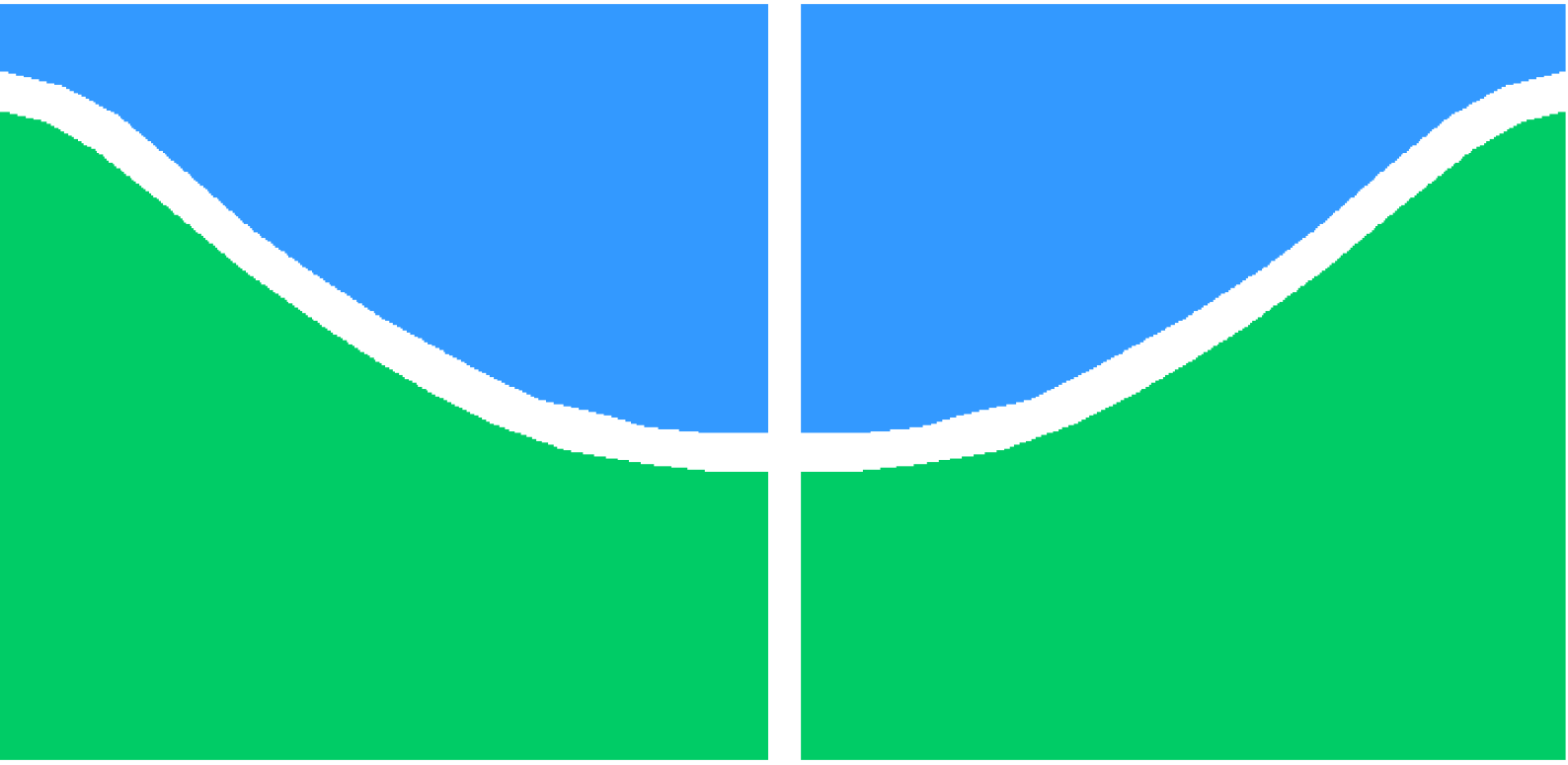

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Renderização de Fluxos de uma Aplicação de Educação Jurídica em Dispositivos Mobile

Autor: João Pedro Alves Machado e Júlio César Martins França

Orientador: Prof. Dr. André Luiz Peron Martins Lanna

Brasília, DF

2024

João Pedro Alves Machado e Júlio César Martins França

Renderização de Fluxos de uma Aplicação de Educação Jurídica em Dispositivos Mobile

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. André Luiz Peron Martins Lanna

Brasília, DF

2024

João Pedro Alves Machado e Júlio César Martins França
Renderização de Fluxos de uma Aplicação de Educação Jurídica em Dispositivos Mobile/ João Pedro Alves Machado e Júlio César Martins França. – Brasília, DF, 2024-
41 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. André Luiz Peron Martins Lanna

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2024.

1. fluxo-juridico. 2. aplicacao-web-mobile. I. Prof. Dr. André Luiz Peron Martins Lanna. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Renderização de Fluxos de uma Aplicação de Educação Jurídica em Dispositivos Mobile

CDU 02:141:005.6

João Pedro Alves Machado e Júlio César Martins França

Renderização de Fluxos de uma Aplicação de Educação Jurídica em Dispositivos Mobile

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 28 de Outubro de 2024:

Prof. Dr. André Luiz Peron Martins Lanna

Prof. Me. Cristiane Soares Ramos

Prof. Me. Ricardo Ajax Dias Kosloski

Brasília, DF
2024

Resumo

Este trabalho aborda o desenvolvimento de uma aplicação web progressiva (PWA) destinada a facilitar o acesso à informação jurídica, com foco na educação sobre direitos trabalhistas. A aplicação permite que os usuários naveguem por fluxogramas interativos, organizados em perguntas e respostas, simplificando a compreensão de temas jurídicos complexos. Utilizando tecnologias como Next.js, AWS Lambda e JSON para a renderização de fluxos, o sistema oferece uma solução acessível, compatível com dispositivos móveis e que funciona tanto online quanto *offline*. A proposta busca democratizar o conhecimento jurídico e promover maior inclusão social, especialmente para pessoas sem formação na área.

Palavras-chave: Aplicação web, Educação jurídica, PWA, Direitos trabalhistas, Fluxogramas.

Abstract

This work focuses on the development of a Progressive Web Application (PWA) aimed at facilitating access to legal information, particularly related to labor rights education. The application allows users to navigate interactive flowcharts, structured as questions and answers, simplifying the understanding of complex legal topics. Utilizing technologies such as Next.js, AWS Lambda, and JSON for flow rendering, the system provides an accessible solution, compatible with mobile devices, and functional both online and offline. The project seeks to democratize legal knowledge and promote greater social inclusion, especially for individuals without legal training.

Key-words:Web application, Legal education, PWA, Labor rights, Flowcharts.

Lista de ilustrações

Lista de abreviaturas e siglas

TCC	Trabalho de Conclusão de Curso
JSON	*JavaScript Object Notation*
API	*Application Programming Interface*
XP	*Extreme programming*
CI	*Continuos Integration*
CD	*Continuos Delivery*
CSS	*Cascading Style Sheets*
HTML	*Hypertext Markup Language*
SQL	*Structured Query Language*
AWS	*Amazon Web Services*
API	*Application Programming Interface*
PWA	*Progressive Web App*
HTTP	*Hypertext Transfer Protocol*
TBD	*Trunk-Based Development*

Sumário

Parte I

Introdução

1 Introdução

É crucial que os cidadãos de uma sociedade tenham pleno conhecimento de seus direitos e deveres, ambos assegurados por lei. No Brasil, não é diferente. Contudo, devido ao baixo grau de instrução em grande parcela da sociedade, muitas pessoas não têm ciência do alcance real de seus direitos e deveres. Isso faz com que seu conhecimento jurídico seja fraco ou inexistente em várias áreas, tornando-as mais suscetíveis a situações que nao ocorreriam se tivessem conhecimento pleno dos seus direitos em determinado contexto.

Como destaca (FREIRE, 1996) em sua obra "Pedagogia da Autonomia"(1996), a educação de qualidade é fundamental para o desenvolvimento do senso crítico e da autonomia do indivíduo. No contexto jurídico, isso se traduz na capacidade de compreender e exercer seus direitos e deveres, buscando soluções para seus problemas e contribuindo para uma sociedade mais justa e igualitária, onde o cumprimento dos deveres fortalece os vínculos sociais e promove o bem-estar coletivo, essencial para a construção de uma sociedade equilibrada e igualitária . No âmbito jurídico, é fundamental não apenas conhecer os direitos e deveres, mas também compreender como o sistema legal funciona e os mecanismos de acesso à justiça, sendo essencial a participação ativa dos cidadãos na busca por soluções que promovam uma sociedade mais justa e democrática. Conforme abordado em (Politize!, 2024), o princípio constitucional do acesso à justiça, garantido pelo artigo 5º da Constituição Federal de 1988, assegura que todos possam recorrer ao Judiciário para proteger seus direitos, promovendo uma justiça acessível e equitativa, o que é essencial para a pacificação social e o exercício pleno da cidadania.

1.1 Problema e Contexto da Solução

A maior parte do conteúdo jurídico ainda é transmitida de maneira tradicional, por meio de textos longos e complexos, o que dificulta o acesso a esses conhecimentos por pessoas fora da área, devido ao uso de linguagem técnica e de difícil compreensão para quem não possui preparo adequado. Esse cenário acentua a necessidade de uma plataforma que torne o ensino jurídico mais inclusivo, eliminando barreiras ao aprendizado por meio de ferramentas interativas e intuitivas, que facilitem a compreensão e democratizem o acesso ao conhecimento jurídico.

A solução proposta envolve o desenvolvimento de uma aplicação mobile que complementa um sistema previamente construído nesta parceria, permitindo a integração de fluxos de navegação baseados em perguntas e respostas, com sua renderização em telas de dispositivos móveis ou navegadores. Esses fluxos interativos visam proporcionar um aprendizado mais envolvente e acessível, eliminando a necessidade de conhecimento téc-

nico para criar e acessar os conteúdos. Além disso, o uso de dispositivos móveis amplia o alcance e a flexibilidade do ensino, permitindo que os usuários aprendam em qualquer lugar e a qualquer momento. Dessa forma, a aplicação mobile responde à crescente demanda por métodos educacionais modernos, focados em acessibilidade e praticidade, alinhados à era digital e às expectativas de usuários que valorizam soluções interativas e intuitivas.

1.2 Impactos da Solução

A dificuldade no entendimento de termos jurídicos e a linguagem técnica utilizada nas leis são questões recorrentes que afetam o acesso à justiça para muitas pessoas. De acordo com (RANDALL, 2024), o uso de "legalês"e a apresentação de textos densos e complexos podem tornar informações básicas inacessíveis para aqueles que não possuem formação na área jurídica. Isso cria barreiras significativas, impedindo que o público em geral compreenda seus direitos e participe de forma ativa no sistema legal. Simplificar a linguagem jurídica é uma forma de tornar o sistema mais inclusivo e acessível, permitindo que todos, independentemente de sua formação, possam lidar com questões jurídicas com maior independência e desenvolver uma capacidade analítica mais robusta para avaliar as situações.

A aplicação mobile visa tornar a educação jurídica mais acessível e simples. Ao renderizar telas interativas, facilita a compreensão de questões legais tanto para profissionais quanto para o público geral. A plataforma amplia o alcance da educação, permitindo o acesso a qualquer momento e lugar, promovendo maior inclusão e eficiência no aprendizado. Além disso, o uso de aplicativos mobile para difusão de conhecimento tem grande impacto, visto que as pessoas têm mais facilidade em aprender temas complexos quando apresentados de forma didática e interativa, como demonstrado por (Mobile App Daily, 2024) que indicam que ferramentas digitais aumentam o engajamento e a retenção de informações, tornando o aprendizado mais eficaz e acessível para todos.

1.3 Objetivos

1.3.1 Objetivos Gerais

O objetivo geral deste trabalho é o desenvolvimento de uma aplicação *web* que apresente renderização de telas em dispotivos *mobile* de acordo com a estrutura e os elementos oriundos de um ou mais fluxogramas.

1.3.2 Objetivos Específicos

- **Implementação do *Design*:**

- Utilizar a identidade visual definida no protótipo de alta fidelidade.

- Garantir navegação fluída entre as telas, conforme as respostas na seção.

- Permitir que seja buscado um Fluxo específico para ser renderizado.

- **Requisição e Renderização de Fluxogramas:**

  - Criar um serviço para buscar a lista de fluxogramas disponíveis na aplicação que disponibiliza os fluxo em formato especifoco, através de uma requisição em um serviço de função *lambda* da AWS.

  - Fazer requisições para a aplicação que disponibiliza os fluxos para obter os dados em formato específico e armazená-los localmente.

  - Atualizar os fluxogramas consultados sempre que tiver acesso a conexão e renderizar com a tela baseado nos novos conteúdos.

- **Seção Informativa:**

  - Informar que o aplicativo não substitui um advogado.

  - Esclarecer que o aplicativo não garante embasamento legal, apenas renderiza fluxos.

  - Oferecer recomendações para aprofundamento em temas de interesse.

  - Incluir um campo de ajuda, se o formulário original que gerou o fluxo possuir essa opção.

1.4 Estrutura do Texto

Capítulo 2 - Referencial Teórico: Apresenta os principais conceitos e a base teórica necessária para o entendimento do projeto.

Capítulo 3 - Suporte Tecnológico: Detalha as tecnologias escolhidas e justifica seu uso no desenvolvimento da solução.

Capítulo 4 - Metodologia: Descreve os métodos e processos adotados para a execução do projeto.

Capítulo 5 - Resultados: Mostra os resultados obtidos, com exemplos práticos da solução desenvolvida.

Capítulo 6 - Considerações Finais: Apresenta as conclusões e sugere direções para futuros trabalhos com base nos resultados.

2 Referencial Teórico

2.1 Engenharia de *Software*

A Engenharia de *Software* é uma disciplina fundamental no desenvolvimento de *Software*, aplicando princípios, técnicas e métodos científicos para garantir a entrega de produtos de qualidade, eficientes e confiáveis (IEEE Computer Society, 2006). Esta área de estudo abrange todas as fases do ciclo de vida do *Software*, desde a análise inicial de requisitos até a manutenção contínua do sistema.

Um dos aspectos centrais da Engenharia de *Software* é a sistemática aplicação de métodos e técnicas. Isso implica em abordagens estruturadas para o desenvolvimento, garantindo que cada etapa do processo seja conduzida de maneira ordenada e eficiente. A qualidade do *Software* é geralmente vista como um conjunto de atributos, como confiabilidade e desempenho, que impactam diretamente a experiência do usuário e a viabilidade do *Software* a longo prazo. Como é apresentado em (KEKRE; KRISHNAN; SRINIVASAN, 1995), a satisfação do cliente depende tanto do bom funcionamento do *Software* quanto da sua capacidade de atender às expectativas em termos de confiabilidade e eficiência.

Além das atividades técnicas, a Engenharia de *Software* também lida com a gestão de projetos. Segundo (O'REGAN, 2019), isso envolve o planejamento cuidadoso, a alocação eficiente de recursos e o controle de qualidade ao longo de todo o processo de desenvolvimento. Adotar processos e práticas adequadas é essencial para o sucesso de projetos em Engenharia de *Software*, contribuindo não apenas para a entrega eficiente, mas também para a satisfação dos clientes.

Por meio da aplicação consistente de seus princípios, a Engenharia de *Software* não apenas assegura o sucesso dos projetos individuais, mas também promove o avanço tecnológico e a inovação na área de *Software* como um todo. Essa constante busca por melhorias e inovações é fundamental para o desenvolvimento contínuo da Engenharia de *Software* como disciplina e para a evolução do campo da tecnologia da informação.

2.2 Desenvolvimento *Web*

O desenvolvimento web abrange diversas metodologias, cada uma com suas características específicas. A pesquisa destaca que muitas empresas optam por abordagens híbridas (MOLINA-RíOS; PEDREIRA-SOUTO, 2020), devido a flexibilidade ao ganhar características das duas tecnologias. A escolha da metodologia é crucial e depende de fatores como o tamanho do projeto, prazos de entrega, parâmetros de qualidade e *exper-*

*tise*da equipe. Dimensionar e escolher a metodologia adequada é essencial para o sucesso do projeto, garantindo eficiência, qualidade e alinhamento com as necessidades específicas da equipe e do contexto do desenvolvimento.

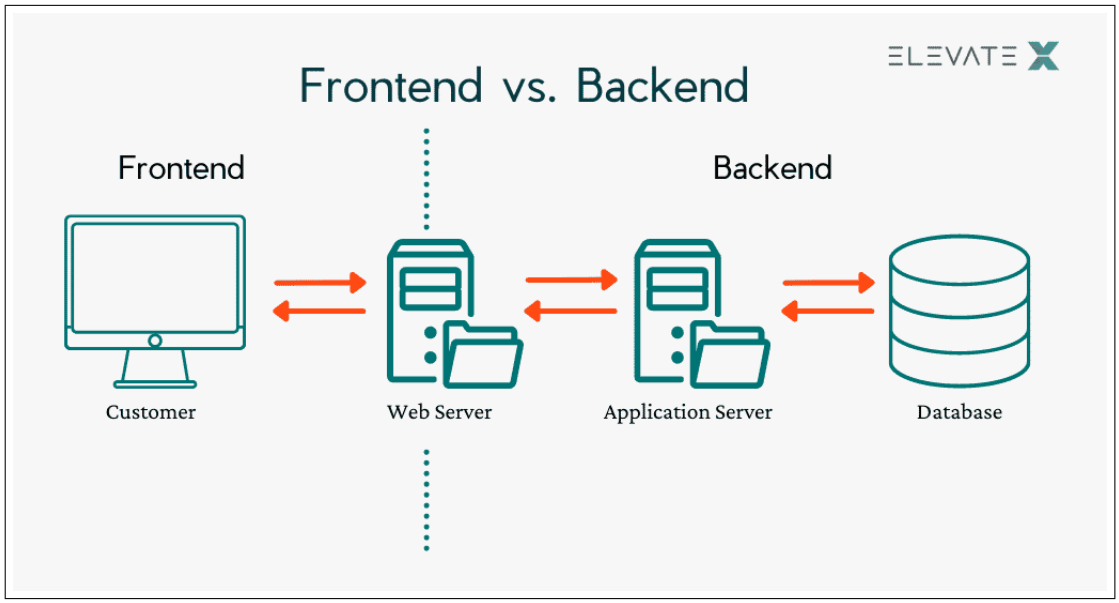

Figura 1 – *Frontend* vs *Backend* (ELEVATEX, 2022)

Frontend: Refere-se à interface do usuário, a parte da aplicação com a qual os usuários interagem diretamente.Ele é responsável pela interface gráfica e pela experiência do usuário (UI/UX), sendo composto por três principais tecnologia:

- HTML (*Hypertext Markup Language*): É a linguagem de marcação usada para estruturar o conteúdo da página, definindo elementos como textos, imagens, títulos e links. Ele estabelece a base sobre a qual a página é construída, criando a hierarquia de informações.

- CSS (*Cascading Style Sheets*): CSS define a aparência visual do conteúdo criado com HTML, permitindo estilizar elementos, como cores, tipografia, espaçamento e *layouts*. Ele possibilita o design responsivo, ajustando a aparência da página em diferentes dispositivos (como celulares e tablets).

-JavaScript: Essa linguagem de programação torna a página web interativa. JavaScript adiciona funcionalidades dinâmicas, como *sliders*, animações, validação de formulários e comunicação com o *backend*, tornando a experiência do usuário mais fluida e interativa.

Backend: O *Backend* é a parte da aplicação que lida com a lógica de negócios, manipulação e processamento de dados, além de interagir com o banco de dados. Ao contrário do *frontend*, que se preocupa com a interface do usuário, o *backend* gerencia o que acontece "nos bastidores"de uma aplicação. Ele processa requisições enviadas pelo

frontend, realiza cálculos, recupera ou armazena informações no banco de dados e envia as respostas de volta ao cliente.

Banco de Dados: Armazena e gerencia os dados da aplicação. Pode ser relacional (*SQL*) ou não relacional (*NoSQL*). Os bancos de dados relacionais (SQL) organizam os dados em tabelas com um esquema rígido, permitindo relações claras entre diferentes conjuntos de informações, ideais para garantir consistência e integridade dos dados. Já os bancos de dados não relacionais (NoSQL) oferecem maior flexibilidade ao lidar com dados não estruturados, sendo mais indicados para aplicações que demandam escalabilidade e manuseio de grandes volumes de dados de maneira mais ágil e adaptável. A escolha entre os dois depende das necessidades específicas de estruturação e escalabilidade da aplicação.

API: Conjunto de regras e protocolos que permitem que diferentes sistemas e aplicativos se comuniquem entre si. Funciona como uma ponte, possibilitando que um *Software* envie ou receba dados de outro de forma estruturada e segura, sem precisar entender os detalhes internos de seu funcionamento. As APIs são amplamente utilizadas em diversas aplicações, como serviços *web*, onde o *frontend* interage com o *backend* para buscar ou enviar dados, ou na integração de sistemas de terceiros, como plataformas de pagamento e serviços de mapa.

Comunicação entre Serviços: Em aplicações modernas, diferentes partes podem ser distribuídas em serviços independentes. A comunicação entre esses serviços ocorre geralmente via *APIs* (*Application Programming Interfaces*), utilizando protocolos como *REST* ou *GraphQL*. acima

Hospedagem: Refere-se à disponibilização da aplicação na *web*. Isso envolve a escolha de um serviço de hospedagem, como AWS, Heroku ou Azure, e o processo de *deploy*, que coloca a aplicação em produção para ser acessada pelos usuários.

Esses conceitos representam a arquitetura básica de uma aplicação *web*, com o *frontend* cuidando da interação com o usuário, o *backend* gerenciando a lógica e os dados, o banco de dados armazenando informações, a comunicação entre serviços permitindo a integração e a hospedagem disponibilizando a aplicação online. A compreensão e a aplicação adequada desses conceitos são fundamentais para o desenvolvimento eficiente e bem-sucedido de aplicações *web*.

2.3 Desenvolvimento *Mobile*

O desenvolvimento *mobile* refere-se à criação de aplicativos específicos para dispositivos móveis, como *smartphones* e *tablets*, que podem ser instalados diretamente nos sistemas operacionais nativos, como Android ou iOS. Ao contrário das aplicações *web*, que rodam em navegadores, os aplicativos mobile são desenvolvidos para plataformas es-

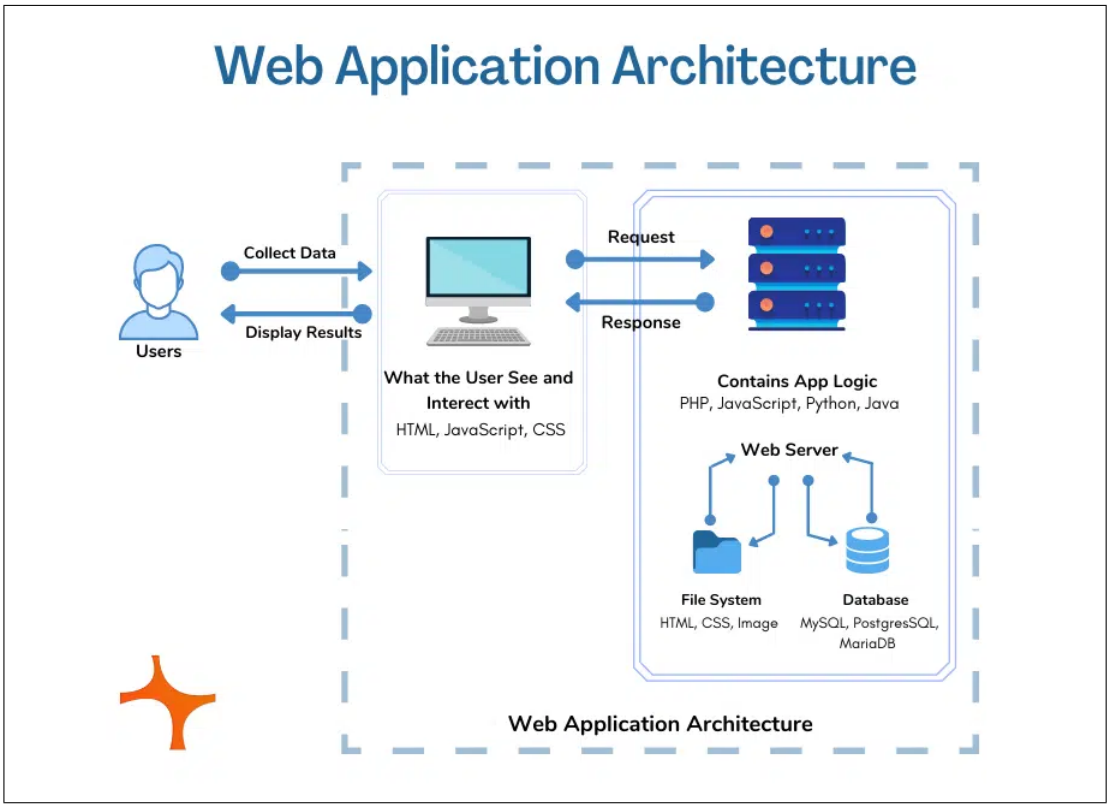

Figura 2 – Arquitetura aplicação *Web* (CYNOTECK, 2021)

pecíficas. Segundo (MOLINARI; FIGUEIRA, 2020), a escolha entre o desenvolvimento *mobile* nativo e abordagens híbridas ou *cross-platform* depende diretamente dos recursos que se pretende explorar no dispositivo e da eficiência que se deseja alcançar.

2.3.1 Lojas de aplicativos

As lojas de aplicativos são os repositórios oficiais de *Software* para cada plataforma móvel, sendo as principais atualmente a *App Store* (para dispositivos iOS) e a Google *Play Store* (para dispositivos Android). Essas lojas são as maiores e mais influentes, pois estão diretamente vinculadas aos sistemas operacionais mais utilizados no mercado de *smartphones*. De acordo com um estudo recente, o *Android* domina o mercado global com aproximadamente 72% de participação, enquanto o *iOS* corresponde a cerca de 27% (STATISTA, 2023). Esse alcance faz com que a presença e o bom desempenho desses aplicativos nas lojas sejam essenciais para seu sucesso.

Como mencionado por (SMITH, 2020), a qualidade de um aplicativo comumente é avaliada pelos usuários por meio de métricas, como a quantidade de usuários ativos, as notas atribuídas ao aplicativo e o número de *downloads*, o que reflete diretamente a experiência do usuário e a aceitação no mercado. Aplicativos com um maior número de avaliações positivas e uma alta taxa de *downloads* tendem a ser recomendados com mais frequência para novos usuários, gerando um ciclo positivo de crescimento e retenção. Além

disso, o sistema de *feedback* das lojas é fundamental para os desenvolvedores, pois permite melhorias contínuas com base nas sugestões e críticas dos usuários.

2.3.2 Vantagens

- Integração com funcionalidades nativas: Aplicativos *mobile* têm acesso direto aos recursos do dispositivo, como câmera, GPS, acelerômetro e notificações *push*. Isso permite uma experiência mais rica e personalizada para o usuário.
- Melhor desempenho: Aplicativos desenvolvidos de forma nativa tendem a ter melhor desempenho e resposta, pois são otimizados para o sistema operacional e *hardware* específicos.
- Acesso *offline*: Diferentemente das aplicações *web*, que necessitam de uma conexão constante com a *internet*, muitos aplicativos *mobile* permitem o uso *offline*, oferecendo funcionalidades mesmo sem conexão.

2.3.3 Desvantagens

- Desenvolvimento multi-plataforma: Desenvolver aplicativos nativos para diferentes plataformas (*Android* e *iOS*) requer o desenvolvimento e manutenção de códigos distintos, o que pode aumentar os custos e o tempo de desenvolvimento.
- Distribuição e atualização: Ao contrário das aplicações *web*, que podem ser atualizadas diretamente no servidor, aplicativos mobile dependem de aprovação das lojas de aplicativos (Google *Play*, *App Store*) e exigem que os usuários façam o download de atualizações manualmente.
- Compatibilidade de dispositivos: A grande variedade de dispositivos móveis no mercado pode trazer desafios relacionados à compatibilidade de *hardware* e *Software*, exigindo testes mais amplos para garantir a usabilidade em diferentes modelos e versões de sistemas operacionais.

2.3.3.1 PWA

O artigo (INUKOLLU et al., 2014) aborda o desafio da reutilização de código no desenvolvimento de aplicativos móveis, destacando a falta de interoperabilidade nos códigos nativos, o que historicamente resultou em projetos e ambientes de desenvolvimento separados para diferentes plataformas móveis. Isso significa que, tradicionalmente, os desenvolvedores precisavam criar códigos distintos para aplicativos nativos, *web* e outras plataformas móveis, o que aumentava a complexidade do processo de desenvolvimento.

Diante dessa falta de reusabilidade de código, o desenvolvimento cross-platform surge como uma solução popular para empresas que não possuem os recursos necessários para contratar desenvolvedores especializados para cada plataforma específica. Essa abordagem permite a criação de um único código-fonte que pode ser utilizado em diferentes sistemas operacionais e plataformas móveis, reduzindo assim o esforço de desenvolvimento e acelerando o tempo de lançamento dos aplicativos. A proposta é atrativa, especialmente em termos de economia de orçamento, eficiência de recursos humanos e utilização do conhecimento existente.

Dessa forma, o conceito de *Progressive Web Apps* (PWAs) vem como uma alternativa para superar os desafios mencionados. As PWAs buscam unificar a experiência móvel ao trazer recursos semelhantes aos aplicativos nativos para a plataforma web. Elas são caracterizadas por serem progressivas, responsivas, independentes de conectividade, semelhantes a aplicativos, atualizadas, seguras, descobríveis, reengajáveis e instaláveis. Essa abordagem visa proporcionar uma solução web-native que combina benefícios tanto de aplicativos nativos quanto de aplicativos cross-platform, permitindo que as empresas atendam às expectativa de ambas às neecessidades.

2.4 Desenvolvimento Ágil

A Engenharia de *Software*, por ser uma área de estudos e atuação relativamente recente em comparação com outras engenharias já consolidadas, como a Engenharia de Produção ou Civil, herda as abordagens de processos e ferramentas utilizadas por suas predecessoras, as quais são conhecidas por serem tradicionais e sequenciais.

Embora tenha se mostrado eficaz em outras engenharias e, em alguns contextos, dentro da Engenharia de *Software*, os modelos sequenciais revelaram-se ineficientes devido à dinamicidade desta nova área. Diante desse cenário, o Manifesto Ágil foi concebido por 17 indivíduos durante o período de 11 a 13 de fevereiro de 2001.

De acordo com o Manifesto (BECK, Kent, et al., 2001), destaca-se a importância de indivíduos e interações, *Software* em funcionamento, colaboração com o cliente e a capacidade de responder a mudanças.

2.5 *Scrum*

O Scrum representa uma abordagem ágil que se destaca pela ênfase na comunicação, colaboração e iteração. Esta metodologia eficaz no desenvolvimento de *Software* proporciona à equipe a flexibilidade necessária para se adaptar às mudanças nas necessidades do cliente, resultando na entrega de um produto de alta qualidade em curtos intervalos de tempo (SUTHERLAND; SCHWABER, 2014). Estruturado em ciclos de

trabalho denominados sprints, com duração fixa geralmente variando de duas a quatro semanas, o Scrum conduz a uma série de atividades ao longo de cada sprint, assegurando assim uma entrega contínua de valor ao cliente.

2.6 *Extreme Programming*

O *Extreme Programming* (XP), concebido por Kent Beck, é uma metodologia ágil que destaca a qualidade do Software e a adaptabilidade a mudanças. Com práticas como Desenvolvimento Orientado a Testes, programação em pares e integração contínua, o XP busca garantir a qualidade do código e promover entregas frequentes. Além disso, enfatiza a simplicidade do *Software*, buscando torná-lo fácil de entender. A metodologia prioriza o *feedback* contínuo dos usuários desde as fases iniciais do desenvolvimento, permitindo ajustes ágeis às demandas do ambiente (BECK, 2000). Sua abordagem flexível e adaptativa não apenas facilita a resposta a mudanças, mas também aumenta a eficiência na entrega de valor aos usuários finais

2.7 Prototipagem

A prototipagem é uma etapa essencial no ciclo de desenvolvimento de *Software*. Esse processo consiste na criação de representações iniciais do produto, chamadas de protótipos, que tem o objetivo de testar, validar e ajustar conceitos antes da implementação final.

Prototipar ajuda a identificar problemas, alinhar expectativas entre desenvolvedores e clientes, além de permitir ajustes rápidos sem o custo elevado de refazer o produto em estágios avançados. De acordo com (NIELSEN, Jakob, 2003), os protótipos em papel são muitas vezes subutilizados. O uso desses protótipos proporcionaria um grande ganho de agilidade no *feedback* entre a equipe de desenvolvimento e o cliente. É possível validar várias características da fase de desenvolvimento por meio de desenhos simples e rudimentares. Em contrapartida, a busca por criar uma interface visualmente atraente antes de apresentá-la ao cliente pode gerar atrasos significativos no desenvolvimento, dificultando a obtenção rápida de *feedback* essencial para a tomada de decisões.

A prototipagem pode ser feita com diferentes níveis de fidelidade, variando conforme a fase do projeto e os recursos disponíveis. Esses níveis determinam o grau de detalhamento e proximidade do protótipo em relação ao produto final. Os principais tipos de prototipagem são:

1. Protótipo de baixa fidelidade: Envolve o uso de esboços simples, como protótipos em papel, e é útil para explorar ideias iniciais de forma rápida e de baixo custo.

Esses protótipos permitem ajustes ágeis, ajudando na validação de conceitos antes que muitos recursos sejam investidos.

2. Protótipo de média fidelidade: Oferece um grau maior de detalhamento e é uma representação mais precisa do produto em desenvolvimento. Embora ainda não seja a versão final, esse tipo de protótipo é eficaz para testar a viabilidade técnica e promover refinamentos mais detalhados.

3. Protótipo de alta fidelidade: Se aproxima significativamente do produto final, contendo um elevado nível de detalhes e, frequentemente, elementos interativos. Embora mais custosos e demorados de produzir, esses protótipos são ideais para simular a experiência do usuário de maneira realista, sendo usados para testar a usabilidade e validar o design final antes do desenvolvimento completo.

3 Suporte Tecnológico

Este capítulo abordará as ferramentas utilizadas nesta aplicação, descrevendo suas características e os motivos para sua escolha.

3.1 Ferramentas

3.1.1 Netlify

O Netlify é uma plataforma de hospedagem e *deploy* contínuo voltada principalmente para sites estáticos e aplicativos web modernos. Ele permite que desenvolvedores integrem repositórios de código de plataformas como GitHub, GitLab e Bitbucket, facilitando o processo de *build* e publicação automática sempre que há uma atualização no código. pagamento para aplicativos de pequena escala. Além disso, oferece facilidade de uso e um plano gratuito adequado às necessidades iniciais do projeto, considerando o baixo volume de acessos previsto. O Netlify proporciona uma solução integrada com o GitHub para *deploy* contínuo, eliminando a necessidade de configurações manuais, o que simplifica e torna mais eficiente o processo de publicação e hospedagem. Especificamente, no caso desta aplicação, a cada *pull request* aberto para a *branch* configurada como padrão de *deploy*, ou a cada *commit* realizado nessa mesma *branch*, um novo processo de *build* e execução é disparado automaticamente para atualizar a versão da aplicação em produção.

3.1.2 Figma

O Figma foi utilizado no projeto para a prototipagem das telas e suas interações, permitindo a análise visual das modificações necessárias e do comportamento dos componentes em diferentes resoluções, assegurando a responsividade da interface. A ferramenta também foi crucial na prototipação de novas funcionalidades, proporcionando uma visualização realista de como essas mudanças se comportariam na prática. Além disso, o Figma foi empregado para validar se a interface seguia fielmente a identidade visual definida, garantindo consistência e qualidade no *design*. O uso colaborativo da plataforma permitiu ajustes em tempo real, melhorando a comunicação entre os envolvidos e otimizando o processo de *design* e desenvolvimento.

3.2 Linguagem de Programação

3.2.1 TypeScript:

É uma linguagem amplamente utilizada no desenvolvimento *web* moderno, conhecida por adicionar tipagem estática e recursos avançados ao JavaScript, sem perder a compatibilidade com este. Sua versatilidade, aliada à segurança proporcionada pela tipagem, torna-o ideal para a criação de aplicações escaláveis e robustas.

No projeto, TypeScript foi adotado como a principal tecnologia para implementar a lógica das requisições recebidas e construir o serviço responsável por armazenar as respostas. Além disso, TypeScript foi utilizado para gerenciar a interação das telas da aplicação, garantindo uma experiência dinâmica e responsiva, com a vantagem adicional de maior controle sobre os tipos de dados e verificação de erros em tempo de compilação.

3.3 Bibliotecas e *Frameworks*

3.3.1 React

O React é uma biblioteca de JavaScript desenvolvida e mantida pelo Facebook, projetada para facilitar a criação eficiente de interfaces de usuário, com foco em aplicações web interativas e dinâmicas. Segundo a documentação oficial do (PLATFORMS, 2023), a biblioteca adota uma abordagem baseada em componentes, onde as interfaces são divididas em partes independentes e reutilizáveis, facilitando a manutenção e escalabilidade do código . Essa modularidade permite que os desenvolvedores criem e reutilizem componentes, otimizando o tempo de desenvolvimento em projetos com estruturas repetitivas ou semelhantes.

No caso desta aplicação, em que muitos componentes são semelhantes e reutilizados várias vezes, a abordagem do React provou ser extremamente eficiente, reduzindo o tempo necessário para construir e ajustar diferentes partes da interface. A reutilização de componentes é um dos pilares fundamentais do React, o que demonstra a relevância dessa biblioteca para essa aplicação, visto a necessidade de uma ferramenta eficiente e acessivel.

3.3.2 Next.JS

O Next.js é um *framework* de React projetado para facilitar o desenvolvimento de aplicações web, proporcionando páginas rápidas e eficientes. Ele oferece uma série de funcionalidades e convenções que tornam o processo de criação de aplicações modernas mais produtivo. Entre suas principais características estão a renderização híbrida, roteamento automático, pre-renderização, API Routes, suporte a CSS *Modules* e *Styled-jsx*,*Hot Module Replacement* e suporte nativo a TypeScript, como destacado por (Next.js, n.d.).

Next.js foi escolhido para este projeto devido a suas características, especialmente o sistema nativo de requisições e rotas. Com API Routes e o roteamento automático, ele elimina a necessidade de bibliotecas externas para configurar rotas e requisições, reduzindo significativamente o esforço e a complexidade envolvidos nesse processo. Essa integração nativa facilita a criação dos métodos de requisição para buscar os dados formatados, que são utilizados para renderizar o conteúdo das telas, além de simplificar a navegação da aplicação.

3.4 Ferramentas de comunicação

3.4.1 WhatsApp

O WhatsApp foi utilizado para discussões rápidas sobre os requisitos durante a fase de desenvolvimento, bem como para tratar de necessidades de mudanças em funcionalidades e outras questões semelhantes. Sua principal vantagem era a facilidade de comunicação, devido ao seu caráter informal, o que simplificava muito o processo de interação entre os membros da equipe. Além disso, foi amplamente utilizado para marcar reuniões e agendar horários para sessões de pair programming.

3.4.2 Discord

O Discord é uma ferramenta de bate-papo e comunicação por voz baseada em grupos, que permite que várias pessoas participem de uma chamada ao mesmo tempo, oferecendo recursos como compartilhamento de tela, áudio e voz. Foi bastante utilizado em reuniões de codificação e discussões sobre o desenvolvimento do projeto, pela facilidade de uso e pelo fato de não exigir agendamentos prévios. Junto com o WhatsApp, foi um dos principais meios de comunicação da equipe.

3.4.3 Google Meet

O Google Meet também é uma ferramenta de videoconferência bastante popular e confiável. Foi utilizada em reuniões periódicas com os idealizadores do projeto, incluindo alunos e o professor da Universidade de São Paulo (USP), além de encontros regulares com o orientador deste projeto.

4 Metodologia

A meta deste proposto trabalho foi de desenvolver uma aplicação mobile, utilizando a característica de PWA (*Progressive Web App*), que transforma uma aplicação web em uma experiência similar à de um aplicativo nativo. Assim, a aplicação é capaz de consultar um endereço da web que armazena fluxos de algum tema juridico. Esses fluxos são formados atraves do usuo de formularios feitos no Google Forms e apos isso eles são tratados numa aplicação externa e disponibilizados em formato JSON para eta aplicação em questao consumirem.

O meio de exbição se da por meio de telas com navegação intertativa, baseado no conteudo presente nos JSONs obtidos. Permitindo avançar e retornar no conteudo exibido atualmente. Visto o escopo desse projeto, outro motivo do uso de PWA frente a criaçaõ de um aplicativo nativo foi sua simplificade em publicação ja que é publicado como qualquer *site* ou aplicação na internet, que transformado ele em um PWA possui caracteristicas de um *app* nativo e não sendo necessario requisitos de licenciamento de dados pessoais, uma vez que não armazenará informações dos usuários, facilitando a conformidade com a LGPD (Lei Geral de Proteção de Dados). Com esses recursos, a aplicação será uma ferramenta valiosa para disseminar conteúdo jurídico de maneira educativa e informativa, apresentando direitos e deveres de forma prática, acessível e simples de entender.

A aplicação exibe telas de navegação interativa, baseadas nos dados contidos no JSON obtido, proporcionando uma boa experiência de usabilidade, com foco em facilidade de leitura e compreensão dos fluxos de ações e recursos disponíveis. Além disso, o uso do PWA permitirá que o aplicativo seja facilmente distribuído sem necessidade de cumprir requisitos de licenciamento de dados pessoais, uma vez que não armazenará informações dos usuários, facilitando a conformidade com a LGPD (Lei Geral de Proteção de Dados). Com esses recursos, a aplicação será uma ferramenta valiosa para disseminar conteúdo jurídico de maneira educativa e informativa, apresentando direitos e deveres de forma prática, acessível e simples de entender.

4.1 Gerência de Projeto

A administração de projetos é uma prática vital para o planejamento, organização, coordenação e controle eficientes de projetos. Neste trabalho, foi explorado as vantagens de metodologias ágeis, como *Scrum*, *XP* e *Kanban*, buscando otimizar a gestão do projeto e promover uma abordagem mais flexível e adaptativa.

4.1.1 *Scrum*

Sobre o *Scrum*, foram aplicados alguns conceitos desse *framework*, como as *sprints*, que têm uma duração de 15 dias cada. O *Sprint Planning* ocorreu no início de cada *sprint*, abordando tarefas, metas e a distribuição dessas entre a equipe de desenvolvimento. Para auxílio na gestão e controle das tarefas em desenvolvimento, foi utilizada a ferramenta ZenHub. Por meio dessa ferramenta, é possível criar *cards* de desenvolvimento e vinculá-los a *issues* específicas, estabelecendo assim uma ligação clara entre cada tarefa e a *issue* correspondente. Além desse, acontecerá tambem o *spring review*, que será uma revisão das tarefas idealizadas para serem desenvolvidas na *sprint* em questão e oque foi de fato almejado. Com isso, será possivel ter um controle dos principais impedimentos da equipe no ciclo de desenvolvimento e criar possiveis soluções para evitar esses empedimentos novamente. Para um *feedback* mais contínuo da aplicação, também será realizada uma apresentação ao final de cada sprint, visando mostrar o estado atual da aplicação e confirmar se atende às expectativas dos *stakeholders*. Usando os conceitos e métodos citados acima, a aplicação ganhará um desenvolvimento mais ágil e com menos documentação desnecessária, proporcionando maior eficiência e rapidez no processo de desenvolvimento

4.1.2 *Extreme Programming*

No desenvolvimento desta aplicação, serão aplicados conceitos do *XP*, como o desenvolvimento em pares e o *feedback* contínuo. O desenvolvimento em pares contribuirá para um melhor entrosamento entre a equipe, possibilitando a identificação e correção de erros no momento em que surgem. Também será utilizada a característica de *feedback* rápido, priorizando-a durante todo o processo de desenvolvimento. Isso inclui testes contínuos, integração contínua e revisões regulares de código. O *feedback* rápido permite correções imediatas e aprimoramento contínuo do produto.

- *Pair programming*: O pair programming será usado como recurso de desenvolvimento, pois permite sempre ter o ponto de vista de duas pessoas sobre o mesmo problema, facilitando a busca por soluções e o enriquecimento de ideias. Além disso, possibilita uma revisão de código contínua pela dupla, ao mesmo tempo que o código é escrito. Essa prática não se restringe apenas ao código, mas também pode ser aplicada em outras atividades, como a definição de arquitetura e o escopo do projeto.

- *Backlog*: Será utilizada uma lista de tarefas, que, ao ser concluída, será capaz de entregar uma versão do aplicativo já com as funcionalidades essenciais. Essa lista será ordenada conforme a importância e a necessidade de cada item para a entrega eficiente da aplicação.

- *Design* simples: Refere-se à criação de soluções que atendem às necessidades atuais de forma clara, sem adicionar complexidade desnecessária. No contexto da renderização de fluxos, mesmo sendo uma aplicação simples, ela será capaz de entregar muito valor por meio de uma solução objetiva e eficiente.

4.2 Gerência e Configuração de *Software*

Para garantir a qualidade do software entregue e permitir um rastreamento eficiente das evoluções do projeto, foram adotados conceitos e práticas de gerenciamento de configuração de software. Essas práticas asseguram uma documentação clara e objetiva ao longo de todo o ciclo de desenvolvimento, proporcionando maior solidez em todas as fases do processo. Além disso, essa abordagem facilita a separação de funcionalidades e artefatos, garantindo uma organização adequada das diferentes partes da aplicação.

4.2.1 Git e Github

Git e o GitHub são essenciais no desenvolvimento de *Software*, proporcionando uma base sólida para documentação e adoção de boas práticas do *Scrum* e do *Extreme Programming* (XP). O Git, um sistema de controle de versão distribuído, permite colaboração eficiente, rastreamento de alterações e reversão para versões anteriores. Já o GitHub, plataforma que hospeda repositórios Git, oferece recursos para gestão de projetos, revisões de código, reporte de problemas e integração contínua, facilitando práticas do *XP*, como programação em pares e revisões colaborativas. Como é mostrado em (GITHUB, 2023), essas ferramentas se tornaram fundamentais para o desenvolvimento ágil e a gestão colaborativa de projetos, assegurando controle de versão e melhor organização das equipes. No caso deste projeto, é possível manter todo o histórico de versões de desenvolvimento e, juntamente com a plataforma Netlify, visualizar o log de *build* para cada nova versão em processo de implantação. Isso permite analisar criteriosamente os motivos de eventuais falhas, facilitando a identificação e correção de problemas.

4.2.1.1 *Trunk-Based Development*

Trunk-Based Development (TBD) é uma prática de desenvolvimento de *Software* na qual os desenvolvedores trabalham em um único *branch* principal, geralmente chamado de "*trunk*"ou "*main*", integrando pequenas mudanças frequentemente. Para este projeto, adotamos o TBD devido ao tamanho reduzido da equipe, composta por apenas duas pessoas. O uso de muitas políticas e normas poderia atrasar o desenvolvimento, tornando o processo mais burocrático. Com o TBD, trabalhamos com um *branch* de desenvolvimento para implementar novas funcionalidades e, quando consolidávamos uma versão estável com várias funcionalidades, integrávamos ao *branch* main.

O *deploy* sempre foi feito diretamente a partir do *branch main*, um processo automatizado via Netlify, onde as configurações estavam definidas nos *job details* da plataforma, garantindo uma entrega contínua sem necessidade de muitas intervenções manuais. Como mostrado em (DEVELOPMENT, 2023), o TBD é especialmente útil em equipes pequenas, pois permite uma integração contínua e rápida, sem a sobrecarga de manter múltiplos *branches* de longa duração.

4.3 *Backlog*

O *backlog*, em metodologias ágeis, é uma lista organizada de tarefas a serem realizadas dentro de um projeto, facilitando o planejamento e a execução. Ele é fundamental para fragmentar problemas maiores em partes menores e mais gerenciáveis, o que permite que a equipe foque em resolver essas pequenas partes, que, ao serem concluídas, ajudam a resolver o problema maior como um todo (ATLASSIAN, 2024).

4.3.1 Requisitos Funcionais

Requisitos funcionais são as funcionalidades ou serviços que o sistema deve oferecer, descrevendo o comportamento esperado e as interações com o usuário ou com outros sistemas. Eles definem o que o sistema deve fazer.

- O sistema deve ser capaz de consultar quais fluxogramas estão disponíveis na aplicação externa para serem visualizados e renderizados na interface.
- O sistema deve ser capaz de lidar com a atualização do conteúdo de um fluxograma que já foi renderizado anteriormente.
- O sistema deve ser capaz de retomar o ponto onde o usuário parou, caso ele saia do aplicativo enquanto navega por um fluxograma.
- O sistema deve ser capaz de carregar dados armazenados localmente, caso não haja conexão com a internet no momento do acesso, utilizando a última sincronização feita enquanto *online*.
- O sistema deve permitir ao usuário baixar o aplicativo ou acessá-lo diretamente via navegador.

4.3.2 Requisitos Não Funcionais

Requisitos não funcionais são as características de qualidade do sistema, como desempenho, segurança, escalabilidade e usabilidade. Eles definem como o sistema deve se comportar e as restrições sobre sua operação.

- O sistema deve ser desenvolvido como uma *Progressive Web App* (PWA), garantindo funcionalidades como acesso *offline* e experiência semelhante a um aplicativo nativo.
- O sistema deve permitir a renderização de diferentes fluxos de interação de forma eficiente, adaptando-se conforme o contexto do usuário.
- O sistema deve manter uma identidade visual coesa e harmônica entre as diferentes telas e componentes, promovendo consistência na interface.
- O sistema deve apresentar *design* responsivo, adaptando-se adequadamente a diferentes dispositivos e tamanhos de tela, garantindo uma boa experiência de uso em todas as plataformas.

5 Resultados

O produto final desenvolvido foi um *Progressive Web App* (PWA). O aplicativo é totalmente funcional tanto em *desktops* quanto em dispositivos móveis, embora seja recomendado seu uso nestes últimos, uma vez que a interface e os recursos foram projetados especialmente para esses dispositivos. O aplicativo realiza chamadas a dois endpoints principais no *backend*: um para retornar a lista de fluxos disponíveis e outro para obter os dados de um fluxo específico. Com essas informações, o aplicativo constrói as telas interativas para o usuário, permitindo que ele percorra o fluxo de perguntas e visualize as informações correspondentes.

5.1 Construção das Telas com Base na Estrutura JSON

Como mencionado anteriormente, a construção das telas do aplicativo é totalmente baseada na estrutura JSON recebida do *backend*. Nas subseções seguintes, detalharemos um pouco mais e apresentaremos exemplos de como essas estruturas são transformadas em telas.

5.1.1 Tela Inicial

A tela inicial é composta por um texto, uma barra de pesquisa e as opções de fluxo que o usuário pode escolher seguir. As figuras 3 e 4 representam, respectivamente, a estrutura JSON e a tela que é gerada no aplicativo a partir dessa estrutura. No JSON, há uma lista de objetos contendo duas chaves: id e name. O valor de `id` é utilizado para identificar um fluxo específico, sendo posteriormente usado para realizar uma requisição para obter os detalhes desse fluxo. Já o `name` serve para nomear o fluxo e será exibido na tela para apresentar as opções disponíveis ao usuário, conforme ilustrado na Figura 4.

5.1.2 Fluxo

Quando o usuário seleciona o fluxo que deseja seguir na tela inicial, ele é direcionado para a primeira seção do fluxo. Essa seção pode conter um ou mais elementos, como uma pergunta simples com opções de resposta, uma pergunta acompanhada de uma imagem ou um elemento com texto informativo. Abaixo, são apresentadas, respectivamente, a imagem da estrutura padrão de uma seção (especificamente a estrutura de uma pergunta)(Figura 5), a tela construída com base nessa estrutura (Figura 6), a imagem da estrutura de uma informação (Figura 7) e, por fim, a tela gerada a partir dessa informação (Figura 8).

```
[
    {
        "id": "1vhJXzxgy12Valle8h3z9DJuXFOlzSa0odczLe9kY5aw",
        "name": "Legislação Trabalhista"
    },
    {
        "id": "1ORe5ff4Oo_iu9juZmD4-OCDXdV7s3V8LyYhqc_MK3qY",
        "name": "fluxo2"
    }
]
```

Figura 3 – Estrutura JSON da tela inicial. Fonte: Produção própria

Figura 4 – tela inicial. Fonte: Produção própria

```
{
    "title": "Direitos trabalhistas",
    "sections": [
        {
            "sectionId": "FIRST_SECTION",
            "title": "Direitos trabalhistas",
            "description": null,
            "items": [
                {
                    "type": "RADIO",
                    "required": true,
                    "options": [
                        {
                            "value": "Sim",
                            "goToSectionId": "5b18e789"
                        },
                        {
                            "value": "Não",
                            "goToSectionId": "58a689a1"
                        }
                    ],
                    "itemId": "1e2061a2",
                    "description": "A carteira de trabalho é um instrumento... ",
                    "text": "Você tem carteira assinada?"
                }
            ]
        },
}
```

Figura 5 – Estrutura JSON de uma pergunta. Fonte: Produção própria

Figura 6 – Tela de uma pergunta. Fonte: Produção própria


```
{
  "sectionId": "55c5d3d8",
  "title": "Meus Direitos",
  "description": null,
  "items": [
      {
          "itemId": "2796ef2e",
          "type": "textItem",
          "text": "Férias",
          "description": "É direito do empregado férias anuais de 30 dias e remuneradas com, pelo menos,
⅓ a mais que o salário normal. O direito se dá após cada período de 12 meses de serviço prestado à mesma
pessoa ou família, contado da data da admissão. "
      }
  ]
}
```


Figura 7 – Estrutura JSON de uma informação. Fonte: Produção própria

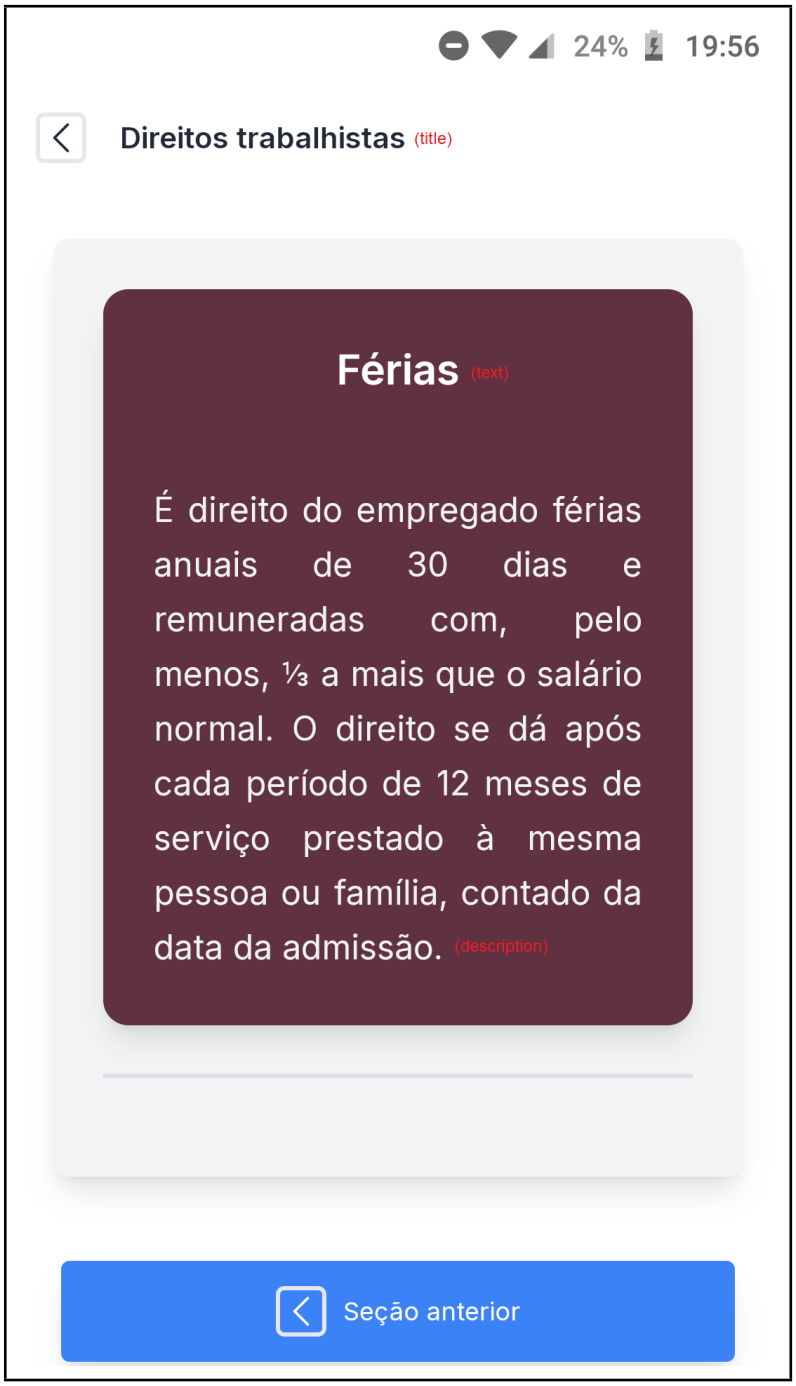

Figura 8 – Tela de uma informação. Fonte: Produção própria

Como pode ser observado, a segunda estrutura (Figura 5) é mais complexa, por isso será dividida em partes menores. O objeto principal, que neste caso representa um fluxo jurídico, é composto por duas chaves: `title` e `sections`. A primeira chave, `title`, é utilizada para nomear o fluxo, enquanto a segunda, `sections`, é uma lista de seções. A seção é uma das principais estruturas do aplicativo, e as opções de cada pergunta direcionam o usuário para seções distintas, possibilitando a criação dos fluxos dinâmicos.

As seções são compostas por várias chaves, entre elas: `sectionID`, `title`,

`description` e `items`. Dentre essas, a chave mais importante é `items`, que contém todos os elementos que podem compor uma seção. A chave `items` é um vetor de objetos que inclui as seguintes chaves: `type`, `required`, `itemId`, `description` e `text`. Quando o elemento for uma pergunta, pode também incluir a chave `options`, e, quando houver imagens associadas, a chave `image`.

No primeiro exemplo de tela (Figura 6), está sendo renderizada uma pergunta. O `title` é exibido na parte superior da tela para indicar em qual fluxo o usuário se encontra. Em seguida, o `text` corresponde ao texto da pergunta propriamente dita. Logo abaixo, são exibidos os valores de cada objeto presente na chave `options`, que representam o texto em cada botão de escolha. Na segunda tela, está sendo exibida uma informação. Nesse caso, o `text` é utilizado como o título do elemento, enquanto o `description` exibe o texto descritivo.

5.2 Desafios Encontrados

Um dos principais desafios enfrentados foi entender como transformar um aplicativo web tradicional em um *Progressive Web App* (PWA), que se comporta como um aplicativo nativo mobile. Isso inclui permitir o download do app, acessá-lo através de um ícone na tela do dispositivo móvel, oferecer suporte offline, entre outras características típicas de aplicativos nativos. Para superar esse desafio, foi realizado um trabalho de pesquisa para compreender quais tecnologias habilitavam essas funcionalidades, além de testar bibliotecas que facilitassem essa configuração.

Outro desafio foi entender como realizar requisições HTTP em componentes renderizados no lado do cliente no framework Next.js, especificamente utilizando o App Router. A documentação do framework não era clara a esse respeito e carecia de exemplos práticos. O problema foi resolvido através de pesquisas em discussões em fóruns de tecnologia, como Stack Overflow, e na seção de issues do GitHub, até que uma solução funcional fosse encontrada.

5.3 Arquitetura

A arquitetura do aplicativo segue o modelo de uma *Progressive Web App* (PWA), garantindo suporte para múltiplas plataformas (desktop e mobile) com funcionalidades aprimoradas, como *cache*, suporte *offline*, e capacidade de *download*. A figura 9 ilustra como os principais componentes interagem.

- *Frontend* (Next.js): O *frontend* do aplicativo, construído com Next.js, está disponível tanto em *desktop* quanto em dispositivos móveis.

- *Service Workers*: Um dos componentes centrais da arquitetura PWA são os *Service Workers*, que gerenciam funcionalidades como: suporte *offline*, *cache* e *download*.
- Manifest.json: O arquivo manifest.json define como o PWA deve ser instalado e exibido no dispositivo do usuário, permitindo que ele funcione como um aplicativo nativo em várias plataformas.
- *Backend* (AWS Lambda): O *backend* é servido por funções AWS Lambda, que fornecem uma arquitetura *serverless*.

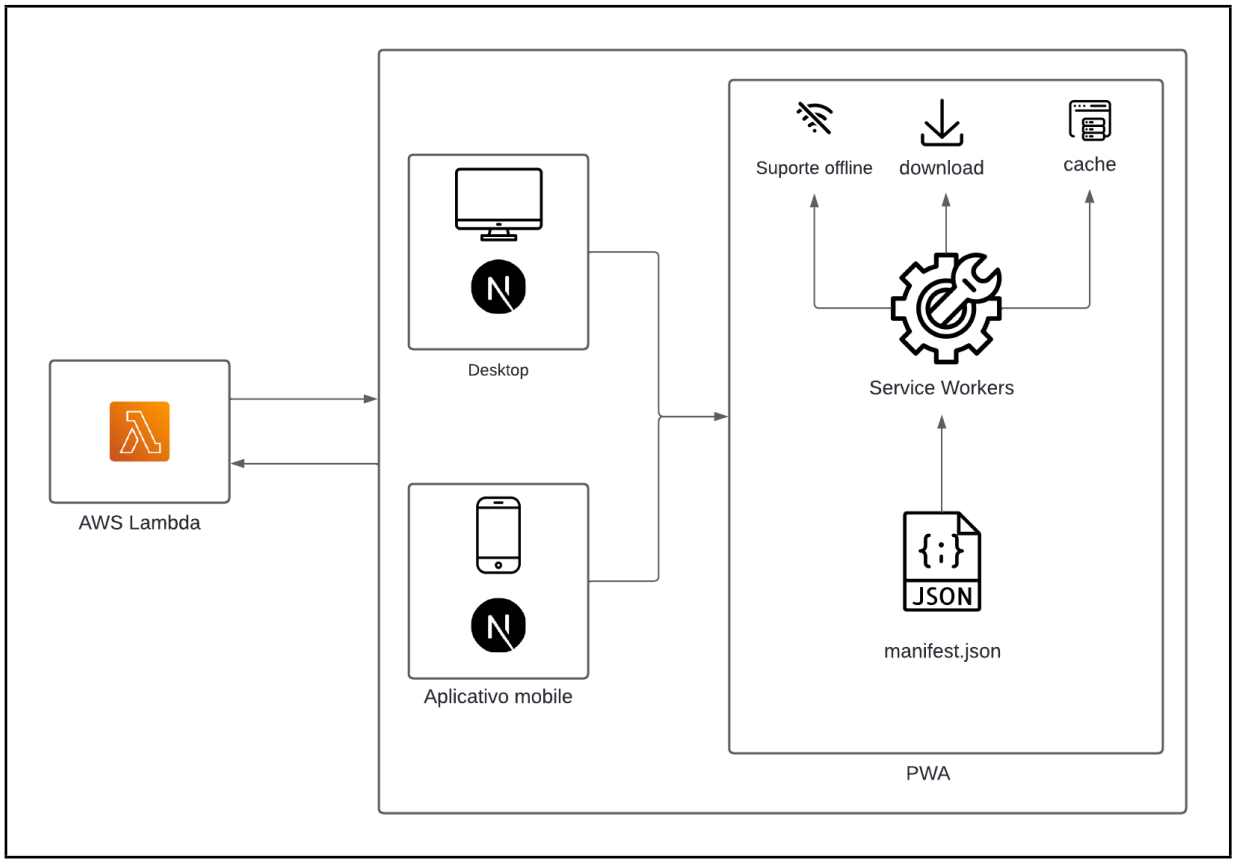

Figura 9 – Diagrama de Arquitetura. Fonte: Produção própria

5.4 Análise de Desempenho e Custo

Para a análise de desempenho, foi utilizada a ferramenta *Lighthouse*, uma solução automatizada de código aberto desenvolvida pelo Google para auditar a qualidade de páginas web. O *Lighthouse* avalia quatro dimensões principais: desempenho, acessibilidade, boas práticas e SEO. A análise foi conduzida utilizando a opção 'mobile' disponível na ferramenta.

- Desempenho: mede a velocidade de carregamento da página e outros indicadores, como o tempo de interação e o tempo até a renderização completa.
- Acessibilidade: verifica se a página é acessível para todos os usuários, incluindo aqueles com deficiências, considerando aspectos como contraste de cores, uso de tags semânticas e compatibilidade com leitores de tela.
- Práticas recomendadas: analisa se a página segue boas práticas de desenvolvimento, como segurança e uso correto de APIs.
- SEO (Otimização para Mecanismos de Busca): avalia se a página está otimizada para ser facilmente encontrada pelos motores de busca, considerando fatores como meta tags e estrutura de links.

Os resultados dessas análises podem ser observados na Figura 10, referente à tela inicial, e na Figura 11, que apresenta o desempenho de um fluxo específico.

Figura 10 – Análise de desempenho tela inicial. Fonte: Produção própria

Figura 11 – Análise de desempenho fluxo. Fonte: Produção própria

Em relação ao custo da aplicação, não houve despesas, pois utilizamos o plano gratuito do Netlify. Analisando o estágio de maturação deste produto e o baixo volume de dados renderizados por meio do acesso, não há necessidade de um plano pago de hospedagem no momento. Além disso, a hospedagem de aplicações *frontend*, como é o caso deste projeto, costuma ter um custo reduzido, além de demandar menos recursos computacionais.

5.5 Planejado x Realizado

Ao realizar a análise geral do que foi proposto no primeiro trabalho e comparar com o que foi entregue neste projeto atual, concluímos que todos os itens do *backlog* foram atendidos e o escopo do projeto foi plenamente cumprido. Graças à definição clara das funcionalidades do aplicativo e de seus recursos, foi possível seguir fielmente o planejamento e entregar o produto conforme esperado. O sistema opera como uma aplicação *web* que consome uma API REST, disponibilizando dados renderizados em telas e acessíveis em dispositivos móveis no formato de PWA.

5.6 Limitações

As limitações do aplicativo desenvolvido estão diretamente relacionadas às limitações inerentes aos PWAs. A principal delas é a compatibilidade com navegadores. Embora os PWAs funcionem bem na maioria dos navegadores modernos, algumas restrições ainda existem em certos navegadores, o que pode impactar a experiência dos usuários. Isso ocorre porque o PWA pode não oferecer a mesma funcionalidade em todos os navegadores, como o suporte a notificações push e a opção de *download* direto.

6 Considerações Finais

Este trabalho apresentou o desenvolvimento de uma aplicação web progressiva (PWA) voltada para a educação jurídica, com o objetivo de democratizar o acesso a informações sobre direitos trabalhistas. A proposta, que envolveu a construção de fluxos interativos para dispositivos móveis, mostrou-se eficiente na apresentação de conteúdos jurídicos de forma acessível e intuitiva, facilitando o entendimento de conceitos complexos para o público geral.

A implementação da renderização de fluxos a partir de estruturas JSON possibilitou uma navegação fluida e flexível, permitindo que os usuários interagissem com o conteúdo de maneira dinâmica. Além disso, o uso de tecnologias como o Next.js e o AWS Lambda contribuiu para a escalabilidade e eficiência do sistema, assegurando uma experiência robusta tanto em ambientes online quanto offline.

Embora os resultados obtidos tenham sido satisfatórios, algumas limitações foram identificadas, como a dependência de compatibilidade com determinados navegadores para que as funcionalidades do PWA sejam completamente acessadas. Essas questões podem ser abordadas em trabalhos futuros, assim como a exploração de funcionalidades adicionais, como integração com APIs externas e aprimoramento da interface de usuário.

Em resumo, este trabalho demonstrou que o uso de tecnologias modernas pode desempenhar um papel crucial na disseminação de informações jurídicas, contribuindo para a inclusão social e o fortalecimento do conhecimento sobre direitos e deveres, especialmente entre aqueles que não possuem formação jurídica.

Referências

ATLASSIAN. *Backlogs in Scrum*. 2024. Accessed: 13-09-2024. Disponível em: <https://www.atlassian.com/agile/scrum/backlogs>. Citado na página 28.

BECK, K. *Extreme Programming Explained: Embrace Change*. [S.l.]: Addison-Wesley Professional, 2000. Citado na página 20.

BECK, Kent, et al. *Manifesto para Desenvolvimento Ágil de Software*. 2001. Acesso em: 10 de Dezembro de 2023. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Citado na página 19.

CYNOTECK. *Web Application Architecture Diagram*. 2021. Disponível em: <https://cynoteck.com/wp-content/uploads/2021/12/Web-Application-Architecture-diagram-3.png>. Citado 2 vezes nas páginas 6 e 17.

DEVELOPMENT, T.-B. *Trunk-Based Development: A Git Workflow Strategy*. 2023. <https://trunkbaseddevelopment.com/>. Acesso em: 12 de setembro de 2024. Citado na página 28.

ELEVATEX. *Frontend vs. Backend*. 2022. Acesso em: 12 de setembro de 2024. Disponível em: <https://elevatex.de/wp-content/uploads/2022/02/Frontend-vs.-Backend-ENG.png>. Citado 2 vezes nas páginas 6 e 15.

FREIRE, P. *Pedagogia da Autonomia: Saberes Necessários à Prática Educativa*. 25. ed. São Paulo: Paz e Terra, 1996. (Coleção Leitura). Dados Internacionais de Catalogação na Publicação (CIP). ISBN 85-219-0243-3. Citado na página 11.

GITHUB, I. *Git and GitHub - Essential Tools for Modern Software Development*. 2023. <https://github.com/>. Acesso em: 12 de setembro de 2024. Citado na página 27.

IEEE Computer Society. Definição de engenharia de software. *IEEE Software*, v. 23, n. 6, p. 10–12, 2006. Retirado de "Guide to the Software Engineering Body of Knowledge". Citado na página 14.

INUKOLLU, V. N. et al. Factors influencing quality of mobile apps: Role of mobile app development life cycle. *CoRR*, abs/1410.4537, 2014. Disponível em: <http://arxiv.org/abs/1410.4537>. Citado na página 18.

KEKRE, S.; KRISHNAN, M. S.; SRINIVASAN, K. Drivers of customer satisfaction for software products: Implications for design and service support. *Management Science*, v. 44, n. 9, p. 1456–1490, 1995. Citado na página 14.

Mobile App Daily. *Benefits of Mobile Apps in Education | Insights & Trends*. 2024. Disponível em: <https://www.mobileappdaily.com/benefits-of-mobile-apps-in-education>. Citado na página 12.

MOLINA-RíOS, J.; PEDREIRA-SOUTO, N. Comparison of development methodologies in web applications. *Information and Software Technology*, v. 119, p. 106238, 2020. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584919302551>. Citado na página 14.

MOLINARI, J.; FIGUEIRA, M. *Desenvolvimento Mobile: Abordagens e Práticas*. São Paulo: Editora de Software, 2020. Citado na página 17.

Next.js. *The React Framework for the Web*. n.d. <https://nextjs.org/>. Acesso em: 10 de Dezembro de 2023. Citado na página 23.

NIELSEN, Jakob. *Paper Prototyping: Getting User Data Before You Code*. 2003. Acesso em: 10 de Dezembro de 2023. Disponível em: <https://www.nngroup.com/articles/paper-prototyping/>. Citado na página 20.

O'REGAN, G. *Fundamentals of Software Quality*. [S.l.]: Springer, Cham, 2019. ISBN 978-3-030-28494-7. Citado na página 14.

PLATFORMS, I. M. *React - A JavaScript Library for Building User Interfaces*. 2023. <https://reactjs.org/>. Acesso em: 12 de setembro de 2024. Citado na página 23.

Politize! *Entenda tudo sobre o acesso à justiça no Brasil!* 2024. Disponível em: <https://www.politize.com.br/entenda-tudo-sobre-o-acesso-a-justica-no-brasil/>. Citado na página 11.

RANDALL, J. Tackling legalese: How linguistics can simplify legal language. 2024. Disponível em: <https://web.northeastern.edu/lingandlaw/wp-content/uploads/2022/11/2014-Tackling-Legalese-2020_03_30-00_40_56-UTC.pdf>. Citado na página 12.

SMITH, J. Avaliação da qualidade de aplicativos móveis baseada nas métricas das lojas de aplicativos. *Revista de Desenvolvimento de Software*, Editora de Tecnologia, v. 15, n. 2, p. 45–60, 2020. Citado na página 17.

STATISTA. *Market share of mobile operating systems worldwide from 2012 to 2023*. 2023. <https://www.statista.com/statistics/272307/market-share-forecast-for-mobile-operating-systems/>. Acesso em: 12 de setembro de 2024. Citado na página 17.

SUTHERLAND, J.; SCHWABER, K. *Scrum: A Discipline for Agile Software Development*. [S.l.]: Addison-Wesley Professional, 2014. Citado na página 19.