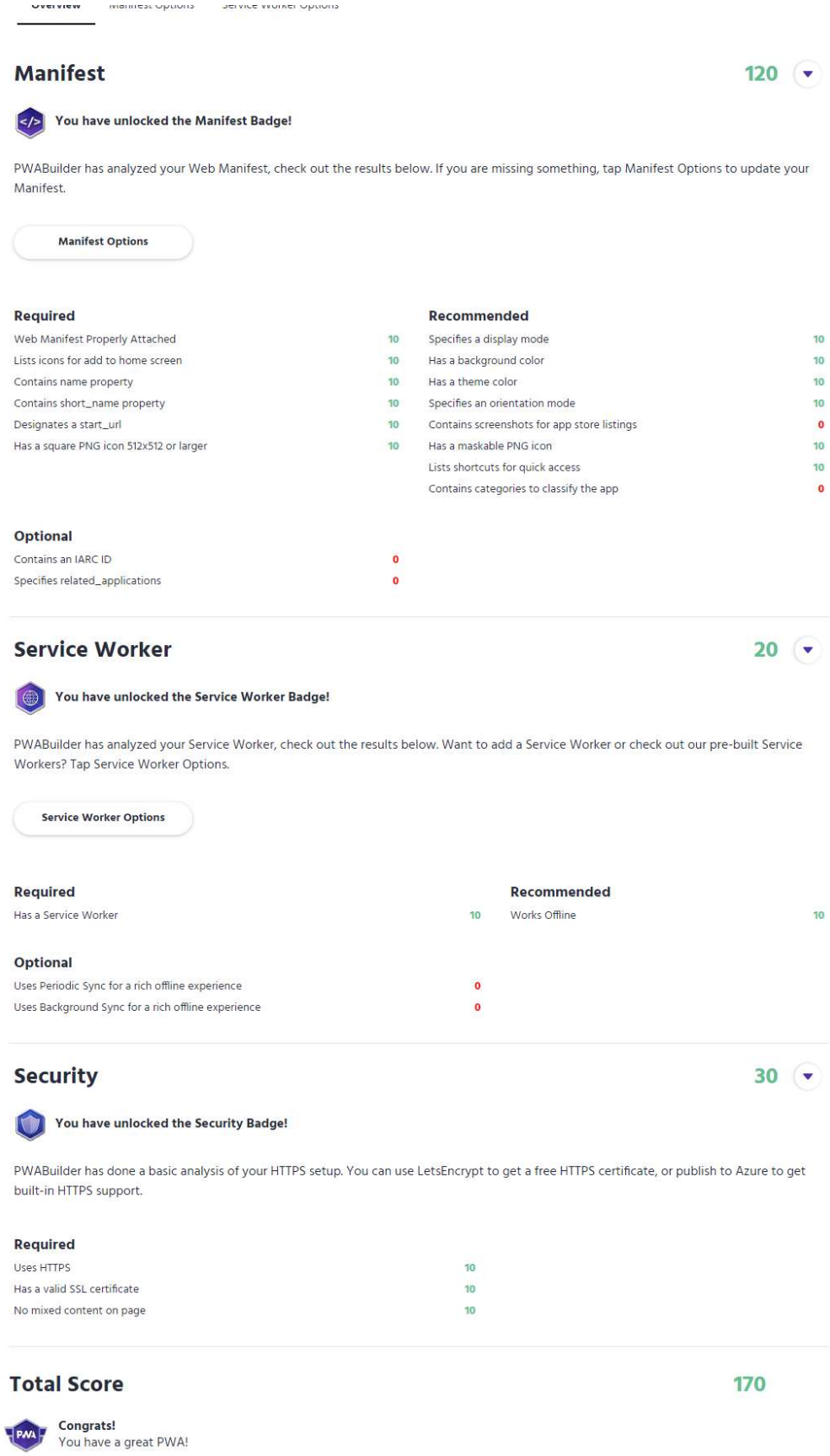
Figura 62 - PWABuilder página inicial



Fonte: Autor, 2021.

Na Figura 62 pode-se ver uma das soluções para geração de aplicações empacotadas baseadas em PWA, a aplicação compartilha alguns pontos em comum com a aplicação Google Lighthouse, contendo uma avaliação ainda mais abrangente do quesito PWA como representadas na Figura 63.

Figura 63 - Avaliações detalhadas do PWABuilder

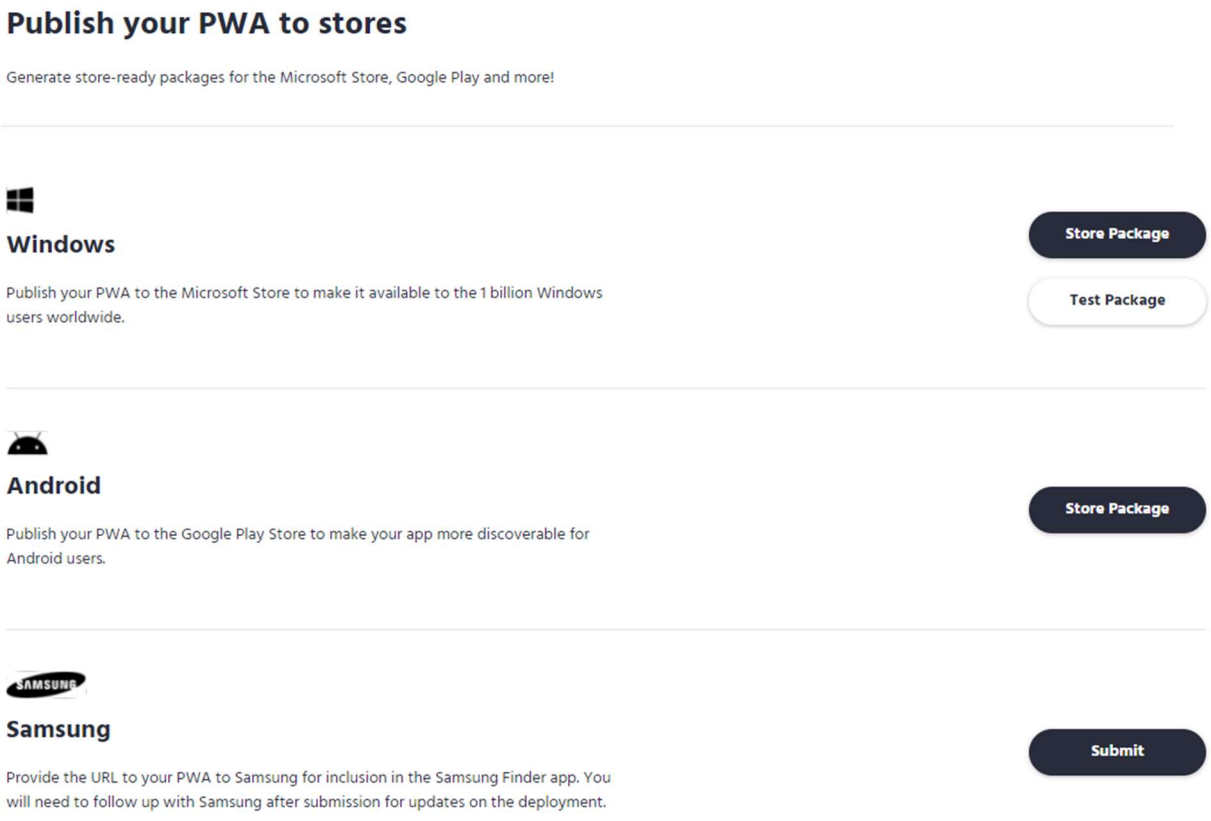


Fonte: Autor, 2021.

Se todos os requisitos obrigatórios estiverem sido corretamente implementados o PWABuilder disponibiliza uma próxima etapa para o empacotamento da aplicação proposta. Na Figura 64 nota-se a presença de 3 itens, a aplicação proposta visa a utilização do item Android onde consta a implementação do TWA.

Alemu (2020) mostra na Figura 22 (pág. 52) a aplicação PWABuilder quando estava em outra versão, onde fornecia também um *package* para MacOS, assim como um processo ou pouco diferente do atual.

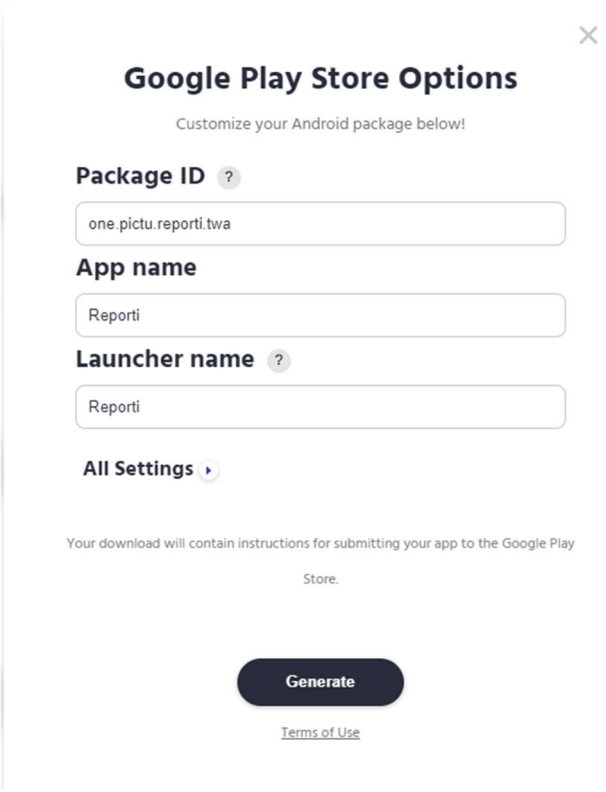
Figura 64 - Publicações disponíveis com o PWABuilder



Fonte: Autor, 2021.

Na Figura 65 tem-se a possibilidade de alterar algumas configurações antes da geração do pacote da aplicação, assim como uma *signing sey* onde pode-se adicionar a chave da aplicação criada na Google Play.

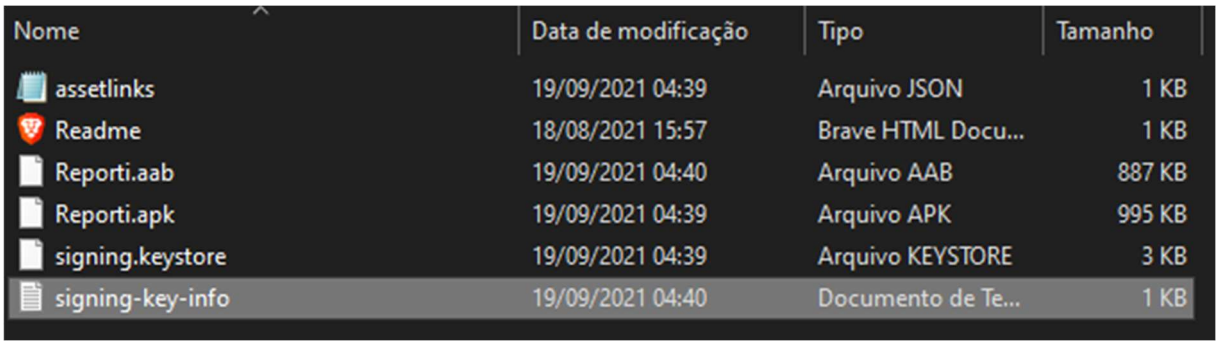
Figura 65 - Gerando o pacote Android no PWABuilder



Fonte: Autor, 2021.

Na Figura 66 pode-se ver os arquivos gerados pelo PWABuilder, assim como o arquivo *assetlinks* que deve ser disponibilizado na aplicação pela rota "/.well-known/assetlinks.json", de forma pública.

Figura 66 - TWA gerada pelo PWABuilder



Fonte: Autor, 2021.

Com os arquivos da Figura 66, torna-se possível o envio da aplicação para a Google Play Store, como um APP nativo convencional, mantendo a praticidade no envio de atualizações.

5.3.5 Configuração do NextJS

A aplicação PWA proposta desenvolvida em NextJS utiliza um *plugin* com a *lib* do WorkBox para a configuração do SW, que também foi utilizado nos trabalhos analisados nas seções 3.2 e 3.5.

Este *plugin* disponibiliza a configuração referente ao *Service Worker* da PWA através do arquivo "*next.config.js*", arquivo utilizado para as configurações do *framework,* representado na Figura 67.

Este *plugin* denominado "next-pwa", oferece um ambiente de configuração simples, precisando apenas da adição das linhas 1 e das linhas 18 a 23 da Figura 67, o *plugin* já estará funcional a partir disso, podendo ser configurado de outras formas posteriormente.

Figura 67 - Arquivo de configuração do NextJS

```
1  const withPWA = require("next-pwa");
2  const Dotenv = require("dotenv-webpack");
3  const runtimeCaching = require("next-pwa/cache");
4  const util = require("util");
5
6  const cacheTime = 365 * 24 * 60 * 60; // 365 days
7
8  runtimeCaching.map((item) => {
9    item.options.expiration.maxAgeSeconds = cacheTime;
10   if (item.options.cacheName === "others") {
11     item.options.expiration.maxEntries = 200;
12   }
13   if (item.options.cacheName === "static-image-assets") {
14     item.options.expiration.maxEntries = 100;
15   }
16 });
17
18 module.exports = withPWA({
19   pwa: {
20     dest: "public",
21     importScripts: ["/worker.js"],
22     runtimeCaching,
23   },
24   webpack: (config) => {
25     // Add the new plugin to the existing webpack plugins
26     config.plugins.push(new Dotenv({ silent: true }));
27
28     return config;
29   },
30   env: {
31     API_URL: process.env.API_URL,
32   },
33   images: {
34     deviceSizes: [320, 640, 768, 1024, 1600],
35     domains: ["reporti.s3.amazonaws.com"],
36   },
37   typescript: {
38     ignoreBuildErrors: true,
39   },
40   eslint: {
41     ignoreDuringBuilds: true,      You, 3 months ago • ini
42   },
43 });
44
```

Fonte: Autor, 2021.

A Figura 67 mostra algumas configurações importantes para o funcionamento da PWA no *framework,* destacando as importações do “*next-pwa*” que é o *plugin,* e do módulo que pode ser utilizado para personalizar as configurações de *cache.*

Neste módulo é atribuído a constante *runtimeCaching* que disponibiliza o acesso as principais configurações de *cache*, onde pode-se ver a ampliação do *cache* para um ano, pois o padrão do WorkBox é de um dia, assim como o *maxEntires* que é um limitador de arquivos mantidos em *cache*.

Os parâmetros são disponibilizados linha 19 da Figura 67, sobrescrevendo a constante *runtimeCaching*. Assim quando a aplicação for executada é gerado o SW na pasta destino conforme a linha 20 da Figura 67.

No presente capítulo foram abordados os procedimentos para o desenvolvimento da aplicação proposta, juntamente com testes comparativos e métricas de desempenho. No próximo capítulo é apresentada a conclusão do estudo realizado.

6 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo explorar o desenvolvimento de uma PWA utilizando os recursos disponíveis no *browser,* com a finalidade de aprofundar o conhecimento e aprimorar as técnicas de desenvolvimento *web*, analisando cada recurso selecionado para uso na aplicação, testando suas viabilidades e funcionamento.

A partir do referencial teórico pode-se constatar que o desenvolvimento e o uso de uma PWA podem ser aplicados em diversos cenários, pois trazem vantagens e benefícios relevantes.

Dentre estas vantagens e benefícios destacaram-se:

- Uso *offline* através do cache estabelecido pelo SW;
- O comportamento como uma aplicação nativa dado através do arquivo de manifesto aliado ao SW e técnicas de responsividade;
- O uso de APIs do navegador para acesso a recursos nativos como (câmera, geolocalização, banco de dados) de forma padronizada;
- Por rodar na web gera um alto alcance de uso por executar em qualquer navegador, podendo ampliar o engajamento, e o seu público;

- Pode contar com otimizações que levam a uma resposta mais rápida e um baixo consumo de dados;

- Apesar de poder ser adicionada a algumas lojas de aplicativos, uma PWA se torna desvinculada destas lojas, não sendo necessário seguir todas as diretrizes impostas, e não necessitando de aprovação prévia para o lançamento de uma atualização voltada ao sistema, tendo atualizações mais rápidas;

- Por rodar em uma gama alta de dispositivos, em muitos cenários pode ser adotada como único *frontend* da aplicação, não necessitando o desenvolvimento de algum outro para uma plataforma específica.

Para o desenvolvimento da aplicação proposta, foi necessário a criação de dois projetos sendo um o *backend* e outro *frontend,* sendo o projeto *frontend* o principal alvo deste estudo, e o que proporciona a interação direta com o usuário final.

O projeto proposto foi implementado com o *framework* NextJS tanto no *frontend* quanto no *backend*, por disponibilizar uma otimização muito relevante em termos de recursos, e conter meios que facilitam sua configuração para a execução da aplicação em modo de produção. Porém, apesar de ser possível o seu uso no *backend,* isso não é considerado o ideal já que o mesmo é voltado para o *frontend.*

A configuração automática do SW através de um *plugin* é uma vantagem, visto que o NextJS gera *builds* otimizados que diminuem a dificuldade de manter um SW configurado de maneira correta. Outro ponto importante é que por meio disso é possível garantir maior integridade da aplicação, pois em caso de atualizações, o *plugin* encarrega-se de resolver problemas que podem ocorrer com o SW.

A aplicação proposta teve um desempenho satisfatório nos testes de validação tanto no Google Lighthouse (Google) quanto no PWABuilder (Microsoft), passando em todos os itens julgados obrigatórios pelos dois *softwares* e tendo notas altas também em outros quesitos.

Por meio da ferramenta Google Lighthouse os principais resultados foram, 90% de performance, 97% de acessibilidade, 100% em boas práticas e 100% *Search*

Engine Optimization (SEO), atendendo todos critérios verificados no quesito PWA que foi o principal critério avaliado na aplicação.

Por meio da ferramenta PWABuilder atendeu a todos os critérios requeridos assim como boa parte dos critérios recomendados, resultando em um *score* de 170, que é considerado alto pela ferramenta PWABuilder.

O desenvolvimento da PWA utilizando o *framework* NextJS trouxe um conjunto muito efetivo em termos *cache*, desempenho, e escalabilidade já que o mesmo foi hospedado no serviço da *Vercel* que utiliza várias técnicas e aperfeiçoamentos e para otimizar a entrega do site ao usuário final.

Pode-se observar que muitos autores citados neste estudo deduziram que essa metodologia de desenvolvimento gera menos custo para ser aplicada, não gerando custo excedente na criação de outras aplicações para plataformas distintas, trazendo um maior foco na qualidade da aplicação.

A aplicação desenvolvida foi pensada para utilizar uma grande quantidade de recursos disponíveis nos navegadores, mas assim como toda a aplicação, é muito importante uma boa análise para deduzir se esta tecnologia vai suprir o que é esperado na aplicação final (BARTIÉ, 2002).

Outro ponto importante é que independente do *framework* ou linguagem utilizados para o desenvolvimento da PWA, o acesso utilizado para as APIs dos navegadores mantém-se o mesmo, pois as mesmas possuem padrões bem definidos de uso.

Como o aprimoramento progressivo é o foco da metodologia de desenvolvimento PWA, o desenvolvedor tem a possibilidade de escrever a aplicação fazendo com que a mesma vá melhorando através da adição de novos recursos com o tempo, como descrito neste estudo.

Além destes pontos a aplicação consegue rodar em qualquer máquina que tenha um navegador, tendo um alcance alto de público podendo desempenhar tarefas em todos estes dispositivos.

É importante analisar que é uma metodologia em constante evolução, onde novas funcionalidades são inclusas com o passar do tempo, e nota-se uma grande evolução dela no período de desenvolvimento deste estudo, onde as principais APIs sendo implementadas no momento são as de acesso ao *bluetooth* e NFC, o que mostra não possuir todos os recursos acessíveis como uma aplicação nativa, esta comparação foi vista em muitos estudos relacionados e citados neste estudo.

Outro ponto notado é o baixo engajamento da Apple onde em seu navegador nativo do iOS o Safari, atualmente pode não ter acesso a todos os recursos e contém anomalias no *layout*, como foi possível demonstrar no desenvolvimento deste projeto. Também não conta com uma opção para gerar um APP para sua loja, o que ainda pode inviabilizar a implementação de algumas aplicações, ainda que seja possível contornar muitos destes problemas.

Um ponto muito importante é que a metodologia de desenvolvimento PWA, auxilia em uma melhor experiência do usuário, em um ambiente que pode ser altamente otimizado, escalado, distribuível e compartilhável, sendo uma opção muito eficiente e vantajosa em diversos aspectos, e que muito ainda tem a ser explorado.

Para trabalhos futuros, há a possibilidade da inclusão de novas funcionalidades, assim como envio de áudio na denúncia, *background sync* para o envio da denúncia caso o denunciante esteja sem internet, o envio de notificação nas alterações de status de uma denúncia, um administrativo para o avaliador distribuir os casos para as entidades pertinentes em cada cidade, ou até mesmo a implementação de outros sistemas utilizando o que foi explorado neste estudo.

REFERÊNCIAS BIBLIOGRÁFICAS

MARTIN, Robert C. **Código Limpo**. Edição Revisada. Rio de Janeiro, 2011.

ALEMU, Yoseph. **Analysis of deploying a React PWA on Google Play store using Trusted Web Activity**. 2020. Disponível em: <https://www.theseus.fi/bitstream/handle/10024/352102/Analyis%20of%20deploying%20a%20React%20PWA%20on%20Google%20Play%20Store%20using%20TWA_Yoseph%20Alemu_Final%20Version%20%281%29.pdf?sequence=2#page=24&zoom=100,148,701>. Acesso em: 6 mar. 2021.

GARCIA, Xavier Baca. **Redisseny del web em WordPress de Ràdio 90 implementant uma PWA**. 2020. Disponível em: <http://openaccess.uoc.edu/webapps/o2/bitstream/10609/120647/7/xbacaTFG0620memòria.pdf>. Acesso em: 6 mar. 2021.

DA SILVA NETO, Azuel Aquiar. **Desenvolvimento de uma aplicação PWA que comporte outras aplicações usando arquitetura de micro frontend**. Goiânia, 2020. Disponível em: <https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/1037/1/TCC%20-%20Azuel.pdf>. Acesso em: 6 mar. 2021.

DA SILVA, Caetano F.B.; COSTA, Fredson. **Desenvolvimento de aplicação para controle de atendimentos de emergência utilizando web app (PWA)**. Palmas, 2018. Disponível em: <https://to.catolica.edu.br/revistas/index.php/riu/article/download/447/254>. Acesso em: 6 mar. 2021.

LEHMAN, M.M., **Laws of software evolution revisited**. London, 1996. Disponível em: <https://link.springer.com/chapter/10.1007/BFb0017737>. Acesso em: 6 mar. 2021.

FIGUEIREDO, Eduardo; LOBATO, Cidiane; DIAS, Klessis; LEITE, Julio; LUCENA, Carlos. **Um Jogo para o Ensino de Engenharia de Software Centrado na**

Perspectiva de Evolução. Rio de Janeiro, 2007. Disponível em: <https://homepages.dcc.ufmg.br/~figueiredo/publications/wei07ready.pdf>. Acesso em: 6 mar. 2021.

BARTIÉ, Alexandre. **Garantia de Qualidade de Software: as melhores práticas de engenharia de software aplicadas a sua empresa**. 5ª Tiragem. Rio de Janeiro, 2002. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=O57Us2kUh4oC&oi=fnd&pg=PR22&dq=criando+um+software&ots=ijhV-ztkG5&sig=SVMLqwLS-YBrMB01IWk67IbK1Oc#v=onepage&q=criando%20um%20software&f=false>. Acesso em: 6 mar. 2021.

TANDEL, Sayali Sunil; JAMADAR, Abhishek. **Impact of Progressive Web App Development**. India, 2018. Disponível em: <https://www.researchgate.net/profile/Sayali-Tandel-2/publication/330834334_Impact_of_Progressive_Web_Apps_on_Web_App_Development/links/5c5605d3a6fdccd6b5dde018/Impact-of-Progressive-Web-Apps-on-Web-App-Development.pdf>. Acesso em: 6 mar. 2021.

DE SOUZA, Camila Brandina Crispin; CINTRA, Fausto Gonçalves. **PWA: Tecnologia da informação a serviço de pequenas empresas**. Franca, 2018. Disponível em: <https://revistaedufatec.fatecfranca.edu.br/wp-content/uploads/2019/03/Camila-B.-Crispin-Souza.pdf>. Acesso em: 6 mar. 2021.

VIEIRA, Pâmela Rocha; GARCIA, Leila Posenato; MACIEL, Ethel Leonor Noia. **Isolamento social e o aumento da violência doméstica: o que isso nos revela?**. 2020. Disponível em: <https://www.scielo.br/pdf/rbepid/v23/1980-5497-rbepid-23-e200033.pdf>. Acesso em: 7 mar. 2021.

OLIVEN, Ruben George. **Violência e cultura no Brasil**. Rio de Janeiro, 2010. Disponível em: <https://static.scielo.org/scielobooks/b8n7j/pdf/oliven-9788579820069.pdf>. Acesso em: 7 mar. 2021.

OKABAYASHI, Nathalia Yuri Tanaka; TASSARA, Izabela Gonzales; CASACA, Maria Carolina Guimarães; FALÇÃO, Adriana de Araújo; BALLINI, Márcia Zilioli. **Violência contra a mulher e feminicídio no Brasil – impacto do isolamento social pela COVID-19**. Curitiba, 2020. Disponível em: <https://www.brazilianjournals.com/index.php/BJHR/article/download/9998/8381>. Acesso em: 7 mar. 2021.

PLATT, Vanessa Borges; GUEDERT, Jucélia Maria; COELHO, Elza Berger Salema. **Violence Against children and adolecents: notification and alert in times of pandemic**. Florianópolis, 2020. Disponível em: <https://www.scielo.br/pdf/rpp/v39/1984-0462-rpp-39-e2020267.pdf>. Acesso em: 7 mar. 2021.

NEDEL, Matheus Berkenbrock. **Sistema integrado para atendimento em restaurantes**. Florianópolis, 2020. Disponivel em: <https://www.riuni.unisul.br/bitstream/handle/12345/10132/tcc_matheus.pdf?sequence=1&isAllowed=y>. Acesso em: 16 mar. 2021.

FRANÇA, Fabiano dos Santos. **Web design responsivo: caminhos para um site adaptável**. Aracaju, 2015. Disponível em: <https://periodicos.set.edu.br/exatas/article/view/2220/1295>. Acesso em: 16 mar. 2021.

FRAIN, Bem. **Responsive Web Design with HTML5 end CSS3**. 2012. Disponível em: <https://books.google.com.br/books?id=fcfarreMQ9sC&printsec=frontcover&hl=pt-BR&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false>. Acesso em: 16 mar. 2021.

PEREIRA, Fabiana Andrade; KRZYZANOWSKI, Rosaly Favero; IMPERATRIZ, Inês Maria de Morais. **Técnicas de Search Engine Optimization (SEO) aplicadas no site da Biblioteca Virtual da FAPESP**. Lisboa, Portugal, 2018. Disponível em. <https://www.bad.pt/publicacoes/index.php/cadernos/article/download/1902/pdf>. Acesso em: 16 mar. 2021.

CECONI, Laura. **Experiência do usuário em Progressive Web Apps**. Caxias do Sul, 2018. Disponível em. <https://repositorio.ucs.br/xmlui/bitstream/handle/11338/4780/TCC%20Laura%20Ceconi.pdf?sequence=1&isAllowed=y>. Acesso em: 17 mar. 2021.

LINS, Gabriel de Souza. **Utilizando ReactJS para o Desenvolvimento de um Sistema de Alocação e Reserva no Campus da UFC em Quixadá**. Quixadá, 2019. Disponível em. <http://repositorio.ufc.br/bitstream/riufc/49762/1/2019_tcc_gdeslins.pdf> Acesso em 30 mar. 2021.

ADETUNJI, Oluwatofunmi; AJAEGBU, Chigozirim; OTUNEME, Nzechukwu; OMOTOSHO, Olawale J. **Dawning of Progressive Web Apps (PWA): Edging Out the Piffalls of Traditional Mobile Development**. Nigéria, 2020. <https://asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/5812/2144> Acesso em 30 mar. 2021.

TRUONG, Huy. **Build and Deploy a High-Performance Full Stack JavaScript Web Application. 2020**. <https://www.theseus.fi/bitstream/handle/10024/400458/Huy_Truong%20thesis%20final%20%282021%29.pdf?sequence=2&isAllowed=y> Acesso em 30 mar. 2021.

FRANCISCO, Igor Moreira. **Plataforma Web para Auxílio de Treinamentos do uso de Próteses para Indivíduos com Amputação de Membros Superiores**. Uberlândia, 2020. <http://repositorio.ufu.br/bitstream/123456789/31109/1/PlataformaWebPara.pdf> Acesso em 30 mar. 2021.

HOANG, Son Chu. **Shopify Upsell App: Using Next.js, React.js to Boost Sale**. 2020. <https://www.theseus.fi/bitstream/handle/10024/354290/Chu_Son.pdf?sequence=2> Acesso em 30 mar. 2021.

OROSZ, Edit. **Modern Web Development with JAMstack**. 2020. <https://www.theseus.fi/bitstream/handle/10024/341469/Modern%20Web%20Development%20with%20JAMstack_Edit%20Orosz_Thesis.pdf?sequence=2&isAllowed=y> Acesso em 30 mar. 2021.

DELGADO, Ismael Navas; NIETO, José Manuel Garcia. **Desarrollo de Aplicación web FIMED 2.0**. Málaga, 2021. <https://riuma.uma.es/xmlui/bitstream/handle/10630/21107/Doblas%20Jiménez%20Daniel%20Memoria.pdf?sequence=1&isAllowed=y> Acesso em 30 mar. 2021.

MOROZ, Bogdan. **Unit test Automation of react-redux Application with Jest and Enzyme**. Finlândia, 2019. <https://www.theseus.fi/bitstream/handle/10024/184586/Moroz_Bogdan.pdf?sequence=2&isAllowed=y> Acesso em 30 mar. 2021.

TRINH, Huy. **A Practical Approach to JavaScript Testing**. 2020. <https://www.theseus.fi/bitstream/handle/10024/349418/huytrinh_thesis.pdf?sequence=2&isAllowed=y> Acesso em 30 mar. 2021.

RIIHIMÄKI, Jukka. **Automatic Testing and Delivery in web Development**. 2019. <https://www.theseus.fi/bitstream/handle/10024/220076/Riihimaki_Jukka.pdf?sequence=2> Acesso em 30 mar. 2021.

GIL, Antônio Cartos. **Como classificar as pesquisas?**. 2002. < https://d1wqtxts1xzle7.cloudfront.net/38881088/como_classificar_pesquisas.pdf?1443122050=&response-content-disposition=inline%3B+filename%3DCOMO_CLASSIFICAR_AS_PESQUISAS_1.pdf&Expires=1622426544&Signature=SxZQC82ifjIP7M4BM9wVzdX~bpUF~p5A~S3lirQx3nSW9xpxtIpuQvZRXpwMhekUjb1PGl8j5PkSRYXfamoUc2l~yAtqJ1pAYnLypJR7JdP6SmHi-olaH1fsLNyElRAnJU~KBbwUioPOHG~EWleY55gAXGAu0p1byqLEvsu5SthVMyflhJmP2PzSOjsi7xMvRQ~jhXrn8E4cy5IF6LOwGbeGvdE1wjRcMJm1Vgi7lgQrsgtCTipAMW3Mu6kS-v6s5N94GU-0sTSowX0KZFegBAHDdI3Vq56AKCKGx3-EsbZDtuo~Nk-FdYewgTRZtHmok-RR3SCVeyYl2gW~QOSzeg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA> Acesso em 30 maio. 2021.

LAKATOS, Eva Maria; MARCONI, Mariana de Andrade. **Fundamentos de metodologia científica**. 2003. < http://docente.ifrn.edu.br/olivianeta/disciplinas/copy_of_historia-i/historia-ii/china-e-india/at_download/file> Acesso em 30 maio. 2021.

MORESI, Eduardo. **Metodologia da Pesquisa**. 2003. < https://d1wqtxts1xzle7.cloudfront.net/34909124/MetodologiaPesquisa-Moresi2003.pdf?1411907309=&response-content-disposition=inline%3B+filename%3DMetodologia_da_Pesquisa_PRO_REITORIA_DE.pdf&Expires=1622419222&Signature=ZFCFi~ERkGnNC-TCGKO4qAnlPITN2onAT57UrWReAS-Susk4WJpLAVe97bvmyJivdE8DRsIo3UvNCVAisR2SiEPoBFHDXaVC4TvDKNjTrrefRB8kWXnGUQUmh3G~HAFtGki2QrWfM~RBrSEnqyp9ywtVMupmLNSDsQRCStPk

FNIOEIugLqQoL73qA9s8s3nhU241P0VMk1T2XotNYva2m0JTptt0RQqQXxtMpqlkm
UlfRXnEQ9garSwTxTJiA3F-ng4mD4n5tuyGwc2W45r-
3UiIfvx5EbwSf20mvW04kiT1ggRN-ii3L70q-
9nZJ6qNuDjH3LMBdXTE6j~tzF7VEg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>
Acesso em 30 maio. 2021.

COCKBURN, Alistair. **Escrevendo casos de uso eficazes**. 2007.<
https://books.google.com.br/books?hl=pt-
BR&lr=&id=gbBRo8CxmFUC&oi=fnd&pg=PR3&dq=casos+de+uso+software&ots=s
W6ttiATEQ&sig=tkaOjPkInRX2NDi8HMNmZf9Laro#v=onepage&q=casos%20de%20
uso%20software&f=false>. Acesso em 13 jun. 2021.

STIVALA, Giada; PELLEGRINO, Giancarlo. **Deceptive Previews: A Study of the Link Preview Thustworthiness in Social Platforms**. Estados Unidos da América, 2020. < https://publications.cispa.saarland/3029/1/link-previews_ndss20.pdf >. Acesso em 31 out. 2021.

JUNIOR, Paulo Maehler. **Safe Space: um aplicativo de denúncias e auxílio legislativo para vítimas de violência**. Lajeado, 2020. <
https://univates.br/bdu/bitstream/10737/3035/1/2020PauloMaehlerJunior.pdf >**.**
Acesso em 10 dez. 2021.

UNIVATES

R. Avelino Tallini, 171 | Bairro Universitário | Lajeado | RS | Brasil