Projeto e implementação do aplicativo MistuRe como um Aplicativo Web Progressivo

Carlos Daniel De Moura Santos

Universidade Federal de Uberlândia
Faculdade de Computação
Bacharelado em Sistemas de Informação

Monte Carmelo - MG
2025

Carlos Daniel De Moura Santos

Projeto e implementação do aplicativo MistuRe como um Aplicativo Web Progressivo

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

Orientador: Rafael Dias Araújo

Monte Carmelo - MG
2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4144 - http://www.portal.facom.ufu.br/ facom@ufu.br

ATA DE DEFESA - GRADUAÇÃO

Curso de Graduação em:	Bacharelado em Sistemas de Informação				
Defesa de:	FACOM31804 - Trabalho de Conclusão de Curso 2				
Data:	29/05/2025	Hora de início:	10:00	Hora de encerramento:	10:56
Matrícula do Discente:	31811BSI022				
Nome do Discente:	Carlos Daniel de Moura Santos				
Título do Trabalho:	Projeto e implementação do aplicativo MistuRe como um Aplicativo Web Progressivo				
A carga horária curricular foi cumprida integralmente?	(X) Sim ( ) Não				

Reuniu-se, de forma remota, na plataforma Microsoft Teams, conta institucional da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Curso de Graduação em Sistemas de Informação - campus Monte Carmelo, assim composta: Prof. Dr. Bruno Augusto Nassif Travençolo - FACOM/UFU; Profa. Dra. Maria Adriana Vidigal de Lima - FACOM/UFU; e, Prof. Dr. Rafael Dias Araújo - FACOM/UFU, orientador do candidato.

Iniciando os trabalhos, o presidente da mesa, Prof. Dr. Rafael Dias Araújo, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao discente a palavra, para a exposição do seu trabalho. A duração da apresentação do discente e o tempo de arguição e resposta foram conforme as normas do curso.

A seguir, o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o(a) candidato:

(X) Aprovado, Nota [90] (Somente números inteiros)

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Rafael Dias Araújo**, **Professor(a) do Magistério Superior**, em 29/05/2025, às 10:58, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Maria Adriana Vidigal de Lima**, **Professor(a) do Magistério Superior**, em 29/05/2025, às 10:58, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Bruno Augusto Nassif Travençolo**, **Professor(a) do Magistério Superior**, em 29/05/2025, às 10:59, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **6376404** e o código CRC **0317BDDA**.

Referência: Processo nº 23117.036442/2025-49 SEI nº 6376404

Resumo

Este trabalho aborda a implementação do aplicativo MistuRe, uma solução em *Progressive Web App* (PWA) destinada a auxiliar produtores rurais, pesquisadores e estudantes na criação de misturas seguras de agrotóxicos em tanques de calda, com base em dados técnico-científicos. O objetivo central é democratizar o acesso ao sistema, originalmente limitado ao sistema operacional Android, por meio de uma multiplataforma que opera via navegador Web, independente do sistema operacional. Para isso, adotou-se o *framework* React em conjunto com Vite, integrando *Service Workers* para garantir funcionalidades offline, cache inteligente e atualizações automáticas. A avaliação técnica utilizou ferramentas como o Google Lighthouse e testes práticos em dispositivos móveis (Android, iOS) e desktops, analisando desempenho, responsividade e compatibilidade. Os resultados demonstraram que o aplicativo atende aos critérios essenciais de PWA, como instalação na tela inicial, operação offline e segurança, embora tenha identificado oportunidades de otimização no tempo de carregamento (desempenho de 56% na interface principal).

Palavras-chave: PWA, React, Misturas de Caldas, Agricultura de Precisão.

Lista de siglas

API Interface de Programação de Aplicação - *Application Programming Interface*

CSS Folha de Estilo em Cascata - *Cascading Style Sheet*

CORS Compartilhamento de Recursos entre Origens - *Cross-Origin Resource Sharing*

GB Gigabytes

HTML Linguagem de Marcação de HiperTexto - *HyperText Markup Language*

HTTPS Protocolo de Transferência de Hipertexto Seguro - *Hypertext Transfer Protocol Secure*

iOS Sistema operacional do iPhone - *iPhone Operating System*

MB Megabytes

PIB Produto Interno Bruto

PWA Aplicativo Web Progressivo - *Progressive Web App*

RAM Memória de Acesso Aleatório - *Random Access Memory*

TWA Atividades da Web confiáveis - *Trusted Web Activities*

UFU Universidade Federal de Uberlândia

URL Localizador Padrão de Recursos - *Uniform Resource Locator*

W3C Rede mundial de computadores - *World Wide Web*

Sumário

Apêndice A

Introdução

Este capítulo visa apresentar a motivação, o problema, a pergunta norteadora, os objetivos geral e específicos, as contribuições e a organização dessa monografia. Esses são, aspectos que contribuem para a idealização e construção do presente estudo baseado no uso da ferramenta *Progressive Web App* (PWA) em um aplicativo de mistura de caldas que oferta ampla colaboração às atividades do produtor rural em sua produção.

A.1 Motivação

O Brasil é atualmente um dos maiores produtores de alimentos do mundo, configurando-se também como um grande exportador mundial, sendo o trabalho do agronegócio, grande responsável por movimentar o Produto Interno Bruto (PIB) brasileiro. Todavia, o país também é conhecido pelo clima tropical e a intensidade da agricultura focada no plantio de um único produto agrícola, este fato, contribui para o surgimento de pragas, doenças e plantas daninhas que prejudicam o crescimento, desenvolvimento e rendimento da produção (GAZZIERO, 2015).

Afim de controlar a problemática criada pelas condições presentes na terra brasileira, o país também se tornou um dos maiores usuários de agrotóxicos, que são substâncias químicas lesivas a principalmente fungos, insetos e plantas específicas, ou ainda no combate a pragas prejudiciais ao plantio (GAZZIERO, 2015).

Comumente, há a ocorrência desses eventos simultaneamente em uma plantação, o que torna necessária a junção de mais de um tipo de agente agrotóxico para a manutenção da produtividade local. Essa prática é conhecida como mistura em tanque de caldas e se determina como a fusão de mais de um tipo de agrotóxico ou similar no tanque da máquina que realizará a pulverização imediatamente antes da aplicação com o objetivo de diminuir custos, reduzir o número de entradas no solo plantado e consequentemente a compactação do mesmo, e amenizar a exposição do trabalhador aos agentes tóxicos bem como aprimorar o manejo de combate às pragas (GAZZIERO, 2015).

Ainda, é preciso salientar que tais misturas precisam ser realizadas com cuidado e embasadas numa gama de conhecimentos técnico-científicos. Nesse sentido, considera-se que, pode haver interação entre os produtos possibilitando a ocorrência de malefícios à estabilidade da calda produzida, ineficiência das moléculas sobre seus alvos, além de no caso das incompatibilidades dos produtos, causar danos graves à capacidade de pulverização pela formação de cristais, flocos ou cremes que geram entupimentos nos aplicadores e consequentemente, custam mais caro ao produtor. Outrossim, o manejo incorreto das caldas pode ocasionar na inviabilidade de comercialização do produto plantado, tendo o produtor, um alto prejuízo financeiro (SILVA, 2021; CARDOSO; SANTOS; ARAÚJO, 2021).

Tendo em vista o especificado cenário, no ano de 2021, através do trabalho conjunto e da parceria entre discentes e docentes dos cursos de Sistemas de Informação e Engenharia Agronômica da Universidade Federal de Uberlândia (UFU) foi desenvolvido um Sistema Web e um Aplicativo Mobile, executável no sistema operacional Android, denominado MistuRe. O referido sistema conta com informações científicas de mais de 900 produtos presentes no mercado e as mais de 200 combinações possíveis de serem feitas entre eles, com o objetivo de garantir ao produtor rural a mistura de uma calda herbicida segura e compatível, baseada em dados seguros e constantes em plataformas científicas (CARDOSO; SANTOS; ARAÚJO, 2021).

Todavia, na execução do projeto, notou-se inviável a programação do aplicativo Mobile para o sistema operacional Sistema operacional do iPhone (iOS) considerando os altos custos dos materiais necessários para sua publicação, sendo considerada como solução, a inserção da ferramenta PWA para viabilizar o presente trabalho. Sendo assim, este estudo tratará da PWA como objeto para conseguir disponibilizar esse sistema e Aplicativo Mobile para qualquer usuário, apesar do sistema operacional utilizado. Nos últimos anos, a divisão entre usuários dos sistemas Android e iOS tem se mostrado cada vez mais desafiadora aos programadores tendo em conta a demanda de conhecimentos em duas áreas muito distintas. Dessa forma, objetivando tornar exequível tecnologias de desenvolvimento web, apesar do sistema operacional do aparelho em uso, colocando-as no âmbito do desenvolvimento Mobile surgiu a ferramenta PWA. O supracitado aparato consiste em conceder uma usabilidade aproximada ao uso de um aplicativo, tornando hábil a sua execução nos aparelhos telefônicos e computadores através de navegadores (SILVA; TIOSSO, 2020).

É válido ressaltar que a própria Apple ao longo dos anos, mesmo que de forma restrita, foi abrindo espaço para que houvesse a possibilidade de programar para seus aparelhos, e a partir de 2018, com o lançamento do iOS 11.3, a marca possibilitou o uso da PWA com um suporte simples de novas em tecnologias. Recentemente, foi lançado o PWA 12.2 ainda limitado para a PWA mas tornando evidente pela marca o incentivo dela ao uso dessa tecnologia nos seus dispositivos (SILVA; TIOSSO, 2020).

A.2 Problema

Atualmente, a plataforma MistuRe é composta por um sistema Web e um aplicativo móvel, que visa agregar e apresentar informações sobre resultados científicos sobre misturas de calda (CARDOSO; SANTOS; ARAÚJO, 2021). O aplicativo móvel foi inicialmente desenvolvido para o sistema operacional Android pela sua ampla base de usuários, maior participação de mercado e pela facilidade oferecida no processo de publicação e testes iniciais. Além disso, houveram publicações de resumos e artigos científicos na área relacionados às conquistas realizadas pelo projeto e documentação dos seus feitos. O trabalho também possui relação com as ações de ensino, pesquisa e extensão, base do sistema superior das universidades públicas brasileiras, englobando projetos nos respectivos âmbitos. Há, ainda, um público utilizador do sistema operacional iOS que não foi contemplado. No entanto, o desenvolvimento e a publicação de aplicativos para esse sistema operacional exige recursos que não são gratuitos e envolve um processo mais burocrático (NARDON, 2020).

No caso do Android, seria possível realizar a programação, no entanto, somente produtores que usam esse sistema seriam viabilizados com o recurso, não contemplando todos os possíveis usuários finais. Sendo assim, por meio da ferramenta PWA, é possível tornar o sistema e aplicativo semelhante a um site web, exequível a todo usuário, independentemente do sistema operacional (PATEL, 2019).

Considerando o supracitado, estabeleceu - se a seguinte questão norteadora: Há viabilidade e benefícios no uso de uma ferramenta PWA no aplicativo agronômico MistuRe?

A.3 Objetivos

É estabelecido como objetivo geral do presente estudo: Habilitar e transformar o sistema web e Aplicativo Mobile MistuRe para livre acesso dos usuários em iOS e Android, por meio de uma ferramenta PWA. Ademais, é proposto como objetivos específicos os seguintes: Contribuir com a criação de novos filtros de pesquisa que possibilitem ao produtor rural realizar buscas sem conhecimento técnico-científico;Disponibilizar um aplicativo multiplataforma via Web;Criar novos mecanismos de busca de dados;Validar o funcionamento do aplicativo em diferentes cenários de uso.

A.4 Contribuições

O presente projeto, terá como base a disponibilização de um Sistema Web e Aplicativo Mobile para todo usuário que tiver interesse, independentemente do sistema operacional dos seus dispositivos. Torna igualitário o acesso a um artefato que tem grandes contribuições para o usuário final, o produtor rural, auxiliando no desenvolvimento do agronegócio.

Ainda, contribui para o desenvolvimento da economia brasileira, considerando o papel da agricultura no país, e evita danos e perdas às plantações.

Ademais, o Sistema Web e aplicativo Mobile já existentes, são instrumentos de grande contribuição para a sociedade, tornar o acesso possível a todos os usuários ampliando o trabalho e colocando em evidência o papel da ciência, da tecnologia, da pesquisa em universidades públicas e da importância do trabalho interprofissional, como no caso desse, uma colaboração entre a Engenharia Agronômica e Sistemas de Informação.

A.5 Organização do Texto

Este texto tem seis partes. A primeira é a Introdução, que aborda as ideias gerais e dos pensamentos que deram vida a este estudo. A segunda parte é a Base Teórica, que concede o suporte técnico e científico sobre os temas da ferramenta PWA. A terceira parte avalia os Estudos Relacionados, mostrando pesquisas já feitas na literatura que ligam ao objetivo deste trabalho. A quarta parte trata do Desenvolvimento da Solução, detalhando as tecnologias usadas, as etapas de implementação e os desafios na criação do aplicativo. A quinta parte se chama Experimentos e Análise dos Resultados, na qual os testes realizados e a avaliação da aplicação são mostrados. Finalmente, o sexto capítulo aborda as Conclusões, contendo as observações finais e as perspectivas para pesquisas futuras.

Apêndice B

Fundamentação Teórica

B.1 Princípios e Características das PWAs

Na área das tecnologias a resiliência é condicionante para o sucesso do desenvolvimento de determinado software, isso se deve ao fato de que algo idealizado hoje pode não ter mais sua plena funcionalidade com o decorrer do tempo. Desse modo, o trabalho do desenvolvedor consiste também em idealizar produtos além da sua realidade temporal com vistas a garantir maior tempo de vida útil à ideia mesmo com as constantes transformações da contemporaneidade (MACCALI, 2021).

É nesse cenário que em 2015, através da Google o engenheiro da referida empresa Alex Russel e a designer Frances Berriman criaram o conceito PWA. Tal termo consiste em uma ferramenta moderna que se fundamenta na criação de aplicativos apoiados nos artifícios de navegadores com vistas a tornar a experiência do usuário a mais originária que se pode alcançar, fazendo uso de tecnologias comuns disponíveis nos meios informacionais como o JavaScript e o Linguagem de Marcação de HiperTexto (HTML) (MACCALI, 2021).

Segundo Gehring e Rossetto (2021) a PWA trata-se da criação da possibilidade de utilizar aplicativos com a única condição que se tenha um navegador no dispositivo, não demonstrando necessidade de criar aplicativos viabilizados somente para o sistema operacional iOS ou Android. Para eles, é um meio de ampliar o uso de inúmeros aplicativos que antes só seria possível o acesso com a compatibilidade do mesmo com o sistema operacional do dispositivo.

Em 2015, quando a PWA foi criada, foram estabelecidas qualidades para caracterizar os aplicativos relacionados com essa ferramenta, sendo essas:

- Responsivo: funcionalidade em telas independentemente do tamanho das mesmas, ou seja, do dispositivo usado pelo usuário;
- Conectividade independente: trabalho *offline* através do aprimoramento progressivo com *Service Workers*;

- ❏ Interações semelhantes a aplicativos: criação de navegação e interações de aplicativos através da adoção de um modelo de aplicativo *Shell + Content* para tornar a experiência do usuário mais nativa;
- ❏ Atualizado: através do *Service Worker* manter sempre atualizado;
- ❏ Seguro: por via do Protocolo *Transport Layer Security* prevenir a espionagem e garantir a seguridade das informações;
- ❏ Detectável: permissão para que os mecanismos de pesquisa os encontrem como aplicativos devido aos Manifestos do W3C e ao escopo de registro do *Service Work*;
- ❏ Reengajável: acesso às interfaces de usuário que reengajam o sistema operacional, como exemplo, as notificações *push*;
- ❏ Instalável: manter na tela inicial através de *prompts* disponibilizados pelos navegadores, os aplicativos que o usuário julgar mais útil sem a necessidade de uma loja de aplicativos;
- ❏ Linkável: inexistência de atritos e instalações, fáceis de compartilhar, devendo usar do poder social das URLs (RUSSEL; BERRIMAN, 2015).

A literatura especifica quanto à fatores imprescindíveis para criar uma PWA, incluindo os seguintes conceitos: manter um arquivo manifest.json na fundação do sistema, o mesmo aplica informações relevantes para o funcionamento do sistema como nome do navegador web, cor, modo de execução e outros; o *script Service Work*, que possibilita a execução do navegador em *background*, separado em uma página web, possibilitando o acesso à recursos que não demandam uma página na web ou interação do usuário (GEHRING; ROSSETTO, 2021; PATEL, 2019; GAUNT, 2019).

Desse modo, posteriormente serão discutidos alguns conceitos básicos presentes na literatura a respeito da ferramenta PWA conectados ao que foi supracitado até aqui e que são fundamentais para o desenvolvimento de um PWA na unificação web e mobile, considerando também o objetivo do presente trabalho.

B.2 Aprimoramento Progressivo e semelhança de interações

Aprimoramento progressivo é um termo atribuído a um tipo de desenvolvimento web que prioriza o conteúdo web. Nesse sentido, a sua construção decorre da fragmentação da apresentação em uma ou mais camadas opcionais, que são ativadas baseadas no navegador e na conexão de internet do usuário final, ou seja, as informações são disponibilizadas de acordo com a navegação do indivíduo.

Essa classificação de desenvolvimento web funciona sob alguns princípios, dentre eles podemos citar: acessibilidade do conteúdo para qualquer navegador, alcance das funções básicas em qualquer navegador, todo conteúdo do site precisa contar com a marcação semântica, o CSS precisa estar linkavel externamente, é contraindicado manusear código no HTML ou inserir script ao longo do conteúdo, e as demandas do navegador devem ser respeitadas.

Com essas especificações, efeitos possíveis de serem feitos em sites como animações, e efeitos visuais ficam restringidos. O site ainda é livre para colocar suas apresentações em diferentes estilos, no entanto, se o navegador utilizado pelo usuário não suportar tais quesitos, a pessoa visualizará um site com o visual bem mais simples, contudo, totalmente funcional (ARAÚJO, 2020).

A PWA funciona com aprimoramento progressivo, dessa maneira, independente do local, do provedor de internet ou do navegador que o usuário esteja utilizando, será possível ter acesso. Ademais, o aprimoramento progressivo funciona na lógica semelhante à da PWA de configurar a melhor experiência possível ao usuário com vistas à semelhança das interações, também uma característica proposta por Russel e Berriman. A mesma, propõe a semelhança de experiências e a melhor experiência possível (MACCALI, 2021).

B.3 Conectividade

Como pontuado por Russel e Berriman (2015), para o pleno funcionamento da ferramenta PWA é importante usar a conexão com a internet. Todavia, através do *Service Work* é possível haver funcionalidade. Isso se deve por esse modelo de *script* executar o navegador em segundo plano, o que ocorre em um sistema de armazenamento em cache. No que tange a confiabilidade, essa prática é considerada segura e todo acesso conta com certificado de segurança, objetivando assegurar a transferência de dados com segurança (CARMARGO, 2019).

B.4 Atualização e segurança

No que se refere à atualização, comumente todas as vezes que o usuário acessar a PWA ela será atualizada, sofrendo atualizações contínuas não necessitando, assim, de interferência do indivíduo no dispositivo (MACCALI, 2021).

Ademais, para a funcionalidade de uma ferramenta PWA é de suma importância mantê-la sempre atualizada e segura. Tais características também são propostas de Russel e Berriman (2015). Salienta-se ainda que para a PWA ter acesso às configurações originais do dispositivo é necessário obter o certificado Secure Socket Layer ativo, acessado por Protocolo de Transferência de Hipertexto Seguro (HTTPS) (MACCALI, 2021) .

B.5 Detectável, compartilhável, linkável e cativante

Como qualquer web app a PWA também é um instrumento fácil de compartilhar. Sem maiores complicações, o compartilhamento é possível através de um simples link, o que faz desse sistema facilmente compartilhável e cativante com novos usuários. Também, como estratégia para cativar novos usuários, a PWA conta com as notificações *push* – oriundas da aplicação em sua ação ou de um serviço externo - que propiciam o engajamento do público geral. (SOUZA; CINTRA, 2018; CECONI, 2019; MACCALI, 2021).

Para se tornar mais detectável a PWA trabalha com indexadores, nesse sentido, o Manifesto Rede mundial de computadores (W3C) por exemplo, torna a web aplicada em PWA facilmente encontrado como aplicativo (AVRELA, 2020).

Para guiar a instalação e padronização da PWA, existe no mercado o Manifesto do Aplicativo Web, o mesmo trata-se da disposição de dados sobre uma determinada aplicação em texto, onde é possível moldar as configurações da PWA, com o intuito de expandir o uso de aplicativos web na tela inicial de um dispositivo, oportunizando aos usuários uma experiência otimizada e o acesso ligeiro. Por esse meio, são repassadas informações que viabilizam a instalação de uma PWA (MOZILLA, 2022).

Apêndice C

Trabalhos Correlatos

Este capítulo objetiva evidenciar estudos encontrados que se relacionam com este trabalho, do ponto de vista de aspectos técnicos, metodológicos ou aplicados sobre *Progressive Web Apps* (PWA), especialmente em contextos de acessibilidade multiplataforma, desempenho e uso em setores específicos, como o agronegócio.

Inicialmente, no trabalho de Maccali (2021), foi considerado o contexto da violência contra a mulher como ponto de partida e o crescimento dessa ocorrência no contexto pandêmico. Em suma, o projeto buscou criar um sistema de denúncias e evidenciou as vantagens e particularidades no desenvolvimento PWA em três sistemas operacionais distintos, sendo eles: Windows, iOS, Android. A referida pesquisa buscou também descrever os conceitos envolvidos de acordo com a literatura, almejando estabelecer parâmetros para determinar a viabilidade da PWA e da Atividades da Web confiáveis (TWA), especialmente no sistema Android. O estudo concluiu, por fim, que a PWA e TWA se apresentam competentes em todos os sistemas operacionais testados, o que corrobora com os estudos relacionados que endossavam a eficiência dessas ferramentas.

Em seguida, o trabalho de Cardoso, Santos e Araújo (2021) foi o que originou, deu base e possibilitou que este acontecesse. Trata-se do estudo que evidencia a criação da Plataforma MistuRe, que serviu de base para o uso da PWA como sequência dos estudos nesse trabalho. Desse modo, o trabalho apresentou o processo de criação do sistema web e mobile MistuRe, com o objetivo de unificar dados científicos no que tange à conformidade de misturas de diferentes agrotóxicos. MistuRe conta com a possibilidade de 224 combinações com 990 produtos comercializados no Brasil. O artigo concluiu com a construção desse sistema web e aplicativo mobile com boa aceitação dos produtores e um bom fator de impacto nas propriedades agrícolas.

Na mesma linha, o trabalho de Gehring e Rossetto (2021) realiza uma análise comparativa entre frameworks JavaScript para o desenvolvimento de PWA, considerando as bibliotecas Vue.js e React. Para realizar tal feito, foram aplicados testes com serviços de localização, armazenamento e hospedagem. Também foi avaliada a funcionalidade dessas aplicações através do instrumento Google Lighthouse. Segundo os autores, o Go-

ogle Lighthouse avalia o desempenho de aplicativos e páginas web, propondo melhorias com base em critérios fundamentais para PWAs. O artigo concluiu que são necessárias melhorias futuras, partindo da análise criteriosa da PWA, alterando de acordo com os apontamentos do Google Lighthouse, melhorando os pontos necessários em cada framework.

Complementando a análise de compatibilidade, Silva e Tiosso (2020) objetivaram avaliar a ferramenta PWA em diferentes navegadores e sistemas operacionais, visando a otimização da experiência do usuário. Os testes foram realizados considerando Android e iOS, com os navegadores Chrome, Firefox, Samsung, Safari e Internet Explorer. Foi utilizada uma PWA auditada e verificada pelo aplicativo Lighthouse. O estudo concluiu que o sistema operacional Android, aliado ao navegador Chrome, propiciou maior suporte para uma experiência mais nativa e otimizada ao usuário, enquanto os demais apresentaram certas limitações.

Por outro lado, o artigo de Souza e Cintra (2018) teve como objetivo demonstrar, com exemplos práticos, o desenvolvimento de software compatível com PWA. O estudo destaca que essa tecnologia permite criar aplicativos com aparência de sistemas web, favorecendo tanto o usuário quanto o desenvolvedor. A proposta foi a de fornecer uma base conceitual para que desenvolvedores iniciantes consigam estruturar sua primeira aplicação progressiva.

Já o trabalho de Costa e Pires (2018) reforça o contexto da crescente popularização dos smartphones e a necessidade de criar aplicações que funcionem em qualquer sistema operacional. O estudo defende o uso da PWA como alternativa viável para atingir esse objetivo. O processo de desenvolvimento foi analisado sob o ponto de vista de custo-benefício, e os autores concluíram que a escolha pela PWA proporciona uma experiência similar à de aplicativos nativos, com atualização automática e menor custo de produção.

Dessa forma, observa-se que os trabalhos analisados convergem ao reconhecer as Progressive Web Apps como uma solução eficiente, multiplataforma e de baixo custo para o desenvolvimento de sistemas modernos. Em comum, destacam a viabilidade técnica da PWA, sua responsividade e a capacidade de operar offline. No entanto, divergem nos contextos de aplicação: enquanto alguns focam em utilidades sociais ou comerciais genéricas, este trabalho se diferencia por aplicar a tecnologia em um cenário agrícola específico, com forte embasamento técnico-científico para auxiliar na mistura segura de agrotóxicos.

Em síntese, todos os trabalhos revisados concordam na eficácia das PWAs para prover experiência semelhante à de apps nativos, com instalação via navegador e operação offline. Diferenças surgem principalmente nos detalhes de performance (medidos pelo Lighthouse) e no suporte variável de recursos em navegadores (mais completo no Chrome/Android e mais limitado no Safari/iOS). O MistuRe PWA, ao juntar requisitos de compatibilidade de agrotóxicos e práticas de PWA, dialoga diretamente com esses achados, mas seu foco específico em agronegócio o distingue das aplicações genéricas estudadas na literatura.

Apêndice D

Desenvolvimento da Solução

Este capítulo descreve de forma sistemática as etapas envolvidas no desenvolvimento do aplicativo MistuRe, concebido como um Aplicativo Web Progressivo (PWA). São apresentadas as principais decisões técnicas adotadas ao longo do processo, os recursos e ferramentas empregados, bem como as fases que compuseram o ciclo de desenvolvimento, desde a concepção inicial até a implementação final. O objetivo é oferecer uma visão clara das escolhas de arquitetura, tecnologias utilizadas e estratégias de construção da aplicação.

Em um trabalho desenvolvido anteriormente por (CARDOSO; SANTOS; ARAÚJO, 2021), há um sistema Web gerencial que permite o cadastro das substâncias e misturas, geralmente feitas por pesquisadores. As misturas são classificadas com base nos resultados biológicos encontrados em pesquisas científicas, podendo ser positivos (sinérgicos) ou negativos (antagônicos). A Figura 1 apresenta o exemplo de uma combinação de agrotóxicos proposta pelo usuário e sua funcionalidade.

D.1 Metodologia de Desenvolvimento

O processo de desenvolvimento do software seguiu uma abordagem ágil (PRESSMAN; MAXIM, 2021), embora sem equipe, por meio de atividades iterativas com os responsáveis pelo desenvolvimento do aplicativo original MistuRe, a fim de entender os requisitos postos e projetar a integração com o sistema *backend* já desenvolvido. O trabalho passou pelas seguintes etapas:

1. **Entendimento dos requisitos**: Realizou-se uma análise detalhada das funcionalidades presentes na versão mobile original do aplicativo, com o intuito de identificar quais recursos deveriam ser preservados, modificados ou descartados na nova implementação. Esse processo envolveu a compreensão do uso atual da aplicação e a visão dos coordenadores do projeto.

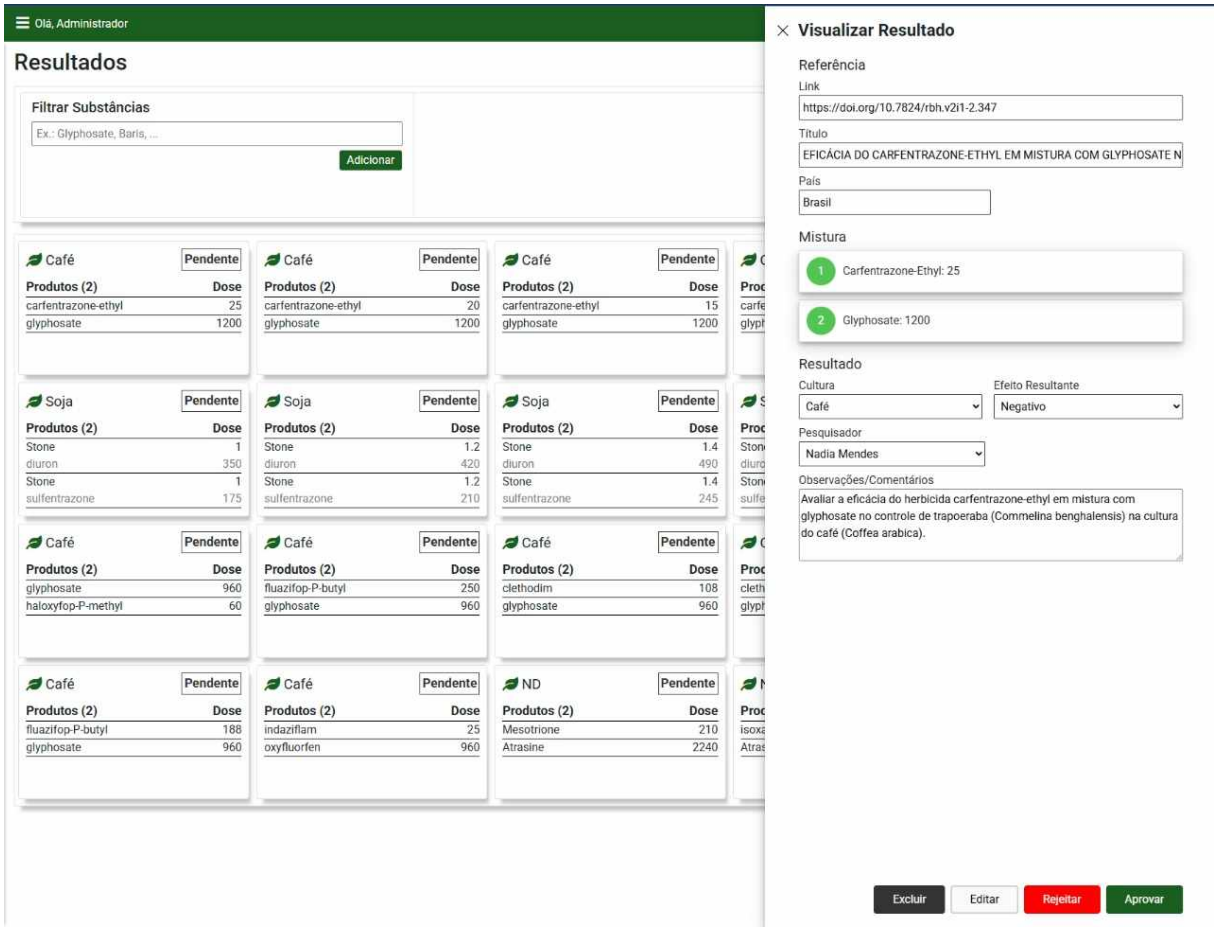


Figura 1 – Resultados de mistura no sistema em 2021. Fonte: Elaborado pelo autor

2. **Configuração do ambiente**: Estabeleceu-se a infraestrutura básica para o projeto por meio da instalação do Vite, uma ferramenta moderna de compilação (*build*) e empacotamento (*bundling*), juntamente com a configuração inicial do *framework* React. Também foi realizada a integração com o plugin específico para suporte a funcionalidades de um PWA, criando a base técnica necessária para o desenvolvimento posterior.

3. **Adição de funcionalidade offline**: Foi configurado o *service worker* com o objetivo de habilitar o funcionamento offline da aplicação, por meio do armazenamento em cache dos recursos fundamentais. Esta etapa visou melhorar a experiência do usuário em situações de conectividade limitada ou ausente.

4. **Configuração do manifest.json**: Elaborou-se o arquivo de manifesto da aplicação, contendo informações como o nome de um PWA, ícones, cores do tema e outros metadados obrigatórios. Esses elementos são essenciais para que a aplicação possa ser instalada em dispositivos móveis e integrada de forma mais fluida ao sistema operacional.

5. **Testes locais**: Foram conduzidos testes manuais emulados em diferentes dispositi-

vos móveis, com o intuito de verificar o correto funcionamento dos recursos característicos de um PWA. Foram avaliados aspectos como a instalação da aplicação, o comportamento offline e a adequação da interface em múltiplos contextos de uso.

D.2 Visão Geral do Fluxo Principal do Usuário

Quando entrar no sistema pela primeira vez, o usuário verá a tela para logar. Aqui, deve dar seu e-mail. Se o e-mail já for conhecido, irá direto para a *dashboard*. Se não, irá para a parte onde realiza seu cadastro, colocando dados simples para abrir sua conta.

No painel principal (*dashboard*), vai ver *cards* com os dados já carregados pelo sistema. Cada *card* tem suas informações importantes como a cultura feita, a substância usada e quanto foi usada. Para achar o que precisa fácil, tem um botão de filtro. Com ele, o usuário pode escolher por cultura, tipo de dado, classe da substância, ingrediente ativo, e doses mínima e máxima. A Figura 2 mostra a tela em que o usuário pode selecionar filtros relacionados às misturas cadastradas.

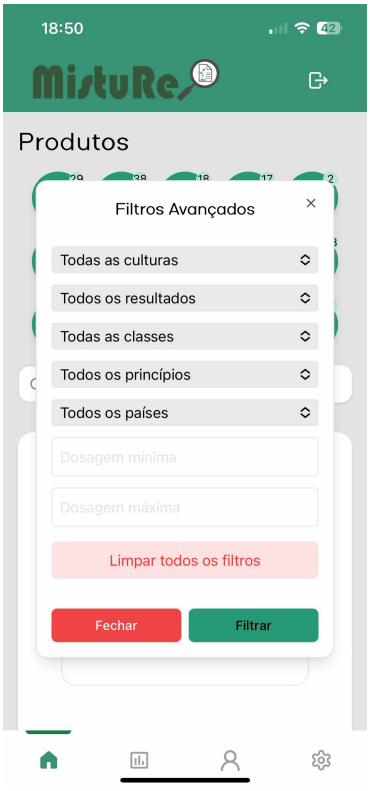


Figura 2 – Filtros que usuário pode selecionar. Fonte: Elaborado pelo autor

Além disso, o sistema pode operar mesmo sem acesso à internet. Se a conexão falhar, o usuário será avisado sobre estar offline, mas ainda poderá explorar normalmente as páginas dos resultados. Se o usuário aplicar algum filtro enquanto estiver desconectado, o sistema registrará as preferências e, assim que a conexão for restabelecida, enviará uma

notificação questionando se ele deseja aplicar os filtros que configurou durante o tempo offline.

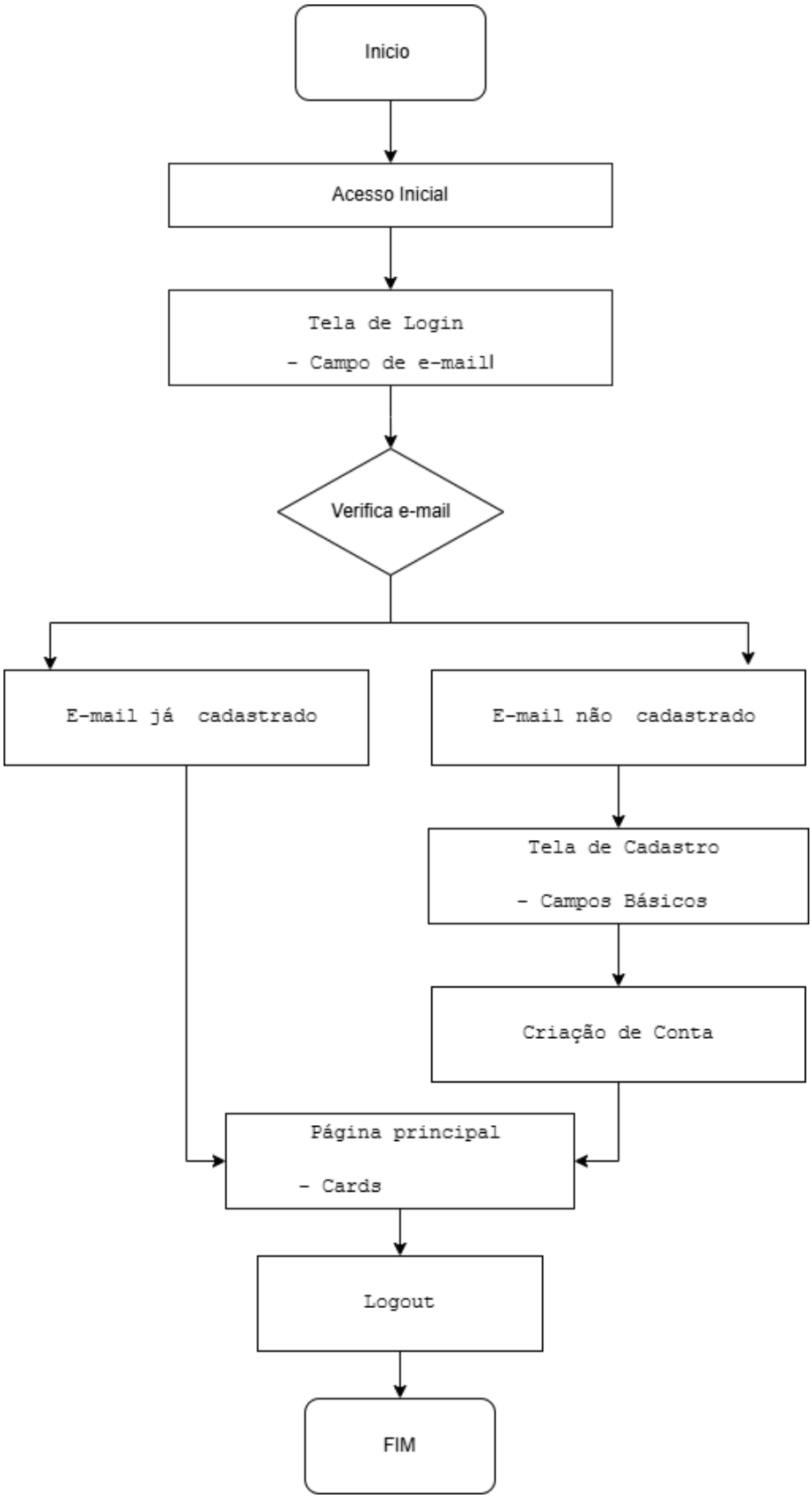


Figura 3 – Representa o fluxo do usuário em modo geral. Fonte: Elaborado pelo autor

D.3 Arquitetura da Aplicação

A aplicação *MistuRe* foi desenvolvida utilizando a arquitetura de Aplicação de Página Única, um modelo frequente em aplicativos do tipo PWA. Esse método assegura que toda a informação seja carregada de uma única vez, o que indica que as futuras ações do usuário não requerem o reenvio de toda a página. Dessa forma, a experiência de navegação se torna mais ágil e suave.

O sistema é projetado a partir de componentes que podem ser reutilizados, seguindo o princípio do React. Isso torna a manutenção, a ampliação e a organização do código mais fáceis. Ademais, essa escolha simplifica a inserção progressiva de novas funcionalidades, auxiliando na adaptação a futuras mudanças e demandas.

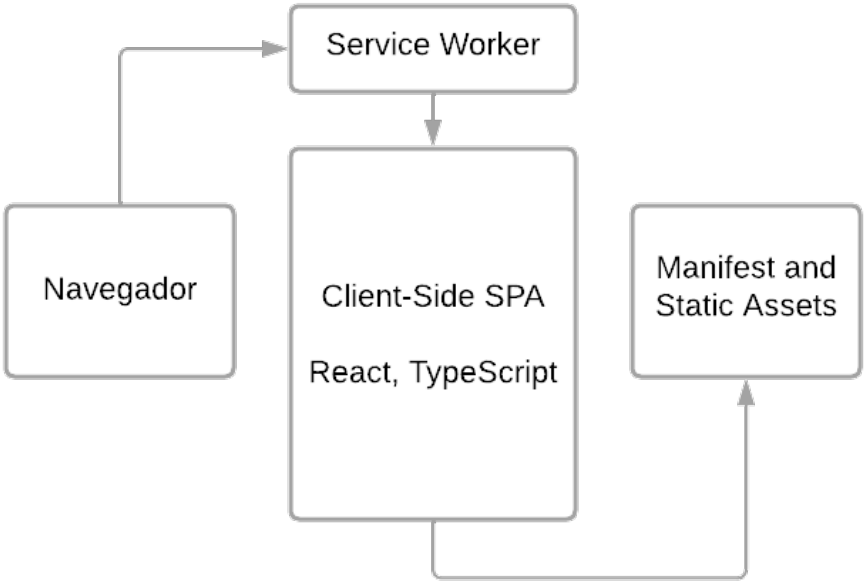


Figura 4 – Arquitetura geral da aplicação MistuRe. Fonte: Elaborado pelo autor

O sistema MistuRe é uma aplicação do tipo PWA, que pode ser usada tanto em dispositivos móveis quanto em computadores desktop. Para o bom funcionamento da aplicação, os seguintes requisitos mínimos são recomendados:

D.3.1 Requisitos para dispositivos móveis

- ❏ Sistema operacional: Android 8.0 ou superior, iOS 13 ou superior.
- ❏ Navegador compatível: Google Chrome, Safari ou Firefox atualizados.
- ❏ Conexão com a internet para o primeiro acesso (funciona offline após instalação).
- ❏ Armazenamento disponível: mínimo de 50 Megabytes (MB).

D.3.2 Requisitos para computadores (desktop ou notebook)

- ❏ Sistema operacional: Windows 10 ou superior, macOS 10.13 ou superior, ou distribuições Linux modernas.
- ❏ Navegador: Google Chrome, Microsoft Edge, Mozilla Firefox (versão mais recente).
- ❏ Processador: Dual core ou Superior.
- ❏ Memória Memória de Acesso Aleatório (RAM): mínimo de 2 Gigabytes (GB).
- ❏ Espaço em disco: mínimo de 100 MB livres.
- ❏ Acesso à Internet para o primeiro uso (funciona offline após instalação).

D.4 Tecnologias Utilizadas

A seguir, são apresentadas as tecnologias empregadas no desenvolvimento da solução, cada uma desempenhando um papel específico na construção da interface, na organização da lógica interna e na viabilização de suas funcionalidades. A seleção dessas ferramentas levou em conta critérios como desempenho, escalabilidade, facilidade de manutenção e aderência às boas práticas contemporâneas de desenvolvimento web.

- ❏ **HTML/CSS**: Essas tecnologias são a base da web, usadas para criar a estrutura e o visual da interface da aplicação. No MistuRe, o *HTML* organiza os elementos na tela, enquanto o *CSS* — especialmente através do Tailwind — garante que os componentes tenham uma aparência moderna e responsiva, com botões arredondados, cores que combinam com o contexto e um espaçamento bem distribuído (FREEMAN, 2015).
- ❏ **React**: É a biblioteca principal *JavaScript* do projeto. Ela organiza a aplicação em componentes, como formulários de cadastro, tabelas de substâncias e menus de navegação. Por exemplo, ao acessar o histórico de misturas, o React monta dinamicamente os cards com base nos dados disponíveis no estado do aplicativo (BANKS; PORCELLO, 2020).
- ❏ **React-Toastify**: Responsável por exibir mensagens visuais de alerta ou confirmação. Quando o usuário salva uma mistura ou ocorre um erro de validação, essa biblioteca dispara uma notificação elegante no canto da tela, sem a necessidade de redirecionamento ou recarga de página (KHADRA, 2024).
- ❏ **Shadcn**: Utilizada para compor a camada visual da aplicação com componentes estilizados, como botões, modais e abas. A biblioteca proporciona uma UI padronizada e acessível, acelerando o desenvolvimento sem comprometer a experiência do usuário (SHADCN, 2024).

- **Tailwind CSS**: Framework utilitário de CSS que permite criar estilos diretamente no HTML/JSX com classes como `bg-green-500`, `rounded-xl`, entre outras. No MistuRe, ele facilita a construção de interfaces responsivas e leves sem precisar escrever folhas de estilo externas (LABS, 2024).

- **TypeScript**: Adotado no lugar do JavaScript puro, adiciona tipagem estática, permitindo detectar erros durante o desenvolvimento e melhorando a legibilidade e a robustez do código. Por exemplo, ao definir um tipo para uma substância química (`SubstanceType`), o sistema garante que todas as operações com esse objeto respeitem sua estrutura (ROSENWASSER, 2022).

- **Vite**: É a ferramenta de build da aplicação. Ela acelera o tempo de inicialização do projeto e permite uma recarga quase instantânea durante o desenvolvimento. Isso reduz o tempo entre a escrita do código e o teste no navegador (VITE, 2024).

- **vite-plugin-pwa**: Este plugin transforma o MistuRe em um PWA, permitindo que o usuário instale o sistema no dispositivo e o utilize mesmo sem conexão com a internet. Ele cuida da geração do manifesto e do registro do *Service Worker* (HAMEL; CONTRIBUTORS, 2024).

- **Workbox**: Embora não utilizada diretamente no código-fonte, a funcionalidade de Service Worker é baseada nas estratégias oferecidas por essa biblioteca. Ela define quais arquivos devem ser armazenados em cache, como atualizá-los e como responder a falhas de rede, garantindo o suporte offline (GAUNT, 2019).

- **Zustand**: Biblioteca de gerenciamento de estado leve. No MistuRe, é utilizada para controlar estados globais como login do usuário, dados das substâncias, temas visuais, entre outros, sem a complexidade de soluções mais robustas como Redux (COLLECTIVE, 2024).

D.5 Detalhes da implementação

Esta parte descreve as seções mais relevantes do aplicativo MistuRe, suas características e as escolhas técnicas feitas ao longo do processo de desenvolvimento. Além disso, são explicadas as configurações necessárias para converter a aplicação em um PWA.

D.5.1 Conexão com o Sistema *Backend*

A ligação com o *backend* foi realizada através das rotas fornecidas pelo sistema MistuRe Web. Para que o Frontend em React pudesse acessar essas rotas, foi necessário implementar um *Proxy* para permitir consumir em domínios distintos e contornar a questão do Compartilhamento de Recursos entre Origens (CORS).

A seguir, são listadas as *Rotas* usadas no projeto.

❑ **Verificação de email:**

```
GET https://mistureapp.com.br/controller/Appcontroller.php?getByEmail={
    encodeURIComponent(email)}
```

Essa rota é utilizada quando o usuário realiza o login na aplicação. Ela verifica se ele precisa se registrar, caso não precise, ele será redirecionado para página principal.

❑ **Listagem de perfis:**

```
GET https://mistureapp.com.br/controller/PerfilController.php?getAll
```

Com essa rota, realiza-se a busca de todos os perfis cadastrados pelo sistema com intuito do usuário selecionar em um campo *select* no momento do cadastro.

❑ **API do IBGE (sem proxy):**

```
GET https://servicodados.ibge.gov.br/api/v1/localidades/estados
GET https://servicodados.ibge.gov.br/api/v1/localidades/estados/{UF}/
    municipios
```

A Interface de Programação de Aplicação (API) do IBGE foi utilizada para preencher dinamicamente as listas de estados e cidades do formulário de cadastro de usuário.

❑ **Registrar no Sistema:**

```
const payload = new URLSearchParams({
  email,
  uf: estadoSelecionado,
  nome,
  cidade: cidadeSelecionada,
  perfilId: perfilSelecionado,
});

const API_BASE = "https://mistureapp.com.br/controller";
rawUrl = `${API_BASE}/AppUserController.php?getByEmail=${
    encodeURIComponent(email)}`;
```

```
POST https://mistureapp.com.br/proxy.php?url=${encodeURIComponent(
    rawUrl)}
```

Os dados do usuário (nome, e-mail, estado, cidade e perfil) são encapsulados como parâmetros da requisição para realizar o cadastro desse usuário.

```
    const BACKEND_URL = "https://mistureapp.com.br";
    const urlOriginal = `${BACKEND_URL}/controller/ResultadoController.
   php?status=1&filter=true&userId=${userId}`;
    const proxyUrl = `${BACKEND_URL}/proxy.php?url=${encodeURIComponent
   (urlOriginal)}`;

    const response = await fetch(proxyUrl, {
      credentials: "include",
    });
    const data = await response.json();
```

Este caminho é utilizado para obter os resultados associados a um usuário específico. Emprega-se uma Localizador Padrão de Recursos (URL) com opções de filtragem e um proxy para superar limitações de CORS, assegurando que as informações sejam adquiridas de maneira apropriada.

D.5.2 Instalação das dependências necessárias para um PWA

Antes de converter o MistuRe em um PWA, foi preciso instalar bibliotecas específicas para ativar recursos como armazenamento em cache quando offline, o manifesto do aplicativo e alertas para o usuário. As principais dependências que foram incluídas foram:

- ❏ `vite-plugin-pwa`: Plugin oficial para Vite que simplifica a configuração de um PWA.
- ❏ `react-toastify`: Biblioteca para exibição de mensagens e notificações ao usuário.
- ❏ `typescript` (como dependência de desenvolvimento): Para suporte à tipagem e uso correto dos tipos em arquivos de configuração e no `virtual:pwa-register`.
- ❏ `Zustand`: Biblioteca que realiza gerenciamento de estados em aplicativos React
- ❏ `Tailwindcss`: Biblioteca de estilização rápida.

Listing D.1 – Instalação das dependências

```
# Plugin PWA para o Vite
npm install vite-plugin-pwa --save-dev
```

```
# Biblioteca Toastify
npm install react-toastify

# TypeScript (caso ainda nao esteja instalado no projeto)
npm install typescript --save-dev

# Zustand: gerenciamento de estados em aplicativos React
npm install zustand

# TailwindCSS: biblioteca de estilizacao rapida
npm install tailwindcss @tailwindcss/vite
```

D.5.3 Transformando o aplicativo em um PWA

Para transformar o MistuRe em um PWA, foi empregado o plugin oficial `vite-plugin-pwa`, que facilita a configuração do service worker e do manifest em projetos desenvolvidos com Vite.

Os principais passos foram:

1. Instalação do plugin:

```
npm install vite-plugin-pwa --save-dev
```

2. Configuração do plugin no arquivo `vite.config.js`:

Listing D.2 – Configuração do Vite para PWA

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react-swc";
import path from "path";
import { VitePWA } from "vite-plugin-pwa";

export default defineConfig({
  plugins: [
    react(),
    VitePWA({
      registerType: "autoUpdate",
      includeAssets: ["favicon.ico", "apple-touch-icon.png"],
      manifest: {
        name: "MistureApp",
        short_name: "Misture",
```

```
      description: "App de controle e consulta de substancias",
      theme_color: "#379276",
      background_color: "#ffffff",
      display: "standalone",
      start_url: "/",
      icons: [
        {
          src: "pwa-192x192.png",
          sizes: "192x192",
          type: "image/png",
        },
        {
          src: "pwa-512x512.png",
          sizes: "512x512",
          type: "image/png",
        },
        {
          src: "pwa-512x512.png",
          sizes: "512x512",
          type: "image/png",
          purpose: "any maskable",
        },
      ],
      screenshots: [
        {
          src: "screenshots/home.png",
          sizes: "1080x2340",
          type: "image/png",
          form_factor: "wide",
        },
      ],
    },
  }),
],
resolve: {
  alias: {
    "@": path.resolve(__dirname, "./src"),
  },
},
server: {
  proxy: {
    "/api": {
```

```
        target: "https://mistureapp.com.br/controller/",
        changeOrigin: true,
        secure: false,
        rewrite: (path) => path.replace(/^\/api/, ""),
      },
    },
  },
});
```

O trecho de código apresentado configura o plugin `vite-plugin-pwa`, responsável por transformar o projeto React em um PWA.

- A função `VitePWA` é invocada na lista de plugins do Vite.
- O parâmetro `registerType:'autoUpdate'` assegura que o worker de serviço seja atualizado automaticamente sempre que uma nova versão da aplicação estiver disponível.
- Em `includeAssets`, são indicados os arquivos estáticos que devem ser incorporados na construção (como o favicon e o robots.txt).
- O objeto `manifest` estabelece os dados fundamentais de um PWA, como o nome do aplicativo, ícones e cor do tema, que são empregados durante a instalação em dispositivos móveis.

Com essa configuração, foi habilitada a instalação do sistema no formato de um PWA.

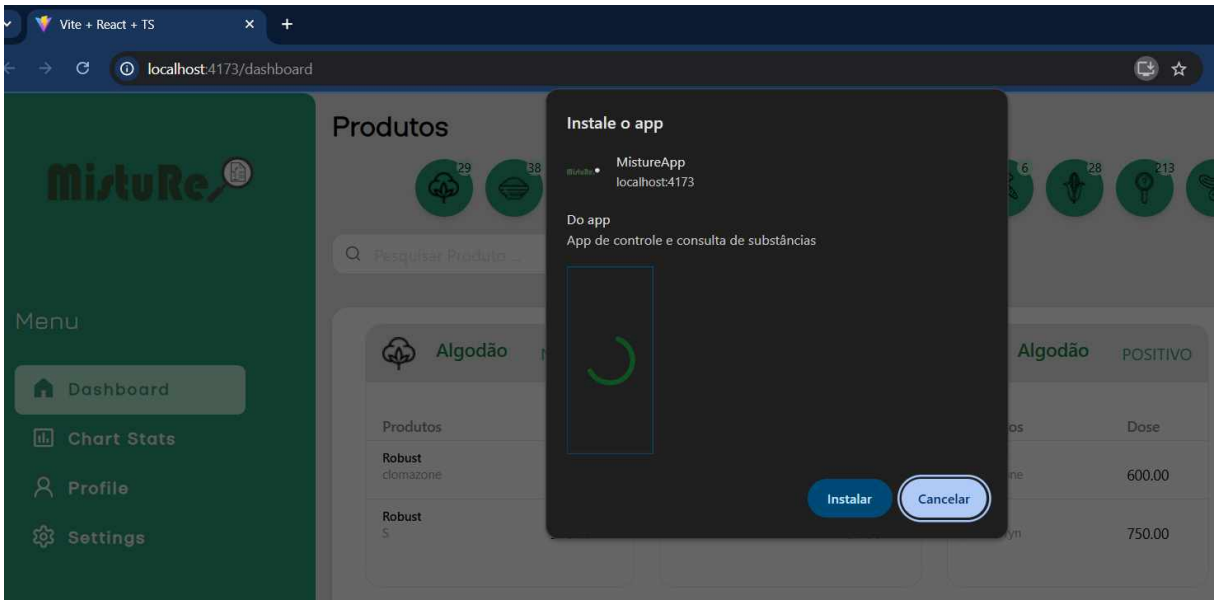


Figura 5 – Interface para instalar o sistema como um PWA. Fonte: Elaborado pelo autor

Todos os componentes da interface foram organizados em estruturas reutilizáveis em React, como ilustrado na Figura 6, garantindo modularidade e responsividade no sistema.

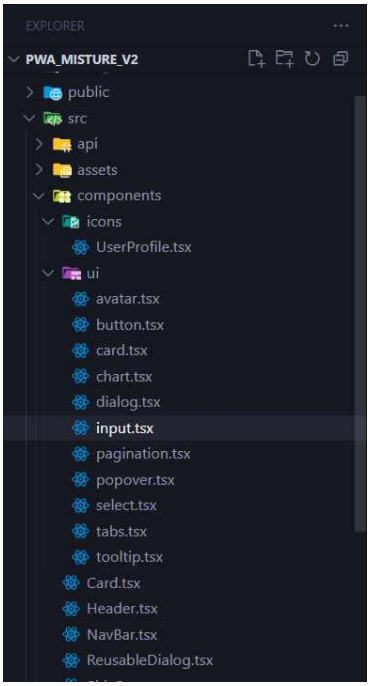


Figura 6 – Interface para instalar o sistema como um PWA. Fonte: Elaborado pelo autor

A Figura 7 mostra os detalhes do `Wokbox`, que funciona como uma caixa de ferramentas inteligente, destinada a armazenar elementos essenciais do aplicativo ou do site como arquivos HTML, CSS, JavaScript e imagens. Ele possibilita administrar a forma como o navegador salva esses arquivos em um cache, assegurando que sua aplicação funcione de forma ágil e até mesmo offline. Na figura 7, é possível observar a existência de dados parte central da figura com uma espécie de tabela dentro da estrutura *cache storage* do navegador listada na parte esquerda.

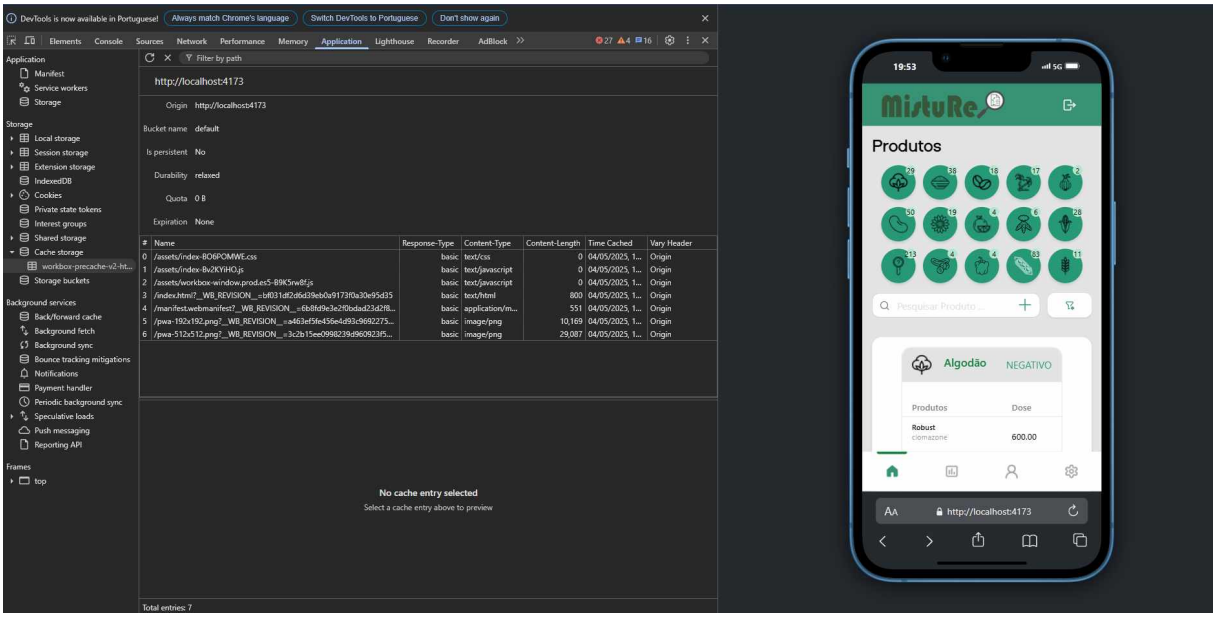
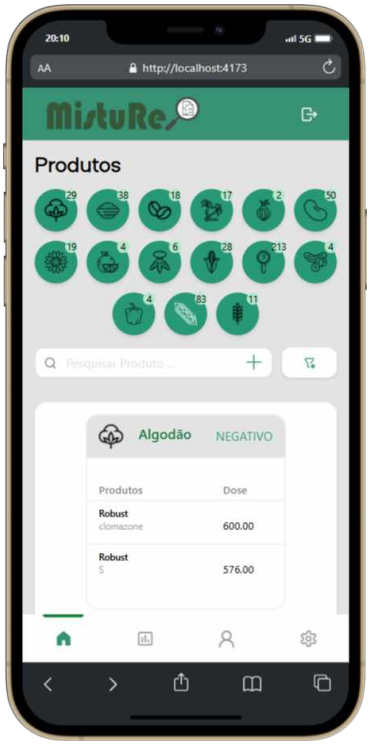
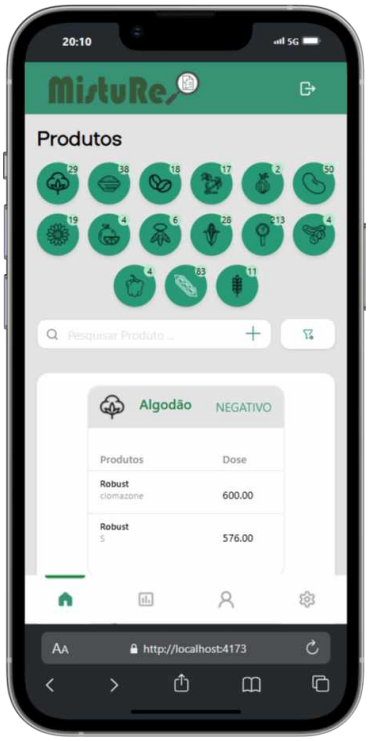


Figura 7 – Interface do Workbox evidenciando o cache ativo dos principais arquivos da aplicação MistuRe. Fonte: Elaborado pelo autor.

A Figura 8 mostra a exibição do sistema MistuRe configurado como um PWA em várias marcas de celulares, evidenciando sua flexibilidade e capacidade de adaptação em diferentes resoluções.



(a) iPhone 12 Pro Max
Resolução: 1284 x 2778 pixels. Fonte: Elaborado pelo autor



(b) iPhone 13 Pro Max
Resolução: 1284 x 2778 pixels. Fonte: Elaborado pelo autor

(c) Galaxy S21 Ultra
Resolução: 1440 x 3200 pixels. Fonte: Elaborado pelo autor

(d) Xiaomi 12
Resolução: 1080 x 2400 pixels. Fonte: Elaborado pelo autor

Figura 8 – Visualização do sistema MistuRe em diferentes telas. Fonte: Elaborado pelo autor

D.6 Desafios Encontrados

Durante o desenvolvimento, foram identificados alguns desafios:

- ❑ **Compatibilidade com Safari/iOS**: Alguns recursos como notificações push e armazenamento em cache apresentaram comportamento limitado neste navegador.
- ❑ **Performance inicial**: A interface principal mostrou um longo tempo de carregamento, necessitando de ajustes no uso da API de maneira mais eficiente.
- ❑ **Adaptação da lógica de busca**: A lógica de consulta de produtos teve de ser adaptada para funcionar em um ambiente web responsivo.

Quando comparado uma aplicação PWA, com um aplicativo nativo, é válido destacar algumas limitações. No iOS, por exemplo, os PWAs ainda apresentam restrições no suporte a notificações *push*, o que afeta a comunicação em tempo real com o usuário. Ademais, há limitações na personalização de ícones e no uso de funcionalidades específicas do sistema, devido às restrições da *Apple*.

Já no Android, o suporte aos PWAs é mais avançado. O sistema permite notificações*push*, instalação na tela inicial com ícone personalizado e até mesmo funcionamento *offline*. No entanto, ainda assim os PWAs não têm acesso total aos recursos nativos do dispositivo, como sensores avançados, Bluetooth, NFC, integração com contatos, calendário e execução em segundo plano com a mesma eficiência dos aplicativos nativos.

Em suma, embora os PWAs ofereçam uma solução mais acessível e multiplataforma, eles ainda perdem em termos de desempenho, acesso a recursos do sistema e integração profunda com o dispositivo, especialmente quando comparados com aplicativos desenvolvidos nativamente para Android e iOS.

D.7 Considerações

A criação da versão PWA do MistuRe fez o sistema funcionar em várias plataformas com menos custo, aumentando o acesso para quem usa dispositivos com iOS e outros sistemas. O uso de ferramentas atuais como Vite e *React*, junto com métodos de cache, foi importante para chegar em uma solução que atende as normas/premissas de PWA e com fluidez do ponto de vista da experiência de uso.

Os produtos e misturas ficam armazenados em um *useState*, uma função do *React*, toda vez que é reiniciado, essa página recarrega esse estado.

Apêndice E

Testes e Discussão dos Resultados

Este capítulo apresenta os métodos utilizados para avaliar a proposta desenvolvida, os testes realizados e a análise dos resultados obtidos. A intenção é validar a aplicação MistuRe como um PWA, observando aspectos técnicos e práticos do seu funcionamento em diferentes dispositivos e cenários.

E.1 Método para a Avaliação

A análise do MistuRe como PWA visou identificar se as expectativas geradas pela implementação dessa tecnologia estavam sendo de fato viabilizadas. Desse modo, foram empregados métodos que possibilitaram avaliar o desempenho da aplicação em propriedades, como: operação offline, utilização de cache, responsividade e conformidade às boas práticas de desenvolvimento para a web.

Funcionamento do PWA

A aplicação foi criada utilizando o *framework* Vite em conjunto com o React, empregando o *plugin* `vite-plugin-pwa`. Este plugin tem como objetivo principal gerar automaticamente um *service worker*, o qual é encarregado de controlar o cache da aplicação. Com esse *service worker*, a aplicação pode operar normalmente mesmo sem conexão à internet e também propicia um aumento na eficiência, em função do carregamento aprimorado de recursos já salvos.

A configuração do aplicativo contempla o registro automático do *service worker* utilizando a opção `registerType: "autoUpdate"`, assegurando que, sempre que uma nova versão do aplicativo esteja disponível, o navegador atualize o *service worker* de maneira transparente para o usuário.

Ao longo dos testes, foi observado, através da aba "Application" nas ferramentas para desenvolvedores do Google Chrome, que:

- O *service worker* foi registrado, ativado e está operando, conforme mostrado na seção “Service Workers”;
- O aplicativo possui uma seção de Cache Storage em funcionamento, onde são guardados arquivos fundamentais, como HTML, Folha de Estilo em Cascata (CSS), JavaScript e imagens, demonstrando a operação do cache através da ferramenta Workbox.

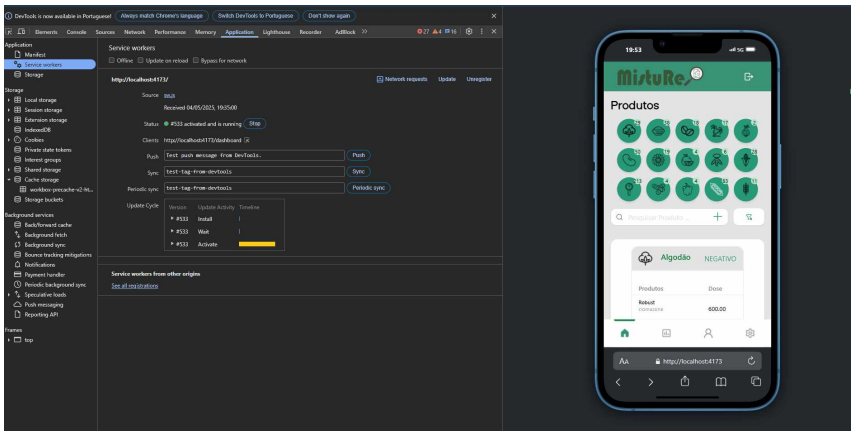


Figura 9 – Registro e funcionamento do *service worker*. Fonte: Elaborado pelo autor

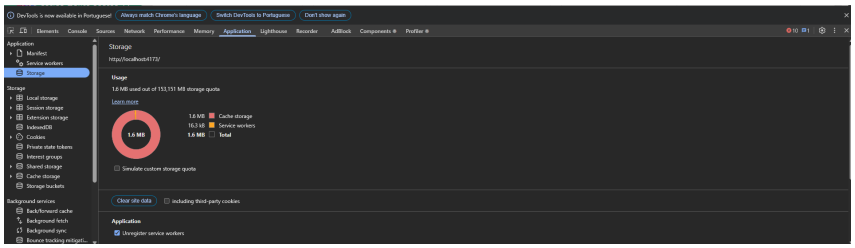


Figura 10 – Cache Storage funcionando com arquivos principais. Fonte: Elaborado pelo autor

Avaliação com a ferramenta Lighthouse

Além da inspeção manual, foi empregada a ferramenta Google Lighthouse para efetuar uma análise técnica automatizada da aplicação. O Lighthouse possibilita a avaliação de diversos parâmetros, como: performance, acessibilidade, boas práticas de desenvolvimento, adequação aos critérios de uma PWA.

A avaliação foi realizada na versão mobile do aplicativo, levando em conta o perfil do público-alvo. Dois aspectos principais foram analisados: a tela de login e a interface principal. Os resultados dos testes indicaram que a aplicação é identificada como uma PWA legítima e que o `manifest.json` e o service worker estavam devidamente configurados.

O desempenho geral da interface principal atingiu 46%, de acordo com a pontuação fornecida pelo Lighthouse. Esse resultado sugere que há espaço para melhorias, especialmente em relação ao tempo de carregamento e à utilização de recursos no carregamento

inicial da aplicação. Contudo, os outros critérios analisados apresentaram um desempenho satisfatório e afirmaram o funcionamento adequado dos elementos esperados em uma aplicação do tipo PWA.

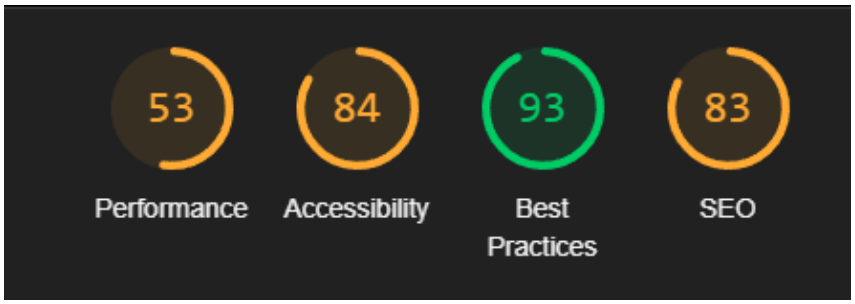


Figura 11 – Relatório Lighthouse. Fonte: Elaborado pelo autor

Destaques:

- ❏ Avaliação de desempenho (53%);
- ❏ Informações como a duração do carregamento, os primeiros bytes recebidos e recomendações para otimização.

Conclusão da Avaliação

O estudo revelou que a ferramenta MistuRe cumpre os principais critérios de uma PWA, incluindo instalação, operação *offline* e compatibilidade multiplataformas. A utilização do plugin `vite-plugin-pwa` provou ser eficaz na criação e administração do *service worker* de maneira automatizada, além de facilitar o armazenamento em cache inteligente por meio do Workbox.

Os dados coletados pelo Lighthouse destacam a solidez da aplicação como PWA, porém, também destacam a necessidade de aprimoramentos em sua performance, especialmente na interface principal. Estas melhorias serão levadas em conta nas versões futuras, com o objetivo de proporcionar ao usuário uma experiência mais fluida e eficiente.

E.2 Cenários de Avaliação

Para validar a eficácia prática do aplicativo MistuRe como uma PWA, foram conduzidos testes práticos em dispositivos reais, simulando diversas situações de uso possíveis. O objetivo principal dos testes foi avaliar o desempenho do aplicativo nos seguintes aspectos: Instalação na tela inicial; Funcionamento offline após o primeiro acesso; Compatibilidade com diversos navegadores e sistemas operacionais; Agilidade na resposta e navegação; Responsividade da interface.

Ambiente de Testes

Os testes foram efetuados em dispositivos com sistemas operacionais variados, visando assegurar a compatibilidade multiplataforma que a arquitetura PWA propõe:

- ❏ Android (Samsung Galaxy S23 Fe) utilizando o navegador Google Chrome;
- ❏ iOS (iPhone 13) acessando o navegador Safari;
- ❏ Usando os browsers Google Chrome e Microsoft Edge no computador de mesa (Windows 11). Fonte: Elaborado pelo autor

Além das avaliações em dispositivos móveis, a aplicação também passou por testes em computadores desktop por meio do navegador, para confirmar a adaptabilidade da interface e a correta configuração dos componentes na tela.

Cenários Avaliados

Durante o processo de testes, foram analisados os seguintes cenários:

1. **Instalação do aplicativo:** Foi checado se a aplicação apresentava o *prompt* de instalação no navegador, permitindo que fosse adicionada à tela inicial em dispositivos Android e iOS.
2. **Uso offline:** Após o completo carregamento inicial com conexão à internet, o dispositivo foi colocado em modo avião. A aplicação conseguiu manter parte de suas funcionalidades ativas, evidenciando que os arquivos essenciais estavam devidamente armazenados em cache.
3. **Teste de navegação:** As principais seções da aplicação, como a tela de consulta, filtros e visualização de resultados, foram acessadas para avaliar a agilidade na resposta e a fluidez da navegação.
4. **Compatibilidade entre navegadores:** O acesso à aplicação foi feito nos navegadores Chrome, Firefox e Safari. Em todas as situações, os elementos visuais e interativos preservaram sua disposição e funcionalidade, validando assim a responsividade e compatibilidade esperadas.
5. **Simulação de atualizações:** Com a configuração de `autoUpdate` ativada no `vite-plugin-pwa`, a aplicação foi atualizada e reavaliada nos dispositivos. O *service worker* detectou a nova versão e realizou a atualização automática no navegador, conforme o esperado.

Nos testes realizados, notou-se que a aplicação apresentou um rendimento adequado na maioria dos navegadores e aparelhos, com destaque para o Chrome para Android. Contudo, foi possível notar algumas limitações específicas do Safari para iOS, como a ausência de suporte total para notificações *push* e algumas restrições relacionadas ao armazenamento em cache.

E.3 Avaliação dos Resultados

Com base nos testes realizados e nas informações obtidas através de ferramentas de análise e experimentos práticos, foi possível examinar o desempenho da aplicação PWA em diversas áreas.

Em geral, os resultados foram favoráveis em termos de compatibilidade entre navegadores, instalação do aplicativo, funcionalidade offline e funcionamento do cache através do *service worker*, o que confirma a viabilidade da proposta de converter o sistema MistuRe em uma aplicação multiplataforma fundamentada em tecnologias web.

A aplicação cumpriu os requisitos essenciais de um PWA:

- ❏ Registro automático e operação do *service worker*;
- ❏ Funcionalidade de cache com arquivos guardados via Workbox;
- ❏ Manifesto configurado corretamente;
- ❏ Opção de instalação na tela inicial;
- ❏ Capacidade de funcionar offline depois do primeiro carregamento.

A avaliação através do Google Lighthouse enfatizou esses aspectos ao classificar a aplicação como uma PWA válida. No entanto, o desempenho (56% na interface principal) evidenciou a necessidade de melhorias em aspectos como o tempo de carregamento, a renderização e a otimização de *scripts*.

Nos testes realizados em aparelhos móveis e desktops, a aplicação demonstrou boa adaptabilidade, contudo, algumas restrições foram notadas no navegador Safari/iOS. Essas questões são identificadas como limitações do sistema, e não do aplicativo em si.

Essas avaliações indicam que, mesmo possuindo uma base técnica robusta e uma aplicação em pleno funcionamento, existem possibilidades para progressos em termos de desempenho e funcionalidades específicas, especialmente na procura por uma experiência mais parecida com a de aplicativos nativos.

Apêndice F

Conclusão

Este capítulo apresenta as conclusões obtidas a partir do desenvolvimento e da análise da aplicação MistuRe como um Aplicativo Web Progressivo (PWA). Retomando os objetivos geral e específicos definidos no início do trabalho, é possível observar que a proposta foi concretizada com êxito. O objetivo geral de tornar a aplicação MistuRe uma solução multiplataforma acessível, especialmente para produtores rurais, foi alcançado com a implementação de tecnologias modernas da web.

Entre os objetivos específicos, destacaram-se a compatibilidade com diferentes sistemas operacionais e navegadores, o suporte ao uso offline por meio de *cache* e *service worker*, e a facilidade de instalação como um aplicativo nativo. Todos esses aspectos foram validados com testes práticos e pelo uso de ferramentas como o Lighthouse, que confirmaram a conformidade da aplicação com os critérios de um PWA. Assim, conclui-se que a abordagem adotada foi eficaz em atender tanto aos requisitos técnicos quanto às necessidades do público-alvo.

Ademais, a graduação em Sistemas de Informação contempla o desenvolvimento técnico de profissionais da área, abordando temáticas que possibilitam tornar esse trabalho viável. Durante o curso, aspectos como o desenvolvimento web, mobile e o uso de tecnologias e diferentes tipos de navegadores são amplamente explorados, o que proporcionou uma base sólida para a criação deste projeto.

F.1 Principais Contribuições

A principal meta deste projeto foi tornar a aplicação MistuRe acessível como um PWA, atendendo a diversos sistemas operacionais, com um enfoque particular em dispositivos móveis utilizados por agricultores. Por meio do uso do plugin `vite-plugin-pwa` e da estrutura moderna com React e Vite, foi possível criar uma solução multiplataforma sem precisar desenvolver versões separadas para Android e iOS.

As principais contribuições deste projeto incluem: a implementação de um *service worker* para funcionamento offline; a compatibilidade com variados navegadores e dispo-

sitivos; a utilização de cache inteligente por meio do Workbox; e a redução de custos e barreiras técnicas na distribuição do sistema.

Além disso, foi utilizada a ferramenta Workbox na aplicação MistuRe para permitir seu funcionamento offline por meio de uso de cache. Essa estratégia assegura que arquivos essenciais — como HTML, CSS, JavaScript e imagens — sejam armazenados localmente após o primeiro carregamento.

Adicionalmente, o estudo evidencia o uso de tecnologias web modernas como uma alternativa viável para desenvolver soluções robustas e acessíveis no setor agrícola, mostrando um grande potencial para impactar a rotina dos produtores rurais.

F.2 Trabalhos Futuros

Durante a fase de desenvolvimento e testes da aplicação, foram detectadas oportunidades de melhorias e possíveis direções de evolução: otimização do desempenho, visando a diminuição do tempo de carregamento; inclusão de notificações push para alertas importantes (com limitações no iOS); melhoria da experiência offline em rotas dinâmicas (*fallbacks* e pré-carregamento inteligente); integração com banco de dados local (IndexedDB) para uma persistência offline mais eficiente; e ampliação da base de dados com novos produtos e misturas reconhecidas.

Essas melhorias podem incrementar a usabilidade da aplicação, solidificando o MistuRe como uma referência no setor.

Referências

ARAÚJO, J. V. da S. **Progressive enhancement (Aprimoramento Progressivo)**. 2020. Disponível em: <https://pt.linkedin.com/pulse/progressive-enhancementaprimoramento-progressivo-jhonata>. Citado na página 15.

AVRELA, I. **Experiência de aplicativo nativo, com linguagem de web**. 2020. Disponível em: <https://pt.linkedin.com/pulse/confi%C3%A1vel-engajante-e-r%C3%A1pido-pwa-experi%C3%AAncia-de-nativo-ionara-avrela>. Citado na página 16.

BANKS, A.; PORCELLO, E. **Learning React: Modern Patterns for Developing React Apps**. [S.l.]: O'Reilly Media, 2020. Citado na página 24.

CARDOSO, C. H. M.; SANTOS, E. A.; ARAÚJO, R. D. Misture: Uma plataforma para unificação de dados científicos sobre compatibilidade de produtos em misturas de tanque com calda herbicida. In: SBC. **Anais Estendidos do XVII Simpósio Brasileiro de Sistemas de Informação**. [S.l.], 2021. p. 181–184. Citado 4 vezes nas páginas 10, 11, 17 e 19.

CARMARGO, J. **Conceito entre aplicativos e site: futuro da tecnologia PWA - Progressive Web Apps**. 2019. Disponível em: <https://ipnews.com.br/artigo-conceito-entre-app-e-site-e-futuro-da-tecnologia-pwa/>. Citado na página 15.

CECONI, L. Experiência do usuário em progressive web apps. 2019. Citado na página 16.

COLLECTIVE, P. **Zustand: A Bear Necessities State Management Tool**. 2024. Disponível em: <https://docs.pmnd.rs/zustand/getting-started/introduction>. Citado na página 25.

COSTA, T. F.; PIRES, F. Utilização de progressive web apps para desenvolvimento de aplicações para dispositivos móveis. **Revista Integralização Universitária**, n. 19, p. 72–83, 2018. Citado na página 18.

FREEMAN, E. **Use a Cabeça! HTML e CSS**. São Paulo: Alta Books, 2015. Citado na página 24.

GAUNT, M. **Service worker overview**. 2019. Disponível em: <https://developer.chrome.com/docs/workbox/service-worker-overview/>. Citado 2 vezes nas páginas 14 e 25.

GAZZIERO, D. Misturas de agrotóxicos em tanque nas propriedades agrícolas do brasil. **Planta Daninha**, SciELO Brasil, v. 33, p. 83–92, 2015. Citado na página 9.

GEHRING, A. G. d. M. G.; ROSSETTO. Progressive web apps: Análise comparativa de frameworks javascript para desenvolvimento. **Passo Fundo**, p. 1–20, 2021. Citado 3 vezes nas páginas 13, 14 e 17.

HAMEL, A. du; CONTRIBUTORS. **Vite Plugin PWA**. 2024. Disponível em: <https://vite-pwa-org.netlify.app>. Citado na página 25.

KHADRA, F. **React Toastify**. 2024. Disponível em: <https://fkhadra.github.io/react-toastify/introduction>. Citado na página 24.

LABS, T. **Tailwind CSS Documentation**. 2024. Disponível em: <https://tailwindcss.com/docs>. Citado na página 25.

MACCALI, G. Progressive web apps: um novo paradigma de desenvolvimento frontend. 2021. Citado 4 vezes nas páginas 13, 15, 16 e 17.

MOZILLA. **Web App Manifest**. 2022. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/Manifest>. Citado na página 16.

NARDON, J. K. **6 diferenças entre desenvolver em Android e iOS**. 2020. Disponível em: <https://cwi.com.br/blog/diferencas-entre-desenvolver-em-android-e-ios>. Citado na página 11.

PATEL, N. **Web App Manifest**. 2019. Disponível em: <https://neilpatel.com/br/blog/pwa-o-que-e/>. Citado 2 vezes nas páginas 11 e 14.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software-9**. [S.l.]: McGraw Hill Brasil, 2021. Citado na página 19.

ROSENWASSER, D. **TypeScript Handbook**. 2022. <https://www.typescriptlang.org/docs/handbook/intro.html>. Citado na página 25.

RUSSEL, F. A.; BERRIMAN. Progressive web apps: Escaping tabs without losing our soul. **Infrequently Noted**, 2015. Citado 2 vezes nas páginas 14 e 15.

SHADCN. **shadcn/ui - Componentes de UI para React**. 2024. Disponível em: <https://ui.shadcn.dev>. Citado na página 24.

SILVA, G. de S. Mistura de agrotóxicos e tecnologia de aplicação. **Programa de PósGraduação em Fitotecnia da Universidade Federal Rural do Rio de Janeiro**, v. 1, n. 3, 2021. Citado na página 10.

SILVA, J. K.; TIOSSO, F. Revisao bibliografica sobre conceito de progressive web applications (pwa). **Revista Interface Tecnológica**, v. 17, n. 1, p. 53–64, 2020. Citado 2 vezes nas páginas 10 e 18.

SOUZA, C. B. C. d.; CINTRA, F. G. Pwa: Tecnologia da informação a serviço de pequenas empresas. 109, 2018. Citado 2 vezes nas páginas 16 e 18.

VITE. **Vite: Next Generation Frontend Tooling**. 2024. Disponível em: <https://vitejs.dev>. Citado na página 25.