Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Otimização de Desempenho Orientado ao Tempo de Carregamento em Aplicações *Web Front-end*

Autor: Mariana Nunes Pícolo

Orientador: Profª. Drª Milene Serrano

Brasília, DF

2021

Mariana Nunes Pícolo

Otimização de Desempenho Orientado ao Tempo de Carregamento em Aplicações *Web Front-end*

Monografia submetida ao curso de graduação em Engenharia de Software da 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[a]. Dr[a] Milene Serrano

Coorientador: Prof. Dr. Maurício Serrano

Brasília, DF

2021

Mariana Nunes Pícolo
Otimização de Desempenho Orientado ao Tempo de Carregamento em Aplicações *Web Front-end*/ Mariana Nunes Pícolo. – Brasília, DF, 2021-
89 p. : il. (algumas color.) ; 30 cm.

Orientador: Profª. Drª Milene Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2021.

1. *web front-end*. 2. desempenho. I. Profª. Drª Milene Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Otimização de Desempenho Orientado ao Tempo de Carregamento em Aplicações *Web Front-end*

CDU 02:141:005.6

Mariana Nunes Pícolo

Otimização de Desempenho Orientado ao Tempo de Carregamento em Aplicações *Web Front-end*

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

Trabalho aprovado. Brasília, DF, 20 de Maio de 2021:

Profª. Drª Milene Serrano
Orientador

Prof. Dr. Maurício Serrano
Coorientador

Profª. Drª. Carla Silva Rocha Aguiar
Convidado

Brasília, DF
2021

Agradecimentos

Aos meus pais, que sempre valorizaram o conhecimento e me apoiaram dentro e fora da graduação. Estão sempre comigo, em todos os momentos, sejam eles bons ou ruins, e me apoiam incondicionalmente em quaisquer caminhos que decido seguir.

Aos meus amigos, que me acompanharam e acreditaram em mim durante todos os momentos desta jornada. Vocês contribuíram em meu crescimento pessoal e profissional.

E por fim, mas não menos importante, aos meus orientadores, por sua dedicação e paciência. Vocês me guiaram e ajudaram a concretizar este projeto tão importante para mim.

A todos vocês, meu muito obrigada.

"Uma ilustração de figuras humanas deve ser feita de tal forma que o observador possa reconhecer facilmente, através de suas atitudes, as intenções de suas mentes."

Leonardo da Vinci

Resumo

A evolução das tecnologias de desenvolvimento web possibilitou conceber aplicações cada vez mais complexas, permitindo criar experiências de usuário personalizadas, com o objetivo de simplificar a vida dos usuários. Entretanto, essa riqueza em experiência de usuário pode resultar em aplicações cada vez mais dependentes de recursos como o *JavaScript*. Esses recursos podem impactar negativamente o tempo de carregamento dessas aplicações, surtindo um efeito indesejado de perda de usuários do produto, já que estes podem desistir do acesso às páginas, indo em busca de uma solução mais rápida que o atenda melhor. Este Trabalho de Conclusão de curso tem como objetivo ajudar desenvolvedores a otimizar projetos *web front-end*, através da reunião de material da literatura especializada. O guia, escrito em português, contém material a respeito de como diagnosticar problemas relativos ao desempenho de aplicações *web*, além de ferramentas, bem como técnicas utilizadas em *cases* de sucesso. A aplicação proposta neste trabalho encontra-se disponível como um projeto de código aberto.

Palavras-chaves: *front-end*. otimização. desempenho. tempo de carregamento. *open-source*. guia.

Abstract

The evolution of web development technologies has made it possible to design increasingly complex applications, allowing the creation of personalized user experiences, to simplify users' lives. However, this richness in user experience can result in applications that could be heavily dependent on JavaScript, which can negatively impact the loading time of these web pages, causing an unwanted effect of losing users and lowering conversion rates, since they can give up accessing the page, and simply look up for a faster solution that will serve them better. This Course Final Paper aims to help web developers to optimize their front-end projects, by gathering material from the specialized literature. The guide, written in Portuguese, contains material on how to diagnose problems related to the performance of web applications, as well as tools, and the techniques used in successful cases of web applications optimization. The application proposed in this Paper is available as an open-source project.

Key-words: front-end. performance. page loading times. optimization. open-source. guide.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CSS	*Cascading Style Sheets*
HTML	*HyperText Markup Language*
URI	Identificador Uniforme de Recursos
HTTP	*Hypertext Transfer Protocol*
FTP	*File Transfer Protocol*
DNS	*Domain Name System*
DOM	*Document Object Model*
API	*Application Programming Interface*
XML	*Extensible Markup Language*
FCP	*First Contentful Paint*
SVG	*Scalable Vector Graphics*
LCP	*Largest Contentful Paint*
FID	*First Input Delay*
TTI	*Time to Interactive*
TBT	*Total Blocking Time*
CLS	*Cumulative Layout Shift*
SSG	*Static Site Generation*
MVP	*Minimum Viable Project*
MB	*Megabyte*

Sumário

1 Introdução

Nesse capítulo, com base nos tópicos de interesse de pesquisa e desenvolvimento desse Trabalho de Conclusão de Curso, será apresentada uma breve contextualização, procurando abordar a questão de otimização em termos de desempenho. A contextualização revela ainda a problemática, com foco no tempo de carregamento em aplicações *web front-end*, permitindo conferir justificativas para a realização desse trabalho. A questão de pesquisa então é acordada, bem com os principais objetivos a serem atingidos ao longo dessa proposta. Por fim, é apresentada a organização dessa monografia.

1.1 Contextualização

A evolução das tecnologias de desenvolvimento *web* permitiu conferir *websites* cada vez mais complexos e com as mais diversas características utilizando *JavaScript*, HTML e CSS. O desempenho possui um papel importante no sucesso de uma aplicação *web*. Estudos mostram, por exemplo, que *websites* de alto desempenho retêm e envolvem mais usuários (PAVIC; ANSTEY; WAGNER, 2019).

De acordo com o Web Archive (HTTP ARCHIVE, 2020), o tamanho total de páginas *web* vêm crescendo desde 2011, quando esse número passou a ser contabilizado para *desktop* e *mobile*. O aumento de tamanho das páginas *web*, em especial com uso de *JavaScript*, impacta diretamente no desempenho, uma vez que o navegador precisa de mais tempo para baixar, compilar e executar o código recebido (RUSSEL, 2017).

Um estudo feito pela Ericsson (ERICSSON, 2016) aponta que a o estresse dos usuários advindo de atrasos no carregamento de páginas *web* é maior do que o estresse provocado ao aguardar na fila do caixa de uma loja. Usuários insatisfeitos tendem a não retornar mais ao *site*, o que pode custar dinheiro. O *Google* atribui pontos a todas as páginas *web* com base em seu desempenho. Os sites com mais pontos, e portanto, com melhores posições no *ranking* de busca, possuem maior desempenho (SHROFF; CHAUDHARY, 2017).

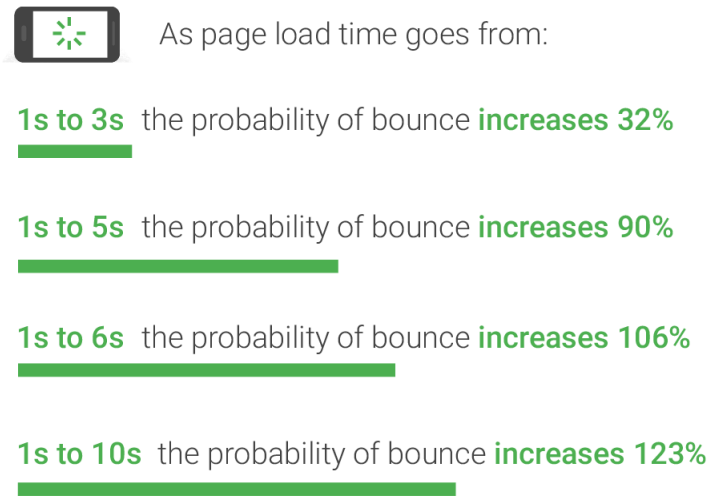
Definir os limites possíveis para as métricas de desempenho coletadas da aplicação pode ajudar os desenvolvedores a manter o controle da aplicação, uma vez que estabelece um ponto de referência, e pode orientar na tomada de decisões de *design* ou *features* (MIHAJLIJA, 2018). Os limites definem um quadro objetivo para determinar quais alterações no código caracterizam progresso e quais caracterizam regressões (RUSSEL, 2017).

1.2 Justificativa

De acordo com uma pesquisa realizada pelo Google(AN, 2018), com 11 milhões de *landing pages*, à medida que o tempo de carregamento da página vai de um segundo a 10 segundos, a probabilidade do usuário desistir e abandonar a página aumenta 123%, conforme apresentado na Figura 1.

Com base no estudo mencionado anteriormente, o tempo ideal de carregamento é de três segundos. Acima desse valor, o usuário tende a desistir da utilização da aplicação, procurando por aplicações similares que o atendam nessa demanda.

Figura 1 – *Benchmarks* de diferentes faixas de tempo de carregamento em dispositivos móveis



Fonte: (AN, 2018).

A principal intenção desse Trabalho de Conclusão de Curso foi propor um conjunto de boas práticas visando minimizar e/ou mitigar essa preocupação com o carregamento das aplicações bem como outras questões associadas ao desempenho de aplicações *web front-end*.

1.3 Questão de pesquisa

Diante do exposto, ao final deste trabalho pretendeu-se responder a seguinte questão de pesquisa: Como prover aplicações *front-end* orientando-se pelos padrões estabelecidos na literatura especializada, e considerando indicadores e/ou métricas de desempenho bem como técnicas de otimização?

Dentre os autores da literatura especializada nesse tópico de pesquisa, destacam-se (ILIEV; DIMITROV, 2014), (SHROFF; CHAUDHARY, 2017), e (MIHAJLIJA, 2018).

1.4 Objetivos

1.4.1 Objetivo geral

Desenvolvimento de uma aplicação *web* que reúna a documentação elaborada como resultado da pesquisa deste trabalho a respeito de desempenho em aplicações *web front-end*, de modo a auxiliar equipes de desenvolvimento para que tenham conhecimento das métricas e técnicas existentes.

1.4.2 Objetivos específicos

- Identificação das principais métricas relativas ao desempenho no *front-end*;
- Identificação das principais técnicas relativas à otimização de desempenho no *front-end*, e
- Desenvolvimento de uma plataforma de código aberto, que confira uma base, se possível de referência, para os desenvolvedores brasileiros consultarem e melhorarem os indicadores de desempenho de suas aplicações.

1.5 Organização do trabalho

Este Trabalho de Conclusão de Curso está organizado nos seguintes capítulos:

- **Capítulo 2 - Referencial teórico:** apresenta a base teórica desse trabalho, com foco no nível arquitetural e no funcionamento básico de um navegador, bem como nas principais métricas utilizadas para aferir desempenho de aplicações *web front-end*;
- **Capítulo 3 - Suporte tecnológico:** apresenta as tecnologias utilizadas no desenvolvimento deste trabalho;
- **Capítulo 4 - Metodologia:** apresenta a metodologia que tem guiado a pesquisa e o desenvolvimento da aplicação proposta;
- **Capítulo 5 - O Guia:** descreve em detalhes o Guia voltado à otimização de desempenho em aplicações *web front-end*;
- **Capítulo 6 - Resultados:** apresenta os resultados alcançados durante os ciclos de pesquisa-ação conduzidos, e
- **Capítulo 7 - Conclusão:** apresenta as conclusões deste trabalho.

2 Referencial Teórico

Para melhor entendimento acerca do funcionamento de aplicações *web front-end*, este capítulo introduz a arquitetura de alto nível e o funcionamento básico de um navegador, bem como os processos necessários para transformação de código em um *website* funcional. Também é apresentada uma seção voltada às métricas que possibilitam avaliar desempenho e suas relações com a experiência de usuário. Por fim, são apresentadas as considerações finais do capítulo.

2.1 O navegador

Navegadores apresentam os recursos solicitados pelo usuário em forma de textos e imagens. Essas informações são solicitadas e recebidas do servidor, interpretadas e posteriormente exibidas pelo navegador (GARSIEL; IRISH, 2011).

Esta seção apresenta tópicos específicos para tanto acordar os principais elementos de um navegador, considerando o nível arquitetural; quanto para abordar o funcionamento básico de um navegador. Com isso, têm-se uma maior compreensão dos principais aspectos que podem influenciar na questão do desempenho de uma aplicação *web front-end*, e assim buscar por práticas que viabilizem otimizar esse desempenho.

2.1.1 Arquitetura do navegador

De acordo com (GARSIEL; IRISH, 2011), o navegador é estruturado como apresentado na Figura 2.

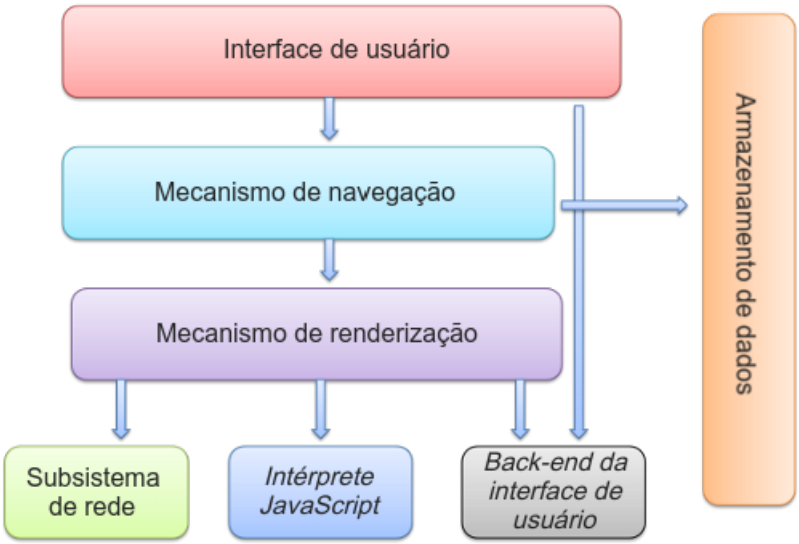
- **Interface de usuário** compreende todas as partes do navegador, incluindo a barra de endereço, os botões de voltar e avançar, o menu de favoritos, dentre outras. Não compreende a janela em que o conteúdo de uma página solicitada é exibido (GARSIEL; IRISH, 2011);

- **Mecanismo de navegação** conecta as ações entre a interface de usuário e o mecanismo de renderização (GARSIEL; IRISH, 2011);

- **Mecanismo de renderização** possui o analisador HTML e produz a representação visual do conteúdo solicitado pelo usuário através de uma URI. É capaz de exibir conteúdo em HTML, XML, estilizados com CSS ou não, documentos e imagens (GROSSKURTH; GODFREY, 2006);

- **Subsistema de rede** utilizado para realizar chamadas de rede, implementando os protocolos de transferência de arquivos, como HTTP e FTP. Também pode conter *cache* de recursos utilizados recentemente (GROSSKURTH; GODFREY, 2006);
- ***Back-end* da interface de usuário** contém elementos de interação com o usuário, como fontes, caixas de seleção e janelas. A construção de interface de usuário é provida pelo sistema operacional (GARSIEL; IRISH, 2011);
- **Intérprete *JavaScript*** analisa e executa código *JavaScript* contido nas páginas *web* (GARSIEL; IRISH, 2011), e
- **Armazenamento de dados** subsistema persistente que armazena dados no disco rígido associados à navegação, como *cookies*, *cache*, configurações do navegador ou páginas favoritas (GROSSKURTH; GODFREY, 2006).

Figura 2 – Arquitetura de um navegador



Fonte: (GARSIEL; IRISH, 2011).

Nas próximas seções, seguem outros detalhes quanto aos processos internos de um navegador, sendo: Processo de navegação e Processo de renderização.

2.1.2 Processo de navegação

O processo de navegação compreende desde a requisição do usuário por uma página *web* até o processo de preparação do navegador para renderizá-la (KOSAKA, 2018).

A navegação inicia-se assim que o usuário digita um endereço *web* na barra de endereços do navegador, ou clica em algum tipo de *link* (ILIEV; DIMITROV, 2014). O navegador determina o protocolo do endereço e o domínio. Em seguida, realiza uma consulta DNS para buscar o servidor que hospeda a página.

Assim que a conexão é estabelecida com o servidor, o navegador dispara a requisição para receber o conteúdo da página requisitada pelo usuário (ILIEV; DIMITROV, 2014). A partir do momento em que o navegador começa a receber os primeiros dados do servidor, começa o processo de **renderização** (KOSAKA, 2018).

2.1.3 Processo de renderização

O processo de renderização é responsável por transformar o conteúdo HTML, CSS e *JavaScript* em uma página *web* interativa (KOSAKA, 2018). A renderização ocorre em uma sequência única, exceto pelas operações de rede. Em navegadores como *Firefox* e *Safari*, esse processo representa a *main thread* do navegador (GARSIEL; IRISH, 2011).

Este processo é dividido em quatro etapas: análise do HTML e construção do DOM (*Document Object Model*); criação da árvore de renderização; *layout* e pintura (KOSAKA, 2018).

2.1.3.1 Análise do HTML e construção do DOM

De acordo com (HéGARET, 2000), o DOM é uma API que define a estrutura lógica de documentos HTML e XML, permitindo acessá-los e manipulá-los.

Cada documento é representado como uma árvore de nós. Alguns nós de uma árvore podem ou não ter filhos. A nomenclatura *object model*, ou modelo de objeto, foi definida orientando-se pelos princípios da orientação a objetos, onde cada nó representa um objeto, com métodos e identidade definidos (HéGARET, 2000).

A Figura 3 representa um trecho de código simples em HTML.

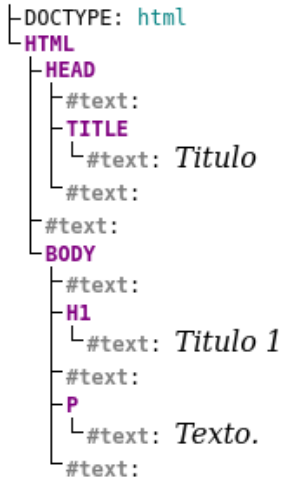
Figura 3 – Exemplo de código HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titulo</title>
    </head>
    <body>
        <h1>Titulo 1</h1>
        <p>Texto.</p>
    </body>
</html>
```

Fonte: Autora.

O trecho de código da Figura 3 resulta na árvore da Figura 4:

Figura 4 – Representação de uma árvore DOM



Fonte: Autora.

Após o recebimento dos recursos via rede e o disparo do processo de renderização feito pela navegação, o navegador começa a analisar o HTML e constrói a árvore DOM (KOSAKA, 2018).

2.1.3.2 Criação da árvore de renderização

No momento da construção da árvore DOM, o navegador também constrói uma segunda árvore, a de **renderização**. Ela dispõe os elementos visuais na ordem que deverão ser exibidos. A árvore de renderização possibilita a pintura do conteúdo da página na ordem correta (GARSIEL; IRISH, 2011).

Os elementos da árvore de renderização, ou renderizadores, correspondem a uma área retangular, e possuem informações geométricas como largura, altura e posição, definidas no CSS do nó. Um renderizador pode pintar e organizar a si mesmo no *layout*, bem como seus filhos (GARSIEL; IRISH, 2011).

2.1.3.3 *Layout*

Um renderizador recém criado e adicionado à árvore de renderização não possui posição nem tamanho. O cálculo desses valores é chamado de *layout*. A árvore de *layout* é criada através de uma travessia na árvore de renderização, que possui as coordenadas x e y dos elementos, bem como seus respectivos tamanhos (GARSIEL; IRISH, 2011).

Para determinar a geometria dos elementos, o navegador utiliza um sistema de coordenadas baseado no elemento raiz. Ele corresponde à janela de visualização do navegador, que representa a parte visível da janela (GARSIEL; IRISH, 2011).

A construção do *layout* ocorre de forma recursiva, iniciando-se na raiz do documento HTML, e prosseguindo pelos elementos da árvore de renderização (GARSIEL; IRISH, 2011).

Segundo Garsiel e Irish (2011), a etapa de *layout* possui os seguintes passos:

1. O renderizador pai define sua largura;
2. O renderizador pai calcula as coordenadas x e y de seu filho;
3. Elementos pais utilizam as alturas de seus filhos e as alturas de margens e preenchimentos para definir sua própria altura, que será utilizada pelo pai do renderizador pai, e
4. Por fim, o pai resolve seu *bit* incorreto como falso.

Um *bit* incorreto sinaliza que o navegador não precisa processar um *layout* por completo para cada pequena mudança. Um renderizador modificado ou adicionado define a si mesmo e seus filhos como "incorretos", indicando que precisam de *layout*, possibilitando que o processo se torne incremental (GARSIEL; IRISH, 2011).

2.1.3.4 Pintura

Ter o DOM, estilos e *layout* definidos, ainda não é suficiente para renderizar uma página. Apesar das formas, tamanhos e coordenadas serem conhecidos, ainda é necessário definir a ordem em que serão pintados (KOSAKA, 2018).

Na etapa de pintura, é necessário percorrer a árvore de renderização novamente, acionando o método de pintura dos renderizadores. A ordem de pintura é definida pelo CSS, que representa a ordem em que os elementos foram empilhados. Essa ordem afeta diretamente a pintura porque os elementos são pintados de trás para frente (GARSIEL; IRISH, 2011).

2.2 Métricas de desempenho

De acordo com uma pequisa conduzida pelo Google (GOOGLE, 2020a), os usuários têm 24% menos de probabilidade de abandonar um *website* durante o carregamento das páginas, quando este possui *Largest Contentful Paint* (LCP), *Cumulative Layout Shift* (CLS) e *First Input Delay* (FID) nos índices recomendados.

Existem diversos tipos de métricas que podem ajudar a classificar a forma como usuários experimentam desempenho em páginas *web*: velocidade, capacidade de resposta, capacidade de resposta à interação do usuário, estabilidade visual e suavidade (WALTON, 2019f).

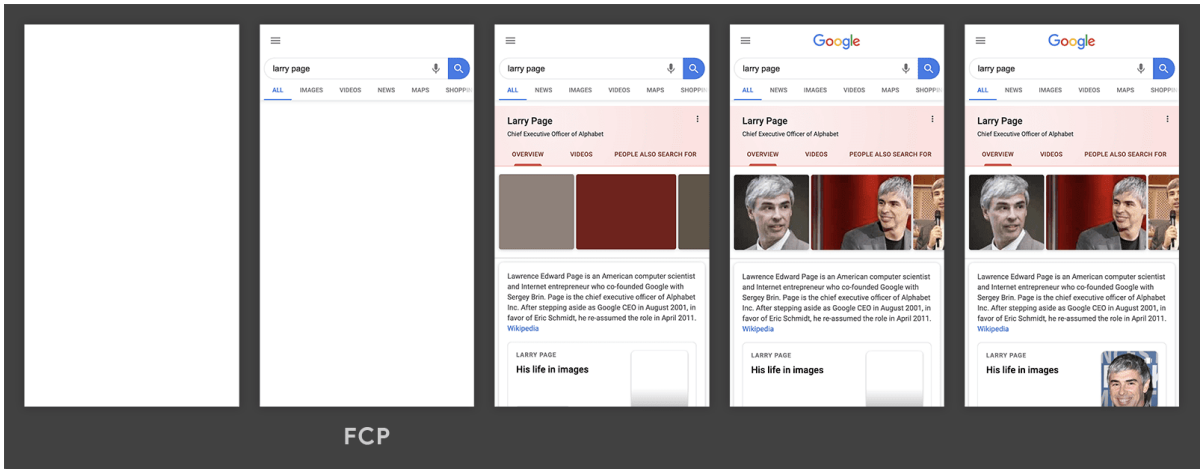
Esta seção aborda as principais métricas utilizadas para classificar qualitativamente a experiência de usuário de um *website*.

2.2.1 *First Contentful Paint* (FCP)

First contentful paint mede desde o início do carregamento de uma página *web* até o momento em que qualquer parte do conteúdo da página aparece na tela, podendo ser texto, imagens ou elementos *Scalable Vector Graphics* (SVG) (WALTON, 2019a).

A Figura 5 representa o processo de carregamento de uma página *web* e o momento em que a FCP ocorre, no segundo instante.

Figura 5 – Linha do tempo de carregamento de uma página *web*



Fonte: (WALTON, 2019a).

De acordo com Walton (2019a), para uma boa experiência de usuário, a FCP deve ocorrer dentro de 1 segundo após o início do carregamento da página.

2.2.2 *Largest Contentful Paint* (LCP)

A métrica *largest contentful paint* representa o tempo de renderização da maior imagem ou porção de texto visível dentro da janela. Se o elemento se estende além da janela, essas partes não serão consideradas no tamanho total do elemento (WALTON, 2019c).

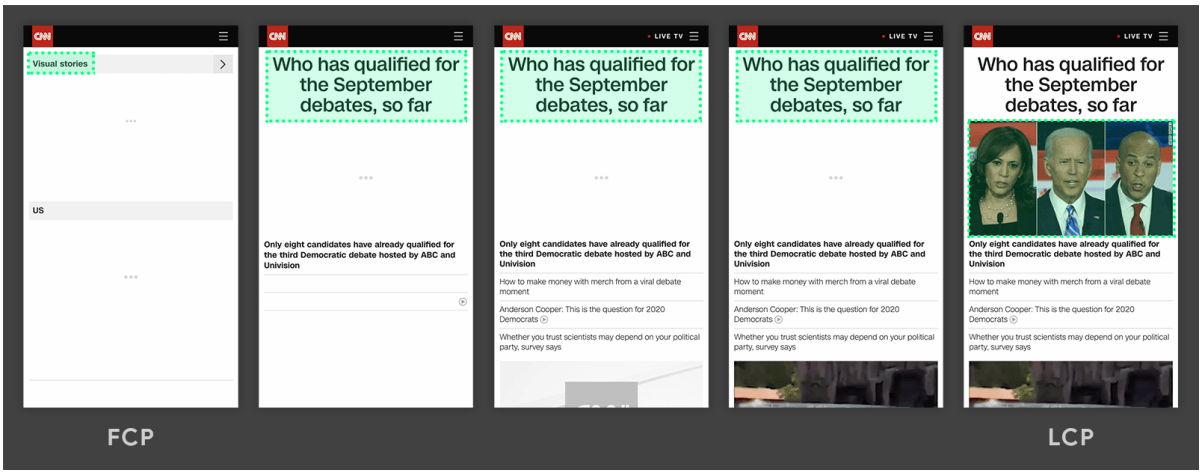
As páginas *web* geralmente são carregadas em partes, o que pode ocasionar mudança do maior elemento visível da janela. Um elemento só é considerado LCP se seu conteúdo já foi renderizado e está visível para o usuário. Imagens que ainda não estão visíveis não são consideradas (WALTON, 2019c).

Uma página que possui um texto e uma imagem, por exemplo, o navegador pode renderizar e exibir primeiramente o texto, classificando-o como LCP. Entretanto, assim que a imagem finalizar o carregamento, o navegador atualizará o elemento LCP como

sendo a imagem. O navegador suspenderá o rastreamento de possíveis novos elementos LCP assim que o usuário interagir com a página, através de rolagem ou cliques, por exemplo (WALTON, 2019c).

A Figura 6 representa a ocorrência da LCP em uma página *web*.

Figura 6 – Linha do tempo de carregamento de uma página *web*



Fonte: (WALTON, 2019c).

De acordo com Walton (2019c), para uma boa experiência de usuário, a LCP de uma página *web* deve ocorrer nos 2,5 primeiros segundos após o início do carregamento da mesma.

2.2.3 *First Input Delay* (FID)

First input delay mede o tempo da primeira interação do usuário com a página, como clique num *link* ou um botão, até o tempo em que o navegador está preparado para receber e responder a esse tipo de ação.

O atraso dos eventos de *input* geralmente está associado ao tamanho do arquivo *JavaScript* enviado ao navegador, que enquanto analisa e executa o código recebido via requisição, impede que o navegador execute os outros processos de renderização da página, pois o código *JavaScript* pode alterar o DOM (WALTON, 2019b).

De acordo com Walton (2019b), para uma boa experiência de usuário, a FID de uma página *web* deve ser menor que 100 milissegundos.

2.2.4 *Time to Interactive* (TTI)

Time to interactive mede o tempo desde o começo do carregamento da página até o momento em que os seus recursos foram carregados e são capazes de responder às interações do usuário, ou seja, o tempo necessário para que a página se torne completamente visível e interativa (WALTON, 2019d).

O Google (2019) define os seguintes critérios para uma página ser considerada completamente interativa:

- Exibe conteúdo, medido através do FCP;
- A página já pode receber eventos de *input* para a maioria dos elementos visíveis, e
- A página responde às interações do usuário dentro de 50 milissegundos.

De acordo com Walton (2019d), para uma boa experiência de usuário, a TTI de uma página *web* deve ser menor que 5 segundos.

2.2.5 *Total Blocking Time* (TBT)

Total blocking time mede o tempo decorrido entre a FCP e a TTI, indicando que a *thread* principal do navegador ficou bloqueada a ponto de impedi-lo de responder às interações do usuário (WALTON, 2019e).

A *thread* principal é considerada bloqueada quando a duração de um processo ultrapassa 50 milissegundos. O navegador não pode interromper uma tarefa em progresso. Portanto, se o usuário interagir com a página durante um processo em andamento, não receberá resposta até o processo estar completo. A TBT é a soma da duração de todos os bloqueios que excedem 50 milissegundos ocorridos entre a FCP e a TTI (WALTON, 2019e).

De acordo com Walton (2019e), para uma boa experiência de usuário, a TBT de uma página *web* deve ser menor que 300 milissegundos.

2.2.6 *Cumulative Layout Shift* (CLS)

Cumulative layout shift mede a pontuação cumulativa de todas as mudanças de *layout* inesperadas que ocorrem durante a vida útil de uma página *web* (WALTON; MIHAJLIJA, 2019).

Mudanças de *layout* ocorrem apenas quando os elementos existentes mudam sua posição inicial. Se um novo elemento é adicionado ao DOM ou um elemento existente muda de tamanho, isso não conta como uma mudança de *layout* (WALTON; MIHAJLIJA, 2019).

De acordo com Walton e Mihajlija (2019), para uma boa experiência de usuário, a CLS de uma página *web* deve ser menor que 0.1.

2.3 Fatores que impactam desempenho

Diversos fatores podem impactar na forma como o usuário percebe desempenho ao acessar uma página *web*, e podem ser considerados durante o desenvolvimento. A seguir estão classificados alguns dos fatores comumente abordados pela comunidade:

- ***JavaScript***: Quando o usuário solicita uma página, o navegador busca o HTML, busca o CSS, e gera uma árvore de renderização combinando o DOM e o CSSOM. Se a página necessita de *JavaScript*, o navegador não começará a renderizá-la até que o mesmo seja executado, atrasando a renderização (FRIEDMAN, 2020).

- **CSS**: O navegador deve baixar e fazer o *parse* dos arquivos CSS antes de mostrar a página, tornando o CSS um dos recursos que podem bloquear o processo de renderização. Se os arquivos CSS forem muito grandes, ou a rede for de baixa velocidade, os *downloads* de arquivos CSS podem aumentar significativamente o tempo de renderização de uma página *web* (MIHAJLIJA, 2019).

- **Arquivos de imagem**: Imagens são o tipo de arquivo mais solicitado para a maioria das páginas *web* e costumam apresentar maiores tempos de *download* do que qualquer outro recurso. No 90º percentil, os sites enviam cerca de 4,7 MB de imagens em computadores e *smartphones* (DJIRDEH; OSMANI; BYNENS, 2020).

É importante ressaltar que estes não são os únicos fatores que podem impactar o tempo de carregamento de uma página *web*, e que podem variar dependendo da tecnologia em que a aplicação foi desenvolvida; como os arquivos foram entregues ao cliente; o navegador utilizado, e o *hardware* em que o *website* foi acessado, como computadores, *tablets* ou *smartphones*.

2.4 Considerações finais do capítulo

De acordo com um estudo conduzido pelo *Google* (GOOGLE, 2020a), desempenho impacta na experiência de usuário, e pode influenciar a navegação e a permanência do usuário em uma página *web*.

Diante do exposto, para melhor compreensão acerca de otimização de desempenho em aplicações *web front-end*, é importante que se conheça os aspectos internos do navegador. Através dele que é possível interagir com páginas *web* criadas a partir de tecnologias como HTML, CSS e *JavaScript* (GARSIEL; IRISH, 2011). O capítulo expõe os processos e como funcionam, desde a primeira requisição para buscar o conteúdo da página, até o ultimo processo, que pinta os elementos posicionados.

As métricas ajudam a revelar os pontos de melhoria e a compreender o processo de otimização de desempenho de uma aplicação. Através da iniciativa *Core Web Vitals*(GOOGLE, 2020b), liderada pelo *Google*, desempenho em aplicações *web front-end* possui diversas métricas propostas, apresentadas ao longo deste capítulo.

3 Suporte Tecnológico

Este capítulo introduz as ferramentas e tecnologias que foram utilizadas na construção deste trabalho, e estão divididas em seções que compreendem desde a concepção dos protótipos de alta fidelidade do Guia desenvolvido, até a ferramenta de controle de versão utilizada. Ao final, têm-se as considerações finais do capítulo.

3.1 *Figma*

Atualmente, existem diversas ferramentas no mercado que permitem criar interfaces de usuário completas e até mesmo interativas, como o *Adobe XD*[1], *Invision*[2], *Figma*[3], entre outras.

A ferramenta escolhida, o *Figma*, é uma ferramenta *online* de prototipagem e permite criar *design systems* completos usando apenas o navegador. O *Figma* também salva e armazena os projetos em nuvem, além de permitir criar protótipos interativos e compartilhá-los na *web*. Foi utilizado para criar os protótipos de alta fidelidade da aplicação *web* resultado deste trabalho.

3.2 *Gridsome* 0.3.4

O Guia fruto deste trabalho foi criado utilizando um *framework JavaScript*. Atualmente, existem diversas opções disponíveis que facilitam a criação de uma aplicação *web front-end*, tais como *Vue.js*[4], *Next.js*[5], *Gridsome*[6], entre diversas outras.

A escolha da tecnologia foi conferida orientando-se pelos seguintes requisitos: curva de aprendizado reduzida; familiaridade com ferramentas já utilizadas pela autora; suporte à *Static Site Generation* (SSG), que gera páginas estáticas em tempo de *build*, e também anula a necessidade de hospedar a aplicação em um servidor, e tamanho do *bundle* menor, favorecendo tempos de carregamentos reduzidos.

Gridsome é um *framework JavaScript* de código aberto, baseado em *Vue.js*, que permite criar aplicações *web front-end* de alto desempenho. Facilita ainda a criação de aplicações de arquitetura modular e permite geração de páginas estáticas em tempo de *build*.

[1] https://www.adobe.com/br/products/xd.html. Acessado pela última vez em 03/05/2021.
[2] https://www.invisionapp.com. Acessado pela última vez em 03/05/2021.
[3] https://www.figma.com. Acessado pela última vez em 03/05/2021.
[4] https://vuejs.org. Acessado pela última vez em 03/05/2021.
[5] https://nextjs.org. Acessado pela última vez em 03/05/2021.
[6] https://gridsome.org. Acessado pela última vez em 03/05/2021.

3.3 *Visual Studio Code* 1.50.1

O *Visual Studio Code*[7] é um editor de código leve, aberto, e está disponível para os sistemas operacionais *Windows*, *macOS* e *Linux*. Possui suporte nativo para Git, *debugger* integrado, e *intellisense*, que pode autocompletar código baseado em nomes de variáveis e funções, por exemplo. Foi utilizado para apoiar o desenvolvimento do código do Guia escrito em *JavaScript*.

3.4 Git 2.28.0

O controle de versão registra as alterações de um arquivo ao longo do tempo. Permite que os arquivos sejam restaurados a estados anteriores, pode exibir as alterações ao longo do tempo, juntamente com quem as realizou, entre outras funcionalidades (CHACON; STRAUB, 2014). O Git[8] é um sistema de código aberto de controle de versão, projetado para trabalhar em projetos de grande ou pequeno porte, e foi utilizado na construção do Guia.

3.5 GitHub

O GitHub[9] é uma plataforma de hospedagem de código que permite além de hospedar, também revisar código; gerenciar projetos, e realizar integração contínua e *deploy* contínuo. É gratuito para projetos de código aberto, sendo utilizado na hospedagem do código do Guia.

3.6 Considerações finais do capítulo

A Tabela 1 lista e descreve as tecnologias que foram utilizadas no desenvolvimento do *software*.

Tabela 1 – Tecnologias utilizadas para o desenvolvimento do Guia

Nome	Descrição	Versão
Figma	Ferramenta de construção de interfaces de usuário	-
Gridsome	*Framework JavaScript*	0.3.4
Visual Studio Code	Editor de código	1.50.1
Git	Ferramenta de controle de versão	2.28.0
GitHub	Ferramenta online que hospeda projetos Git	-

[7] https://code.visualstudio.com. Acessado pela última vez em 03/05/2021.
[8] https://git-scm.com. Acessado pela última vez em 03/05/2021.
[9] https://github.com. Acessado pela última vez em 03/05/2021.

4 Metodologia

Este capitulo apresenta o detalhamento metodológico que direcionou esse Trabalho de Conclusão de Curso em sua totalidade. Em um primeiro momento, é conferida a classificação da pesquisa quanto a vários critérios. Segue a apresentação do fluxo de atividades, compreendendo o escopo do TCC_01 e do TCC_02. Mais adiante, o capítulo aborda as metodologias que orientam: o levantamento bibliográfico; o desenvolvimento do Guia, e a análise de resultados. Há ainda menção ao cronograma, permitindo maior entendimento de como as atividades foram distribuídas em termos temporais. Por fim, têm-se as considerações finais do capítulo.

4.1 Pesquisa

Uma pesquisa científica tem como objetivo resolver um problema através de um exame rigoroso, utilizando-se de procedimentos científicos. Em (GERHARDT; SILVEIRA, 2009), é colocado que uma pesquisa pode ser classificada quanto à **abordagem**, **natureza**, **objetivos** e **procedimentos**.

Esta seção apresenta os tipos de pesquisa que apoiaram o desenvolvimento deste trabalho de acordo com as classificações apresentadas por Gerhardt e Silveira (2009).

4.1.1 Abordagem

A pesquisa deste trabalho pode ser classificada como **qualitativa** e **quantitaiva**. A pesquisa quantitativa possui procedimentos estruturados para coletar dados, sob condições controladas, além de analisar os dados numéricos utilizando métodos estatísticos. Já a pesquisa qualitativa não quantifica os valores, porque tenta compreender o fenômeno como um todo, e não apenas determinados conceitos (GERHARDT; SILVEIRA, 2009).

4.1.2 Natureza

A natureza da pesquisa deste trabalho pode ser classificada como **aplicada**. De acordo com Gerhardt e Silveira (2009), uma pesquisa de natureza básica se propõe-se a criar novos conhecimentos e não tem aplicação pŕatica; enquanto uma pesquisa de natureza aplicada pretende solucionar problemas específicos, de aplicação prática.

4.1.3 Objetivos

Quanto aos objetivos, uma pesquisa pode ser classificada como exploratória, descritiva ou explicativa. Uma pesquisa exploratória é flexível e pode envolver levantamento bibliográfico, entrevistas e análise de exemplos. Uma pesquisa descritiva têm como objetivo identificar as características de um grupo, como por exemplo pesquisas eleitorais. Uma pesquisa explicativa tem como objetivo explicar a razão das coisas e identificar fatores que contribuem para a ocorrência de fenômenos (GIL, 2017).

Esta pesquisa pode ser classificada como **exploratória**.

4.1.4 Procedimentos

Uma pesquisa bibliográfica é elaborada com base em material já analisado e publicado, seja em livros, artigos científicos, entre outros. A pesquisa bibliográfica ajuda o pesquisador a conhecer o que já foi estudado acerca do assunto, podendo fornecer informações prévias do problema no qual são buscadas as respostas (GERHARDT; SILVEIRA, 2009).

Dentre os métodos de pesquisa disponíveis, esta pesquisa pode ser classificada como uma **pesquisa bibliográfica**.

4.2 Fluxo das atividades

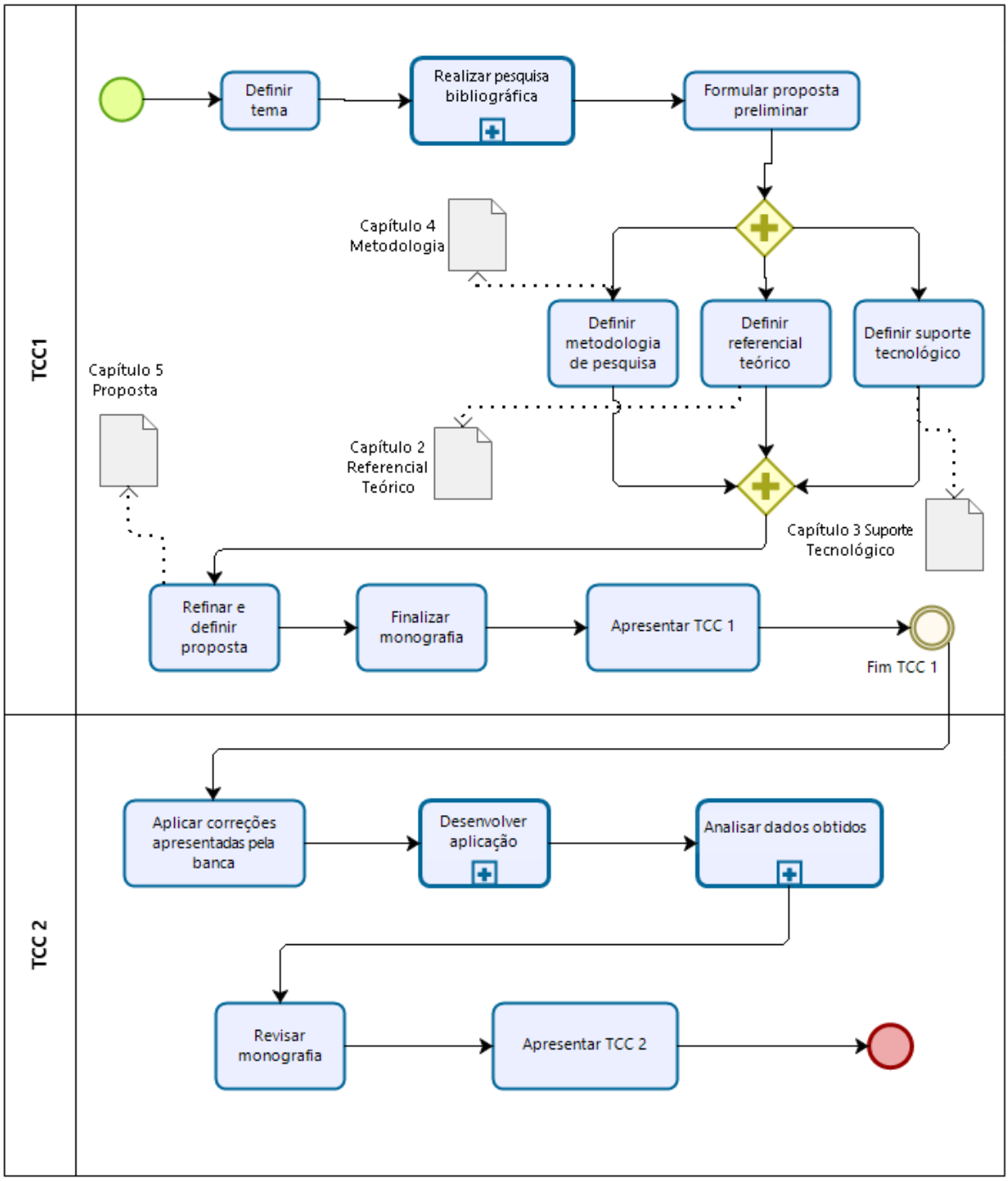
O presente trabalho foi conduzido dentro de etapas, algumas delas resultaram nos capítulos de Referencial Teórico e Suporte Tecnológico. A Figura 7 ilustra o processo de condução do trabalho.

- **Definir tema:** atividade já realizada, executada juntamente com os orientadores, a fim de encontrar um campo de interesse e afinidade por parte da autora, que pudesse resultar numa hipótese a ser explorada. Optou-se por pesquisar a respeito de otimização de desempenho de aplicações *web front-end*;

- **Realizar pesquisa bibliográfica:** atividade já realizada com o intuito de reunir material publicado sobre o tema escolhido de desempenho em aplicações *front-end*. Essa atividade foi orientada pela metodologia de pesquisa bibliográfica descrita na seção 4.3 deste capítulo;

- **Formular proposta preliminar:** após a definição do problema que será solucionado, definiu-se uma solução preliminar, de cunho prático, resultando em uma aplicação que reúne as práticas de otimização de desempenho;

Figura 7 – Processo de atividades do Trabalho de Conclusão de Curso.



Fonte: Autora.

- **Definir referencial teórico:** atividade já realizada, que resultou no capítulo de Referencial Teórico, através de material publicado por autores da literatura especializada em desempenho de aplicações *web front-end*;

- **Definir metodologia de pesquisa:** definido o referencial teórico, estabeleceu-se a metodologia que apoia o trabalho, segundo abordagem, natureza, objetivos e procedimentos;

- **Definir suporte tecnológico:** atividade já realizada, que resultou no capítulo de Suporte Tecnológico, que lista as tecnologias e ferramentas que têm sido utilizadas no desenvolvimento da aplicação proposta;
- **Refinar e definir proposta:** com melhor entendimento acerca do tema, após pesquisa bibliográfica e suporte tecnológico definido, refatorar e evoluir a proposta inicial sugerida, detalhada no capítulo O Guia;
- **Finalizar monografia:** aplicar últimas correções do trabalho apontadas pelos orientadores;
- **Apresentar TCC 1:** atividade já realizada. Consiste na submissão do texto à banca e na apresentação para avaliação;
- **Aplicar correções apresentadas pela banca:** atividade já realizada. Compreende as adaptações do trabalho de acordo com a análise fornecida pela banca;
- **Desenvolver aplicação:** atividade já realizada. Compreende a implementação da aplicação proposta de acordo com o processo de desenvolvimento definido. Essa atividade será orientada pela metodologia de desenvolvimento descrita na seção 4.4 deste capítulo;
- **Analisar dados obtidos:** atividade já realizada. Compreende o estudo dos resultados da aplicação, verificando se a mesma cumpriu os objetivos do trabalho. Essa atividade será orientada pela metodologia de análise de resultados descrita na seção 4.5 deste capítulo;
- **Revisar monografia:** atividade já realizada. Compreende a aplicação de novas correções do trabalho apontadas pelos orientadores, e
- **Apresentar TCC 2:** atividade já realizada. Compreende a apresentação final do trabalho junto à banca.

4.3 Metodologia de pesquisa bibliográfica

De acordo com Gil (GIL, 2017), a pesquisa bibliográfica é uma sequência de etapas que podem variar de acordo com a natureza do problema, entre outros fatores. Uma pesquisa bibliográfica geralmente contém as seguintes etapas: escolha do tema; levantamento bibliográfico preliminar; formulação do problema; elaboração do plano provisório de assunto; busca das fontes; leitura do material; fichamento; organização do assunto, e redação do texto (GIL, 2017).

Após a definição do tema, otimização de desempenho em aplicações *web front-end*, realizou-se a busca de materiais relacionados nas bases de conhecimento IEEE e ACM. A

Tabela 2 lista as *strings* de busca utilizadas durante a pesquisa bibliográfica bem como as bases utilizadas e o número de resultados brutos encontrados.

Tabela 2 – *Strings* de busca utilizadas na busca das fontes.

Base	***String* de busca**	**Nº de artigos obtidos**
ACM	"front-end" AND "performance" AND "optimization", 2005 - 2020	4033
IEEE	"front-end" AND "performance", 2005 - 2021	5165
IEEE	"front-end" AND "website" AND "performance"	8
IEEE	"front-end" AND "optimization", 2005 - 2021	862
IEEE	"browser optimization render"	25
IEEE	"web browser" AND "performance evaluation"	43
IEEE	"web browser" AND "architecture"	418

4.3.1 Critérios de seleção

Após a triagem de parte dos artigos resultantes das pesquisas, iniciou-se o processo de leitura e seleção. Gil (GIL, 2017) classifica a leitura do material nos seguintes tipos: exploratória; seletiva; analítica, e interpretativa.

A leitura exploratória filtrou o material após leitura do resumo e palavras chave dos artigos obtidos das bases, e seguiu os seguintes critérios:

- Relacionado a desenvolvimento *web*;
- Relacionado a desempenho de páginas *web*;
- Relacionado a tempos de resposta em páginas *web*, e
- Relacionado a navegadores *web*.

O tema otimização de desempenho em aplicações *web front-end* ganhou relevância nos últimos anos e, portanto, não foi encontrado um grande número nas bases de conhecimento, como listado na Tabela 3.

4.3.1.1 Resultados

Após leitura seletiva, foram selecionados os seguintes artigos como referência:

- *Front end optimization methods and their effect* (ILIEV; DIMITROV, 2014), e
- *Critical rendering path optimizations to reduce the web page loading time* (SHROFF; CHAUDHARY, 2017).

É importante ressaltar que os artigos selecionados guiaram a pesquisa, bem como suas referências, entretanto, esta pesquisa também utilizou referências de autores que promovem a iniciativa Web.Vitals[1], liderada pelo Google, como Mihajlija.

4.4 Metodologia de desenvolvimento

Schwaber (2004) define o Scrum como um *framework* e uma série de práticas que mantêm o trabalho visível, permitindo à equipe ajustar o projeto à medida que este avança, visando os objetivos definidos inicialmente. O Scrum define papéis, como o *Scrum Master*, o time e o Dono do Produto; *sprint*, uma iteração onde o trabalho priorizado pelo Dono do Produto é realizado, e artefatos como o *backlog* do produto.

O método Kanban utiliza um quadro que exibe diferentes fases do processo de desenvolvimento. Não possui papéis definidos, nem reuniões e artefatos, como o *Scrum*. Os elementos básicos do Kanban são: visualização do fluxo de trabalho em quadros; o trabalho em andamento é limitado em pequenos blocos, não possuindo o conceito de *sprints*; cultura de *feedback* com revisões periódicas de entregáveis, e melhoria contínua (ALQUDAH; RAZALI, 2017).

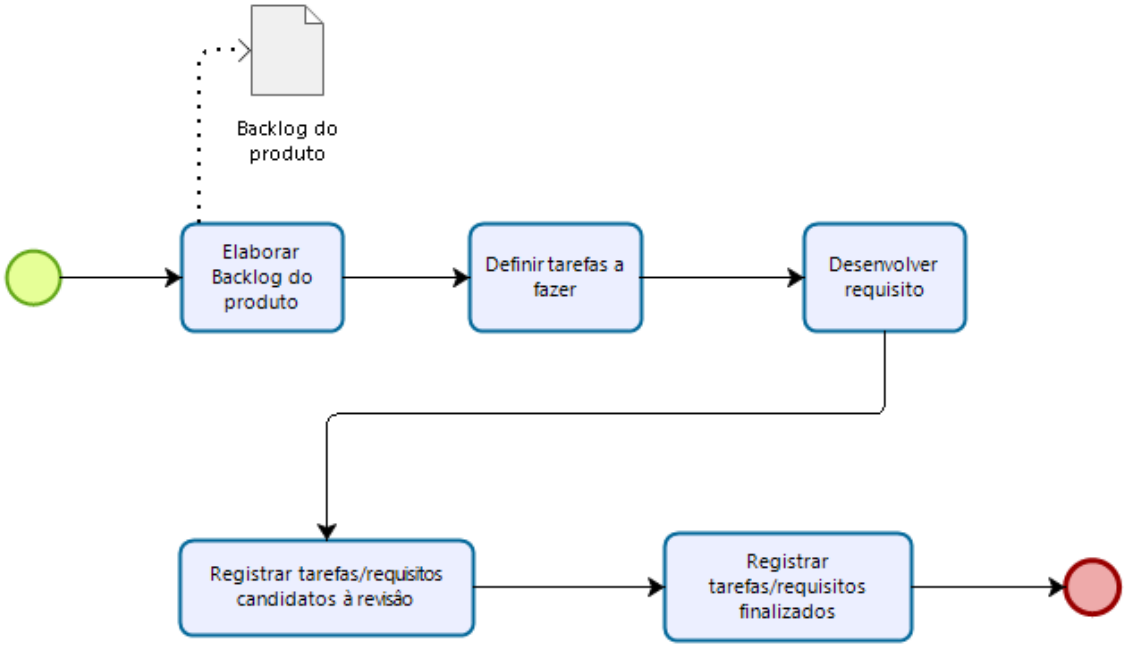
Em vista disso, o desenvolvimento da aplicação foi orientado pelos princípios ágeis, como entregas frequentes e aceitação rápida das mudanças de requisitos durante o processo de desenvolvimento (WILLIAMS, 2010). A abordagem metodológica foi orientada por uma mistura entre *Kanban* e *Scrum*. *Kanban* por este ser mais flexível em relação ao *Scrum*, pois não exige papéis e não definem iterações de tempo com *sprints*, correspondendo, portanto, à forma com que a autora pretende conduzir o trabalho. *Scrum*, pelo *backlog* do produto, que reúne todos os requisitos do projeto, além de ser dinâmico e evoluir juntamente com o produto (SCHWABER, 2004).

O processo de desenvolvimento do Guia foi executado dentro das fases ilustradas na Figura 8.

- ***Backlog* do produto:** Lista todos os requisitos da aplicação entre outras tarefas necessárias;
- **A fazer:** Requisitos ou tarefas que são mais importantes e estão priorizadas para desenvolvimento próximo. Pode mudar de acordo com o parecer dos orientadores das reuniões periódicas de revisão;
- **Em progresso:** Tarefas que estão em desenvolvimento;
- **A revisar:** Tarefa finalizada e que pode ser revisada pelos orientadores ou pelo desenvolvedor para aprimoramento ou re-priorização, e

[1] https://web.dev/learn-web-vitals. Acessado pela última vez em 03/05/2021.

Figura 8 – Processo de desenvolvimento da proposta deste Trabalho de Conclusão de Curso.



Fonte: Autora.

- **Feito:** Tarefa finalizada e seu resultado pode ser incorporado à versão de produção do *software*.

4.5 Metodologia de análise de resultados

Segundo Gil, a pesquisa-ação representa "uma modalidade de pesquisa que não se ajusta ao modelo clássico de pesquisa científica, cujo propósito é o de proporcionar a aquisição de conhecimentos claros, precisos e objetivos" (GIL, 2017, p. 40).

A pesquisa-ação não apresenta sequência cronológica em suas ações, entretanto, podem se enquadrar nas seguintes etapas: formulação do problema; construção de hipóteses; realização do seminário; seleção da amostra; coleta de dados; análise e interpretação dos dados; elaboração do plano de ação, e divulgação dos resultados (GIL, 2017).

Em vista disso, foi definido um protocolo de pesquisa-ação dado seu caráter cíclico, que coincide com a metodologia iterativa de desenvolvimento da aplicação apresentada na seção 4.4 deste capítulo.

- **Coleta de dados**: coletar dados através de cenários de uso previamente planejados;
- **Análise e interpretação dos dados**: avaliar os dados coletados e elaborar a interpretação dos dados obtidos empiricamente;

- **Elaboração do plano de ação**: consiste no planejamento de uma ação para o problema alvo da investigação, e
- **Divulgação dos resultados**: documentar os resultados obtidos ao final do ciclo.

4.6 Cronograma

As Tabelas 3 e 4 exibem os cronogramas das atividades executadas nos TCC's 1 e 2.

Tabela 3 – Cronograma do Trabalho de Conclusão de Curso 1

Atividades	**Ago/2020**	**Set/2020**	**Out/2020**	**Nov/2020**	**Dez/2020**
Definir tema	X				
Introdução	X				
Referencial teórico		X			
Suporte tecnológico		X			
Metodologia			X	X	
Refinar proposta				X	
Finalizar monografia				X	X
Apresentar à banca					X

Tabela 4 – Cronograma do Trabalho de Conclusão de Curso 2

Atividades	**Fev/2021**	**Mar/2021**	**Abr/2021**	**Mai/2021**
Aplicar correções	X			
Desenvolver o Guia	X	X		
Coletar e analisar resultados			X	
Revisar e finalizar monografia			X	X
Apresentar à banca				X

4.7 Considerações finais do capítulo

Este capítulo apresentou a metodologia que apoiou este trabalho, com sua metodologia de pesquisa sendo classificada como: qualitativa e quantitativa, de natureza aplicada, com o objetivo de ser uma pesquisa exploratória e com procedimentos de pesquisa-ação e pesquisa bibliográfica.

Definiu-se ainda a metodologia ágil como a metodologia que guiou o desenvolvimento da aplicação de *software*, através do uso de aspectos do *Scrum* aliado ao método *Kanban*, por este ser mais flexível que o *Scrum* além de facilitar a priorização contínua de requisitos (ALQUDAH; RAZALI, 2017).

5 O Guia

Este capítulo apresenta, em detalhes, o Guia voltado à otimização de desempenho em aplicações *web front-end* que foi desenvolvido neste Trabalho de Conclusão de Curso. Primeiramente, é apresentada uma contextualização acerca da motivação que originou a pesquisa no tema desempenho de aplicações *web front-end*. Em seguida, são cobertos trabalhos relacionados, os quais possuem similaridades com a aplicação que foi desenvolvida. Adicionalmente, é apresentada a prova de conceito, cobrindo aspectos de *design*, juntamente com o estudo prévio que apoiou as decisões acerca de UI e UX.

Também são apresentados os requisitos que guiaram o desenvolvimento da aplicação *web*, além dos requisitos que possibilitaram a redação dos textos, seguidos de uma apresentação da aplicação finalizada e disponível ao público. Por fim, são apresentadas as considerações finais do capítulo.

5.1 Contextualização

A motivação que definiu o tema desempenho em aplicações *web front-end* originou-se a partir de uma experiência vivida em um contexto real pela autora do presente trabalho, que atuou à frente do desenvolvimento de uma aplicação *web* como líder da equipe de desenvolvimento *front-end*, composta por outros dois desenvolvedores.

O produto era oferecido por uma empresa de pequeno porte, que tinha como propósito ajudar imobiliárias a se digitalizarem, otimizando seus processos internos, como gerenciamento de visitas e propostas feitas a determinados imóveis, por exemplo. A aplicação foi criada como um MVP (*Minimum Viable Project*) no início de 2019, e cresceu rapidamente à medida que mais imobiliárias conheciam a proposta da empresa e decidiam contratar o produto, partindo de dois clientes desde o início de sua operação à mais de 40 clientes atualmente.

A aplicação possuía um tamanho de pequeno a médio porte e foi construída sobre um *framework JavaScript*, crescendo rapidamente devido à demanda. Em fevereiro de 2020, a autora começou a notar problemas de desempenho, principalmente, em dispositivos móveis, onde o tempo de carregamento chegava a atingir 30 segundos, conforme testes realizados pela própria autora na época. Um primeiro diagnóstico superficial, levantado pela autora sem conhecimento prévio acerca das métricas e técnicas de otimização, identificou problemas como: alta repetição de código *JavaScript*, seja em forma de componentes criados pelos próprios desenvolvedores, seja pelo uso excessivo de bibliotecas criadas por terceiros, além de uma série de arquivos de imagem em altíssima resolução, chegando a

mais de 2 MB em tamanho.

Diante disso, pesquisas foram iniciadas com o intuito de descobrir técnicas e/ou ferramentas que pudessem auxiliar um possível processo de refatoração da aplicação em questão, visando à redução do tempo de carregamento das páginas. Nessas pesquisas, foi possível encontrar alguns trabalhos relacionados como o *Web Vitals* (GOOGLE, 2020b), o *Front-End Performance Checklist* (FRIEDMAN, 2020), e o *Browser Diet* (ROCHA, 2013a). Dentre esses trabalhos, há um projeto, em particular, que se assemelha ao Guia, fruto desse Trabalho de Conclusão de Curso, conforme apresentado a seguir.

5.1.1 Trabalhos relacionados

O projeto *Browser Diet* (ROCHA, 2013a) é idealizado por Zeno Rocha, desenvolvedor brasileiro. O objetivo do projeto é fornecer um guia para desenvolvedores otimizarem desempenho de páginas *web*. A motivação para dar início ao projeto, como ele mesmo descreve, surgiu de uma experiência pessoal, por volta de 2012 (ROCHA, 2013b). Zeno passava muito tempo viajando, e portanto, não tinha acesso à internet de alta velocidade. Ele percebeu que os *websites* que acessava demoravam muito para carregar. Entretanto, também percebeu que este problema não estava relacionado, em sua maioria, à velocidade da *internet* a qual estava conectado, mas sim ao tamanho das páginas que acessava.

Lançado ao público em 2013, o projeto uniu a monografia de Zeno à sua motivação inicial e resultou num guia completo, com técnicas que poderiam ajudar desenvolvedores a resolver os problemas de desempenho de aplicações *web*. O projeto contou com a participação de outros desenvolvedores brasileiros, que ajudaram Zeno a redigir o conteúdo. O *site* foi criado como um projeto de código aberto, permitindo à comunidade evoluir seu conteúdo.

A Figura 9 lista os principais projetos similares ao proposto neste trabalho, com base nas características consideradas mais importantes, na visão da autora, para um guia informativo de otimização de desempenho no *front-end*.

Como apontado na Figura 9, o projeto que mais se assemelha à proposta deste trabalho é o *Browser Diet*, por este ser voltado à comunidade brasileira, listar técnicas de otimização e apresentar ferramentas úteis ao diagnóstico, além de ser *open-source*. Entretanto, o projeto não é atualizado há mais de um ano seguindo o próprio histórico de versão do repositório do projeto, fazendo com que seu conteúdo não contenha os avanços mais recentes a respeito de otimização de páginas *web*, como informações mais detalhadas a respeito das métricas de desempenho, por exemplo.

Figura 9 – Comparativo entre projetos similares ao proposto neste trabalho

Projeto	Data do projeto	Idioma do conteúdo	Código aberto	Lista e explicita métricas	Descreve casos de sucesso
BrowserDiet	2013	Português	✓	✗	✗
Web Vitals	2020	Inglês	✓	✓	✓
Front-End Performance Checklist	2020	Inglês	✗	✓	✗

Fonte: Autora.

5.2 Prova de conceito

Esta seção apresenta os tópicos que compõem o guia, bem como os protótipos de média/alta fidelidade concebidos, e exibe os recursos com os quais os desenvolvedores conhecerão diferentes métricas e técnicas que podem ajudar a otimizar suas aplicações *front-end*, além de poderem contribuir com a evolução da ferramenta, pois a mesma será de código aberto.

5.2.1 *Design* da aplicação

Como descrito no Capítulo de Suporte Tecnológico, o *design* do Guia foi concebido no *Figma*. Para uma primeira versão, um estudo prévio foi conduzido com o intuito de buscar inspiração em *designs* modernos. No entanto, ainda simples, composto por uma *landing page*, uma tela de listagem de conteúdos e, por fim, uma tela de exibição de texto em detalhes, com menu lateral de navegação por tópicos.

Cores são importantes na criação de uma marca. Diferentes tons de cores podem mudar o tipo de influência exercida sobre uma determinada audiência, de acordo com a psicologia das cores (ART THERAPY, 2020). Cores quentes, como vermelho e amarelo podem invocar emoções desde conforto à raiva. Cores frias, como o azul e roxo podem indicar desde calmaria até indiferença (IMTIAZ, 2016). A cor primária do Guia é azul, uma cor fria, por representar confiabilidade, e por causar efeito relaxante, o que condiz com a proposta de simplicidade do trabalho (IMTIAZ, 2016). A cor secundária, rosa, uma cor quente, que contrasta com o azul e é uma cor suave. Vale ressaltar que a paleta de cores definida também é resultado da percepção da autora do presente trabalho.

A Figura 10 apresenta a paleta de cores definida para o Guia após estudo prévio.

Uma boa parte do conteúdo da *web* é composto por texto. A primeira versão do CSS, em 1997, já fazia menção às fontes. À época, as fontes escolhidas por *designers* precisavam estar instaladas na máquina do usuário para que pudessem ser vistas nos

Figura 10 – Paleta de cores definida.

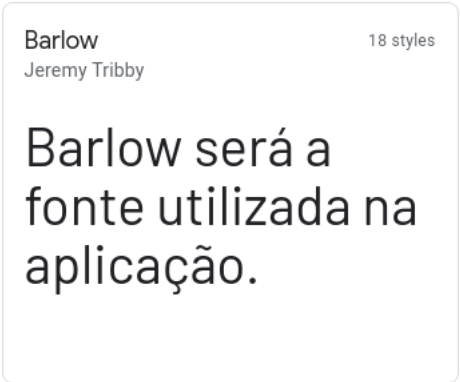
Fonte: Autora.

websites. Para resolver o problema, o CSS2 introduziu uma nova regra, o *@font-face*, que permite aos desenvolvedores manter um elo da URL de um arquivo de fonte para que os navegadores possam baixá-las, assim os usuários poderiam ver as fontes sem a necessidade de instalá-las previamente em suas máquinas (HOFFMANN, 2017).

De acordo com Maria (2014), tipografia não possui regras, mas é um conjunto de princípios e boas práticas. Tipografia é importante para passar uma mensagem, e não é apenas uma característica que serve unicamente para complementar um *design*. Utilizar mais de uma fonte pode, na verdade, enfraquecer um *design*. O recomendado é utilizar de uma a duas fontes diferentes. Apenas uma fonte foi utilizada para os textos do Guia, a *Barlow*. Isso porque é uma fonte do grupo *sans serif*, favorável a títulos e textos. Fontes também impactam em desempenho, pois precisam ser baixadas assim que a página é acessada, e precisam estar visíveis o quanto antes na tela para permitir que o usuário leia seu conteúdo (GRIGORIK, 2019).

A Figura 11 ilustra a fonte apresenta os textos do guia.

Figura 11 – Fonte estilo *sans-serif* utilizada nos textos: *Barlow*



Fonte: Autora.

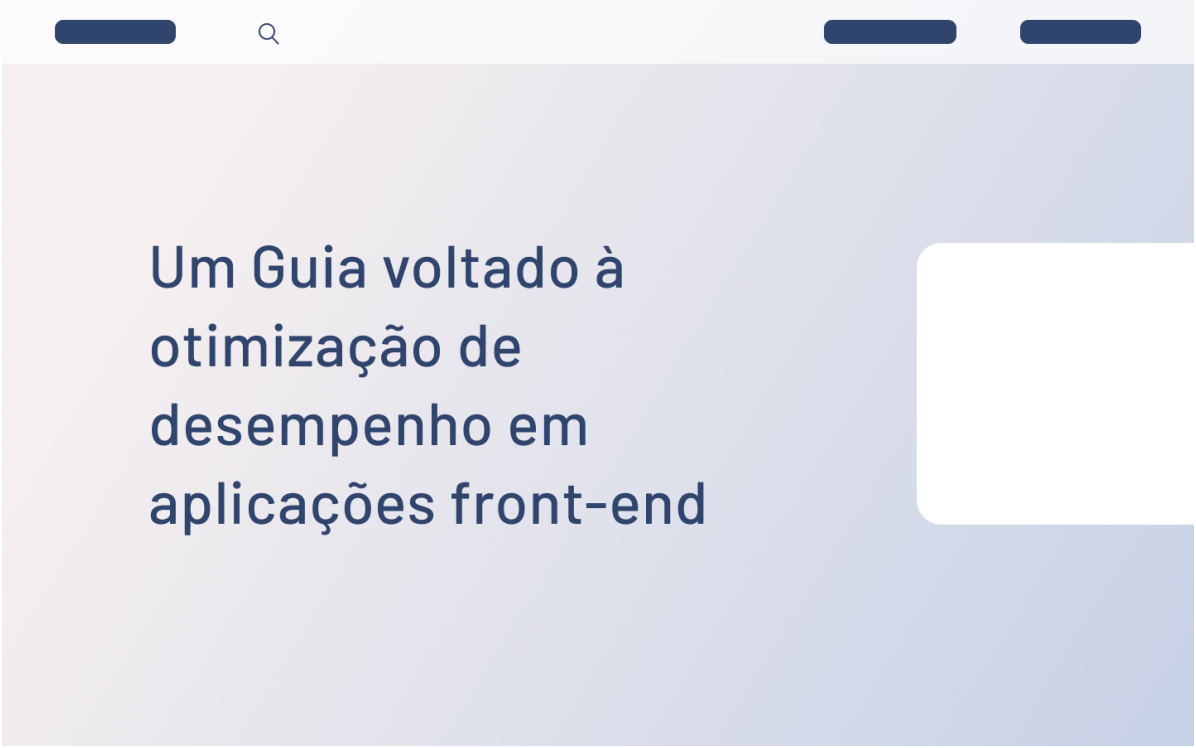
5.2.1.1 Composição final

A *landing page* do Guia representa a tela de boas-vindas, e precisa expor ao usuário a proposta da página em que este se encontra. Desse modo, o título utiliza fonte em

negrito e com tamanho exagerado. A parte ao lado do título, representada pela caixa branca, conterá uma figura que remeta ao contexto da página, ainda sem definição.

A aplicação precisa ser simples e intuitiva. Por isso, a *navbar* foi desenhada para conter uma barra de busca, que pode ajudar o usuário a encontrar informações mais rapidamente. A Figura 12 exibe o protótipo da *landing page*.

Figura 12 – Protótipo de média-alta fidelidade da *landing page*



Fonte: Autora.

Após o contato inicial, ao rolar a página, o usuário é apresentado às seções que compõem o guia. Essas seções estão dispostas em forma de lista de cartões, além de possuírem uma breve descrição. A Figura 13 apresenta a lista de tópicos do guia.

Ao selecionar um dos cartões disponíveis na lista, o usuário é direcionado ao conteúdo em texto. O texto ganha destaque na página, a fim de não causar distrações na leitura. A Figura 14 apresenta a lista de tópicos do guia.

5.2.2 Seções do Guia

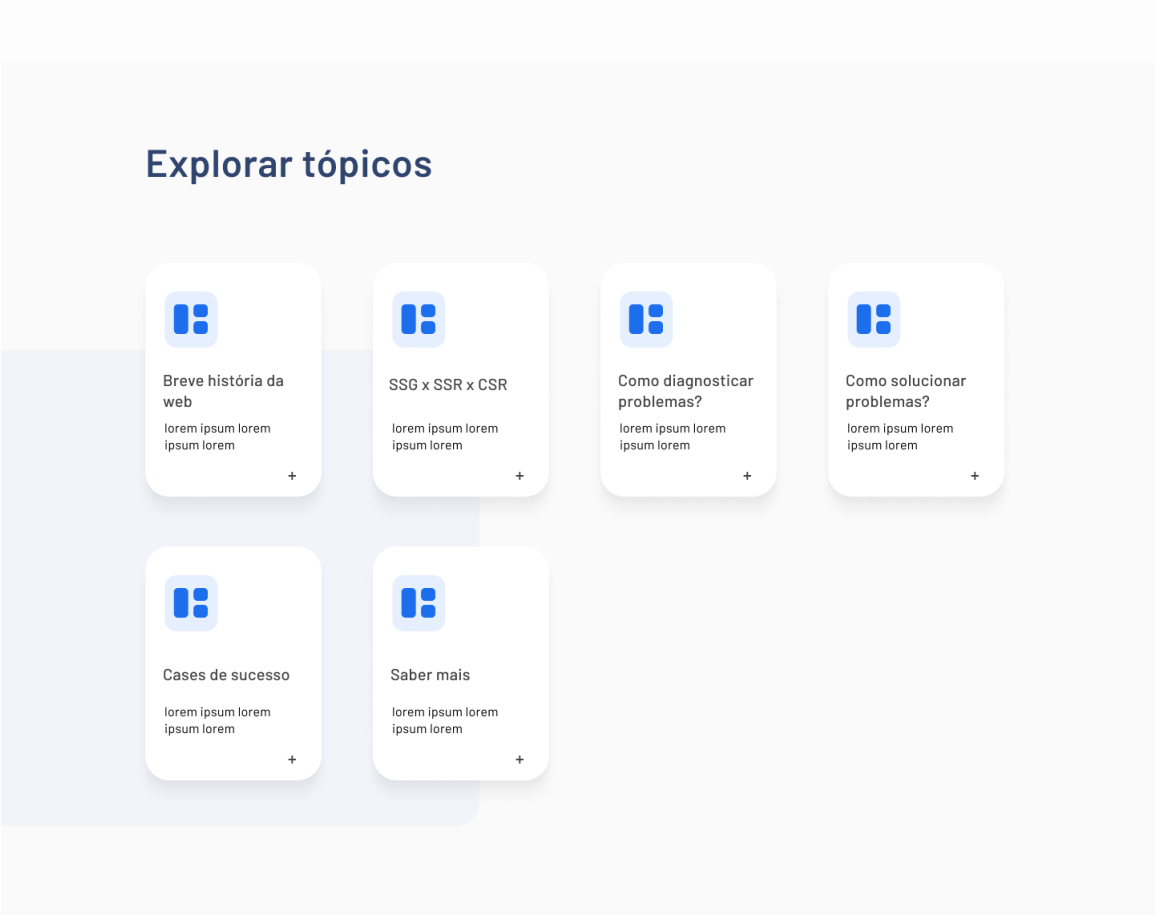
- **Breve história do desenvolvimento *web***

  Esta seção apresenta, de forma sucinta, o surgimento da *web*, bem como o surgimento do HTML, CSS e do navegador através de uma linha do tempo semelhante à apresentada em *The Web's Timeline* (HOFFMAN, 2017).

- **Renderização do lado do cliente, do lado do servidor e geração estática**

  Esta seção explicita a renderização do lado do cliente, do lado do servidor e a

Figura 13 – Protótipo de média-alta fidelidade da lista de tópicos do Guia



Fonte: Autora.

geração estática de conteúdo. Também contém instruções acerca de quando utilizar cada uma dessas abordagens em projetos *front-end*, bem como as tecnologias e *frameworks* disponíveis para cada abordagem.

- **Maiores problemas enfrentados atualmente**

  Seção que expõe, utilizando as fontes que compõem as referências desta pesquisa, os maiores problemas causados pela lacuna deixada pelo desempenho e otimização de uma aplicação *web front-end*.

  Está subdividida em duas partes, **diagnóstico** e **resolução**.

  – **Diagnosticando problemas de desempenho**

    Uma das seções mais importantes do guia. Aqui, serão listadas as ferramentas e *websites* capazes de executar análises em páginas *web*, como o *Google Lighthouse*[1] e o *Webpage Test*[2], bem como as métricas, o que são, e como interpretá-las através de referências como o *Web Vitals*.

[1] https://developers.google.com/web/tools/lighthouse. Acessado pela última vez em 03/05/2021.
[2] https://webpagetest.org. Acessado pela última vez em 03/05/2021.

Figura 14 – Protótipo de média-alta fidelidade do conteúdo em detalhes

Fonte: Autora.

  - **Solucionando problemas de desempenho**

    Seção que apresenta e detalha as técnicas capazes de otimizar desempenho, como por exemplo, compressão de imagens e redução do *JavaScript*.

- **Casos de sucesso**

  Seção que lista exemplos de organizações que aplicaram técnicas e obtiveram sucesso nos tempos de carregamento de suas aplicações *front-end*. Esta seção contém referências de casos de sucesso documentados na iniciativa *Web Vitals*, do Google.

- **Saber mais**

  Esta seção listará todas as referências utilizadas neste trabalho e na escrita dos textos do guia, com os créditos aos autores originais. Também contém um guia de contribuição para que outros usuários/leitores possam ajudar a evoluí-lo por ser uma aplicação de código aberto.

É importante destacar que o Guia contém detalhamento teórico e conceitual, e também prático e aplicado. Portanto, serão conferidos exemplos implementados para ilustrar soluções e boas práticas diante de problemas recorrentes em *front-end*, e que impactam no desempenho.

A ideia é auxiliar de forma mais direta os desenvolvedores, não apenas mencionando que determinados aspectos influenciam, por exemplo, no tempo de carregamento, como adicionalmente demonstrando em termos aplicados como atuar de forma concreta

para mitigar esses aspectos. Essas demonstrações utilizam como base as métricas apresentadas na Seção 2.2. Espera-se que esse embasamento permita apresentar de forma clara os reais ganhos/acertos em termos de otimização de desempenho.

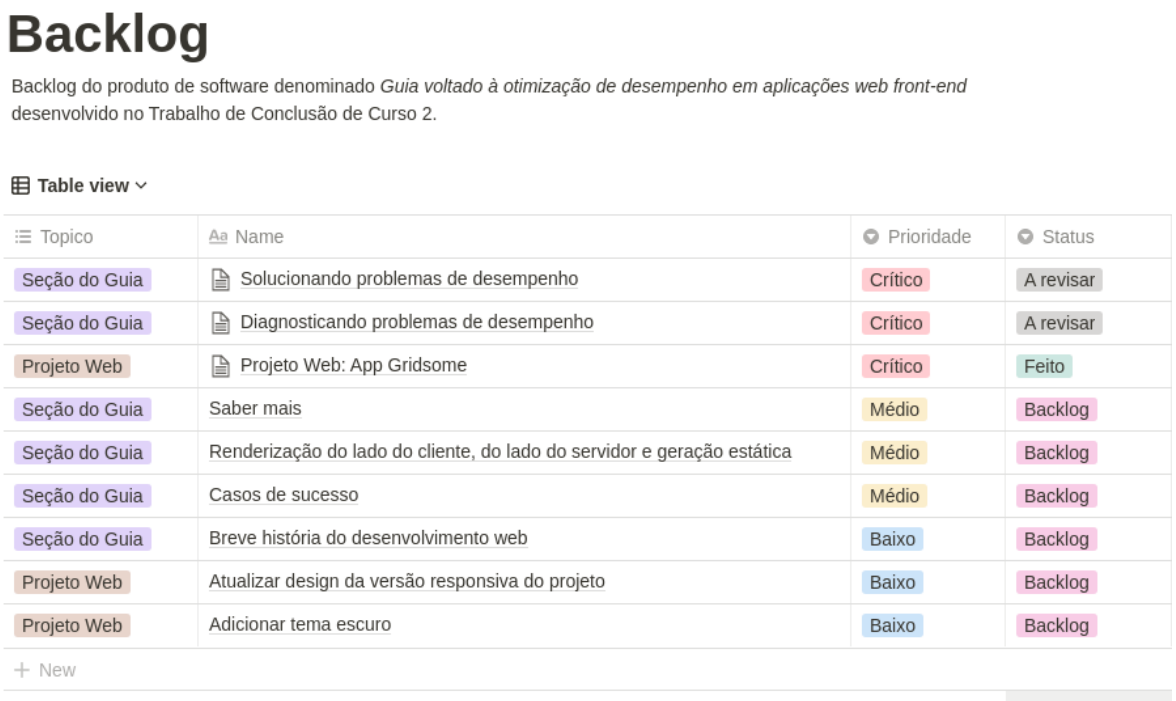
5.3 Detalhamento de requisitos

As Figuras 15 e 16 apresentam o *Backlog* do produto. Os requisitos foram separados em dois épicos, sendo eles:

- **Projeto Web**: lista os requisitos necessários para a construção da aplicação *Gridsome*. São de cunho técnico como adição do mecanismo de busca, finalização de itens de *layout* como o *design* dos componentes de listagem de artigos, e navegação e modo responsivo para a visão de leitura, além de possíveis pontos de evolução, como modo escuro para leitura mais confortável.

- **Seção do Guia**: lista os principais requisitos mapeados que devem conter as seções de leitura do Guia, como diagnóstico que explicita as principais métricas e ferramentas que podem auxiliar os desenvolvedores, além das principais soluções utilizadas atualmente para otimizar as aplicações *web*.

Figura 15 – *Backlog* do Produto: visão simples em formato tabela



Backlog

Backlog do produto de software denominado *Guia voltado à otimização de desempenho em aplicações web front-end* desenvolvido no Trabalho de Conclusão de Curso 2.

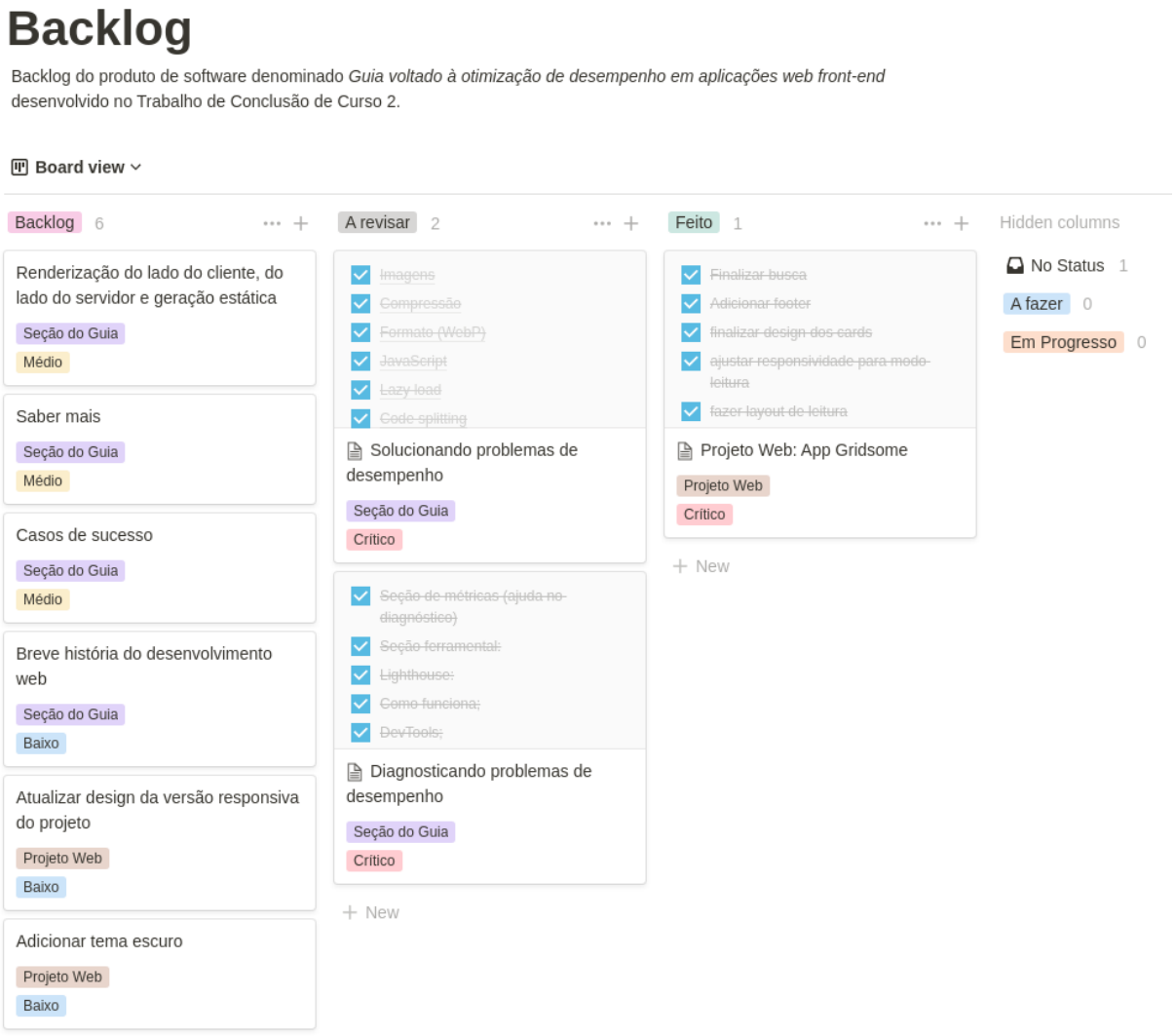
Table view

Topico	Name	Prioridade	Status
Seção do Guia	Solucionando problemas de desempenho	Crítico	A revisar
Seção do Guia	Diagnosticando problemas de desempenho	Crítico	A revisar
Projeto Web	Projeto Web: App Gridsome	Crítico	Feito
Seção do Guia	Saber mais	Médio	Backlog
Seção do Guia	Renderização do lado do cliente, do lado do servidor e geração estática	Médio	Backlog
Seção do Guia	Casos de sucesso	Médio	Backlog
Seção do Guia	Breve história do desenvolvimento web	Baixo	Backlog
Projeto Web	Atualizar design da versão responsiva do projeto	Baixo	Backlog
Projeto Web	Adicionar tema escuro	Baixo	Backlog

+ New

Fonte: Autora.

Figura 16 – *Backlog* do Produto: visão formato *Kanban*



Fonte: Autora.

5.4 O Guia: versão final

Esta seção apresenta a versão final do Guia, seu repositório pode ser encontrado neste endereço: https://github.com/MarianaPicolo/guia-otimizacao.

As informações que compõem os textos do Guia foram curadas conforme as experiências vividas pela autora, que desenvolve aplicações para o *front-end*, além de ter participado do desenvolvimento, diagnóstico e otimização de uma aplicação com problemas de desempenho. Isso permitiu que os textos fossem elaborados segundo o seguinte critério: no máximo três fontes por tópico. A ideia era criar um referencial objetivo através do cruzamento de informações de fontes atualizadas do universo *front-end*, utilizando artigos de autores ativos na comunidade, como os citados no Capítulo de Referencial Teórico.

A Figura 17 apresenta a versão final do Guia disponível ao público, e compreende

o *design* estabelecido na seção 5.2 deste capítulo. A busca, fator importante estabelecido no *backlog*, foi implementada, e encontra os resultados através de busca por termos chave que podem conter nos textos, como por exemplo a palavra 'métrica'. Também é possível navegar através da *navbar* que leva aos textos localizados abaixo do *hero* exibido.

A Figura 18 apresenta a visão de leitura com um dos artigos como exemplo. É possível navegar de forma rápida entre os tópicos de cada artigo através do menu lateral. Também é possível acompanhar as referências utilizadas como base em cada parágrafo através de sua numeração correspondente nas referências ao final dos artigos.

5.4.1 Artigos disponíveis

A versão final do Guia contém os seguintes artigos:

- **Métricas**: inicialmente priorizado no *backlog* como parte de 'Diagnosticando problemas de desempenho', foi fragmentado em um novo artigo. Isso porque facilita a leitura e cria uma sub-classificação no tópico de diagnóstico que pode ajudar o leitor a buscar sobre métricas mais rapidamente.

  O artigo aborda as principais métricas disponíveis atualmente, além de apresentar os últimos resultados reais obtidos pelo *Chrome User Experience Report (CrUX)*[3] através das análises destes dados providas pelo Web Almanac (HTTP ARCHIVE, 2021);

- **Ferramentas de diagnóstico**: também inicialmente priorizado como parte da seção 'Diagnosticando problemas de desempenho', resultou em um novo artigo, desta vez focado no âmbito prático, onde apresenta as ferramentas que permitem diagnosticar problemas em aplicações *web*, além de introduzir pequenos tutoriais das mesmas para incentivar o desenvolvedor a utilizar as ferramentas em seus próprios projetos;

- **Otimizando sua aplicação**: priorizado como parte de 'Solucionando problemas de desempenho', apresenta as principais técnicas que podem ser utilizadas para reduzir o tempo de carregamento de aplicações *web*. Tem foco nas técnicas mais simples, que podem ser adotadas sem consumir muito tempo e não exigem refatorações no código da aplicação como um todo, sendo elas:

  - **Imagens**: apresenta os modos de compressão, além de ferramentas que podem ajudar os desenvolvedores a executar este processo. Além disso, apresenta o novo formato de imagens para a *web*, o WebP, que possui tamanhos de imagem menores que os formatos utilizados atualmente, além dos cuidados a se tomar ao

[3] https://developers.google.com/web/tools/chrome-user-experience-report. Acessado pela última vez em 03/05/2021.

utilizar este formato, como por exemplo como oferecer suporte aos navegadores que ainda não são capazes de exibir imagens em WebP, e

  - **Javascript**: apresenta as técnicas de *Lazy Loading*, ou carregamento preguiçoso, *Code Splitting* e *Tree-shaking*, que podem reduzir o tamanho dos arquivos enviados ao navegador, além de possibilitar redução no código *JavaScript* das aplicações.

- **Exemplo de caso real**: inicialmente priorizado no *backlog* como parte de 'Casos de sucesso', foi fragmentado, e deu origem a um novo artigo. Isso porque esta seção foi planejada para exibir casos de sucesso de organizações documentadas pelo *Web Vitals*. Entretanto, também exibe um caso real de otimização conduzido pela autora do presente trabalho, que utilizou algumas das práticas de otimização para refatorar a aplicação citada na seção 5.1 deste Capítulo.

5.5 Considerações finais do capítulo

Este Capítulo apresentou, em detalhes, a aplicação *web* concebida neste Trabalho de Conclusão de Curso, denominada Guia. A aplicação é constituída de uma série de textos voltados à otimização de desempenho em aplicações *front-end*, a partir da motivação inicial vivida e descrita pela autora.

Adicionalmente, foram apresentadas as seções que compõem o Guia, sendo elas: breve história do desenvolvimento web; renderização do lado do cliente, do lado do servidor e geração estática; maiores problemas enfrentados atualmente; casos de sucesso, e saber mais.

Por fim, como mostrado em detalhes no Capítulo de Suporte Tecnológico, a aplicação foi construída utilizando tecnologias mais emergentes, e que ajudam a otimizar desempenho. Dentre essas tecnologias, destaca-se o SSG, que gera páginas estáticas em tempo de *build* (JACQUART, 2020).

Figura 17 – Visão completa da página inicial do Guia

Fonte: Autora.

Figura 18 – Visão de um artigo do Guia

Fonte: Autora.

6 Resultados obtidos

Este capitulo apresenta os resultados obtidos ao logo dos ciclos de pesquisa-ação conduzidos neste Trabalho de Conclusão de Curso e compreende os processos de coleta, análise e interpretação dos dados, além dos planos de ação tomados para melhoria e evolução do Guia, juntamente com os resultados obtidos. Por fim, tem-se um breve resumo deste capítulo.

6.1 Atividades da pesquisa-ação

Como definido na seção 4.5 do Capítulo de Metodologia, a pesquisa-ação aplicada neste trabalho possui as seguintes atividades: Coleta de dados, Análise e interpretação dos dados, Elaboração do plano de ação e Divulgação dos resultados.

Na fase Coleta de dados, são coletadas impressões de outros desenvolvedores *front-end* via questionário, que são utilizadas como insumo para a próxima atividade. Na fase Análise e interpretação dos dados, os dados coletados são analisados de forma quantitativa e qualitativa. Em Elaboração do plano de ação, é realizado um planejamento com objetivo de solucionar possíveis falhas encontradas. Por fim, na fase de Divulgação dos resultados, são divulgados os resultados obtidos após a execução do plano de ação previamente definido dentro do ciclo.

6.2 Primeiro Ciclo - Consulta à comunidade

Este ciclo tem como objetivo validar o conteúdo apresentado no Guia junto à outros desenvolvedores inseridos no domínio *front-end*, conhecendo o ambiente de trabalho em que estão inseridos, experiências que tiveram com problemas de desempenho e impressões iniciais ao ler o conteúdo disponível. O público que respondeu o questionário era composto de desenvolvedores *front-end*, com anos de carreira variáveis, garantindo assim, uma amostra diversa que pudesse avaliar o Guia.

Para a coleta de dados foi definido e aplicado um questionário, disponível no Apêndice A.

6.2.1 Coleta de dados, análise e interpretação

Esta seção introduz as perguntas e respostas do questionário aplicado, onde cada pergunta apresenta uma figura com os resultados, além de sua respectiva análise e interpretação.

Primeira pergunta do questionário, “Há quanto tempo você desenvolve software?”, feita para conhecer e categorizar o público alvo, dado que os respondentes possuíam tempos de carreira diferentes dentro da área de desenvolvimento de *software*, onde o intuito do Guia é fornecer uma base de conhecimento para desenvolvedores em diferentes fases de carreira. As respostas estão representadas na Figura 19. Percebe-se que a maioria dos respondentes possui de 1 a 3 anos de experiência. Mas, a amostra ainda conta com um representante mais experiente, e um representante mais novato, com menos de um ano de experiência.

Figura 19 – Pergunta 01 do questionário

Fonte: Autora.

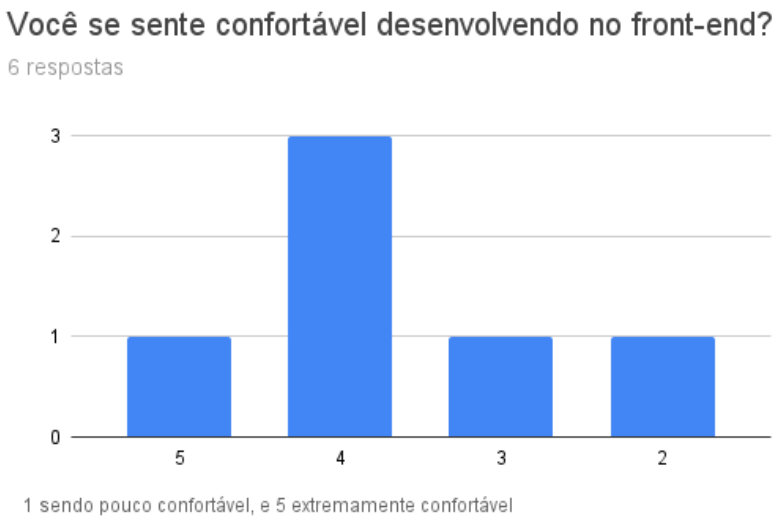
Segunda pergunta do questionário, pergunta de escala baseada na escala de Likert, “Você se sente confortável desenvolvendo no *front-end*?”, também com intuito de conhecer o público alvo, dado que o Guia pode servir como referência sobre otimização de aplicações *front-end* para desenvolvedores iniciantes, além de servir como base para eventuais consultas para desenvolvedores com mais conhecimento e tempo de carreira.

É possível notar que mais da metade dos respondentes se sente de alguma forma confortável, com a maioria das respostas variando entre níveis os 3 a 5. As respostas estão representadas na Figura 20.

Terceira pergunta do questionário, focada no dia a dia do desenvolvedor, e também de escala Likert, “Desempenho no *front-end* é um tópico importante no seu dia a dia de trabalho?”, com o objetivo de conhecer seu ambiente de trabalho, visto que autores como (RUSSEL, 2017) reforçam a importância da criação de uma cultura de desenvolvimento que também se preocupe com o desempenho das aplicações.

É possível notar que 60% das respostas apontam para algum tipo de preocupação com o desempenho das aplicações, o que sugere que os desenvolvedores que participaram

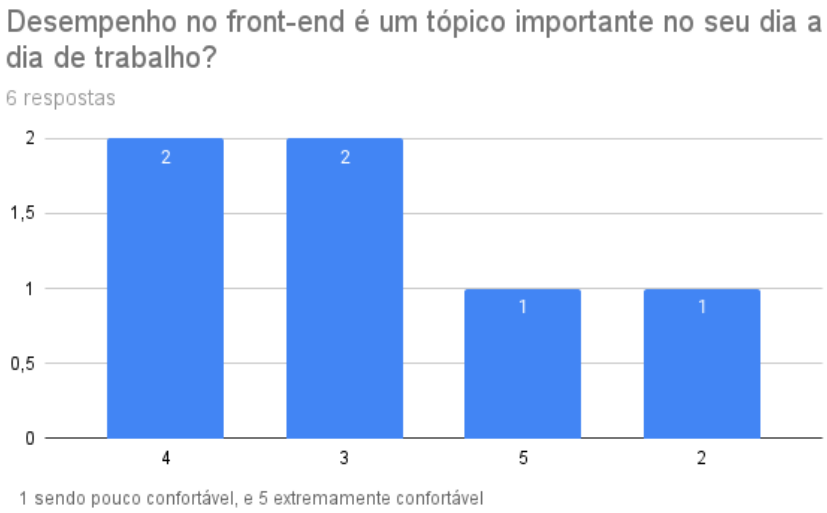
Figura 20 – Pergunta 02 do questionário



Fonte: Autora.

da pesquisa têm algum tipo de conhecimento prévio acerca do tópico. As respostas estão representadas na Figura 21.

Figura 21 – Pergunta 03 do questionário



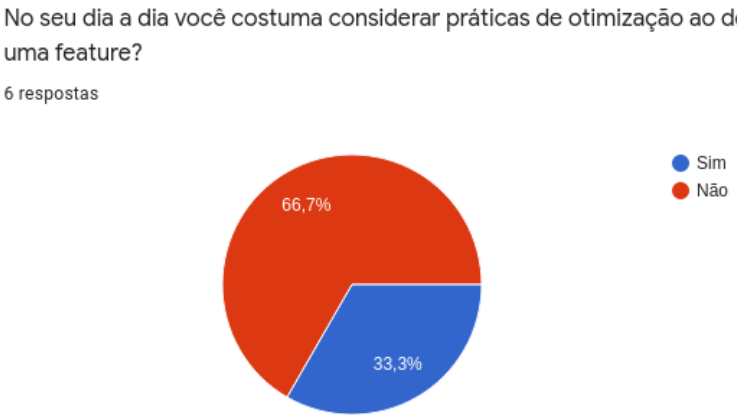
Fonte: Autora.

Quarta pergunta do questionário, focada no modo como o desenvolvedor aborda um requisito a ser transformado em uma possível funcionalidade e se existe uma preocupação com desempenho antes e durante o desenvolvimento, “No seu dia a dia, você costuma considerar práticas de otimização ao desenvolver e entregar uma *feature*?”.

É possível notar que apesar de existir algum nível de preocupação com desempenho, como apontado na questão anterior, mais da metade das respostas apontam que o tópico não é considerado pelos desenvolvedores em tempo de construção de uma *feature*.

As respostas estão representadas na Figura 22.

Figura 22 – Pergunta 04 do questionário



Fonte: Autora.

Quinta pergunta do questionário, primeira dissertativa da sequência, funciona como complemento à pergunta anterior, "Caso você tenha respondido 'sim,' na questão anterior, compartilhe quais técnicas e/ou práticas costuma utilizar.".

As respostas estão representadas na Figura 23, com destaque para duas respostas detalhadas entre dois perfis contrastantes. A segunda resposta da lista pertence a um profissional com mais experiência no desenvolvimento *front-end*, e o último item de um desenvolvedor com menos experiência. Ressaltam-se menções às práticas de monitoramento, como o acompanhamento das métricas geradas pelo *Lighthouse* como também o tamanho da aplicação final que será enviada ao navegador do usuário, através do tamanho do *bundle*.

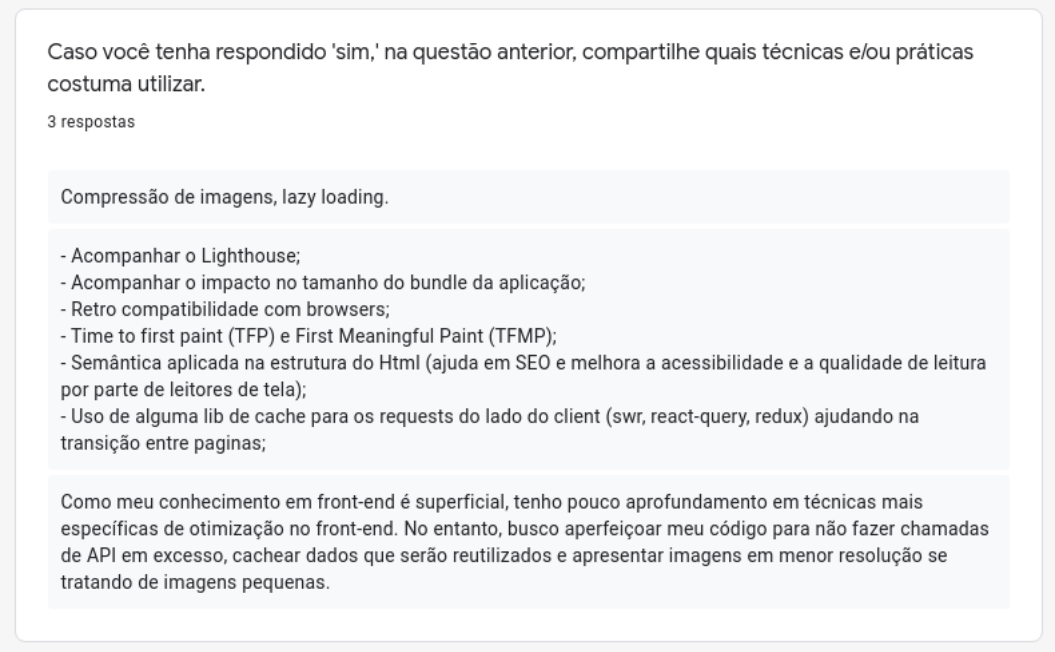
Sexta pergunta do questionário, segunda pergunta dissertativa, agora buscando indícios de uma visão voltada ao diagnóstico por parte dos desenvolvedores, "Você já percebeu problemas relativos a desempenho no *front-end* de alguma aplicação que contribuiu/trabalhou? Se sim, compartilhe uma breve descrição do problema.".

As respostas estão representadas na Figura 24. Cabe colocar que a maioria dos respondentes apresentou alguma observação acerca de lentidão, espera demorada e outros fatores que sugerem problemas de desempenho.

Sétima pergunta do questionário, ainda voltada ao diagnóstico, procura conhecer o nível geral de familiaridade dos desenvolvedores a respeito do uso das ferramentas que possibilitam monitorar desempenho, "Você já utilizou ferramentas de diagnóstico (como *lighthouse* ou *webpagetest*) para apurar, de forma geral, o desempenho no *front-end* de alguma aplicação que contribuiu/trabalhou?".

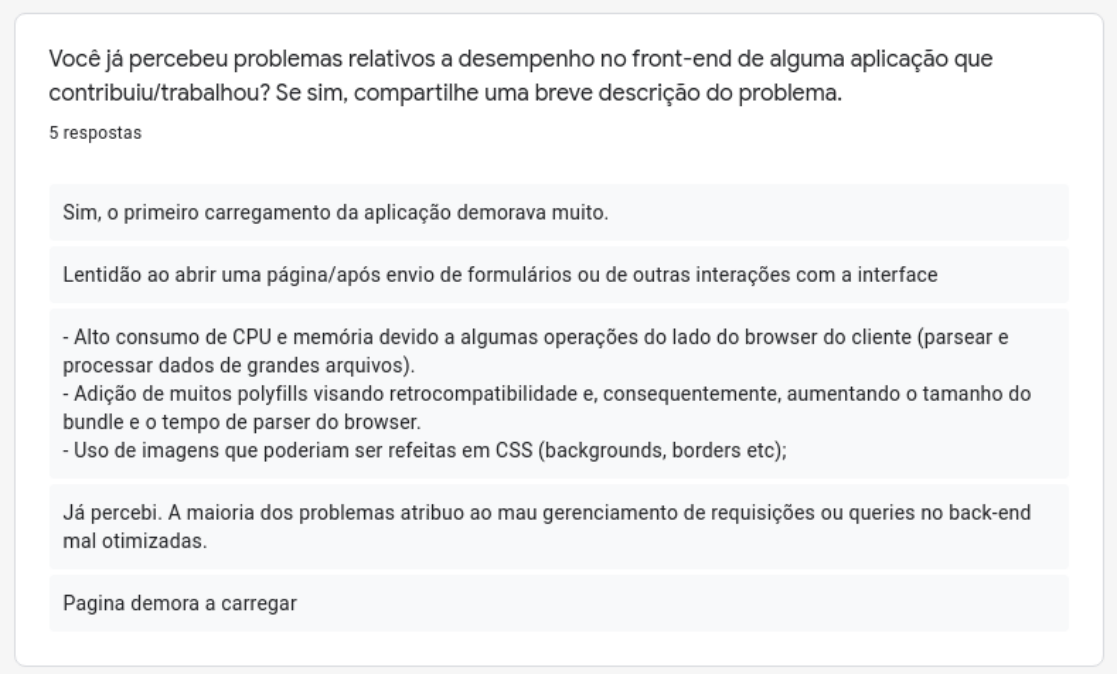
É possível notar que 60% das respostas apontam para uma preocupação com desempenho, pois realizam algum tipo de monitoramento de suas aplicações através do uso

Figura 23 – Pergunta 05 do questionário



Fonte: Autora.

Figura 24 – Pergunta 06 do questionário

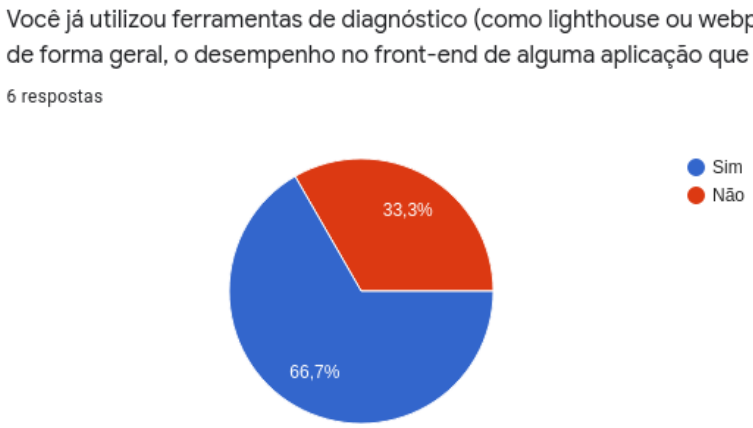


Fonte: Autora.

das ferramentas de diagnóstico. É importante ressaltar que mais da metade das respostas não consideraram desempenho durante o desenvolvimento de uma *feature*, como apontado nos resultados da questão 4. As respostas estão representadas na Figura 25.

Oitava pergunta do questionário, procura conhecer as ferramentas utilizadas pelos desenvolvedores que tenham respondido ‘sim’ na pergunta anterior, “Caso você tenha

Figura 25 – Pergunta 07 do questionário

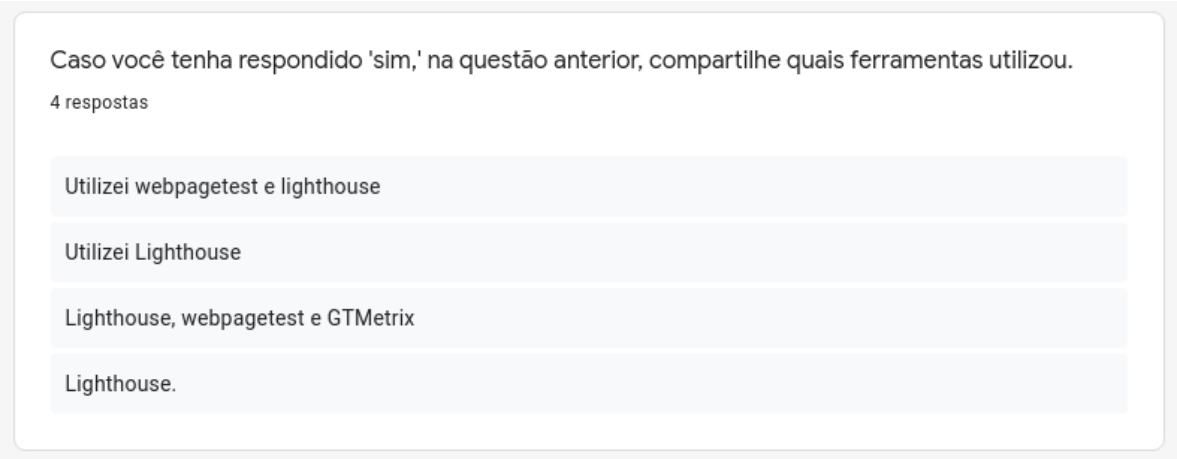


Fonte: Autora.

respondido 'sim,' na questão anterior, compartilhe quais ferramentas utilizou.".

As respostas estão representadas na Figura 26. Dentre as ferramentas mencionadas, destacam-se duas apresentadas pela seção de monitoramento do Guia: o *Lighthouse* e o *WebPageTest*.

Figura 26 – Pergunta 08 do questionário



Fonte: Autora.

6.2.1.1 Impressões gerais

Para a coleta de *feedbacks* gerais a respeito do Guia, foram disponibilizadas duas perguntas de cunho dissertativo para os respondentes, e suas respostas estão representadas na íntegra através das Figuras 27 e 28.

A pergunta 09 permite o compartilhamento de sugestões e para os textos disponíveis no Guia, sob o olhar de outros desenvolvedores *front-end*.

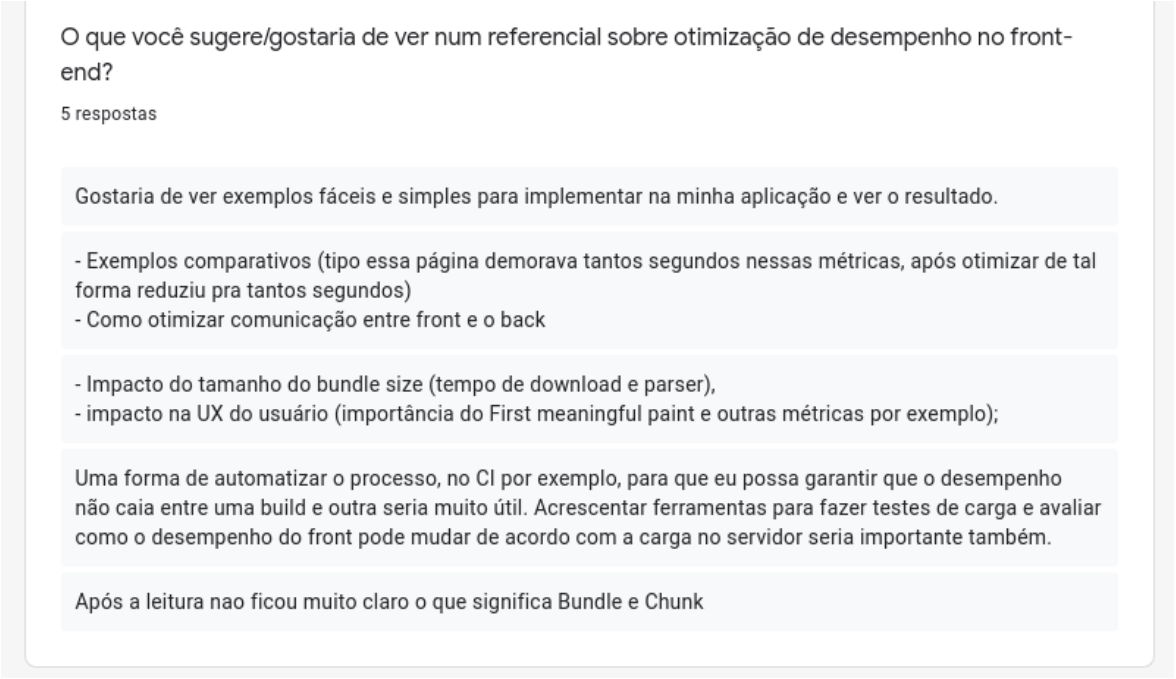
Dentre as sugestões apontadas pelos respondentes, merecem menção:

- "Impacto do tamanho do *bundle size* (tempo de *download* e *parser*) e impacto na UX do usuário (importância do *First meaningful paint* e outras métricas por exemplo)"

- "Uma forma de automatizar o processo, no CI por exemplo, para que eu possa garantir que o desempenho não caia entre uma *build* e outra seria muito útil. Acrescentar ferramentas para fazer testes de carga e avaliar como o desempenho do *front* pode mudar de acordo com a carga no servidor seria importante também"

A última pergunta está direcionada a *feedbacks* gerais após a leitura, além de buscar conhecer os pontos fortes do Guia também sob a perspectiva de outros desenvolvedores. É possível perceber que boa parte dos respondentes gostou da seção de métricas, seção que, de acordo com um dos mesmos "fornece uma excelente base para começar a avaliar um projeto e construir melhorias".

Figura 27 – Pergunta 09 do questionário



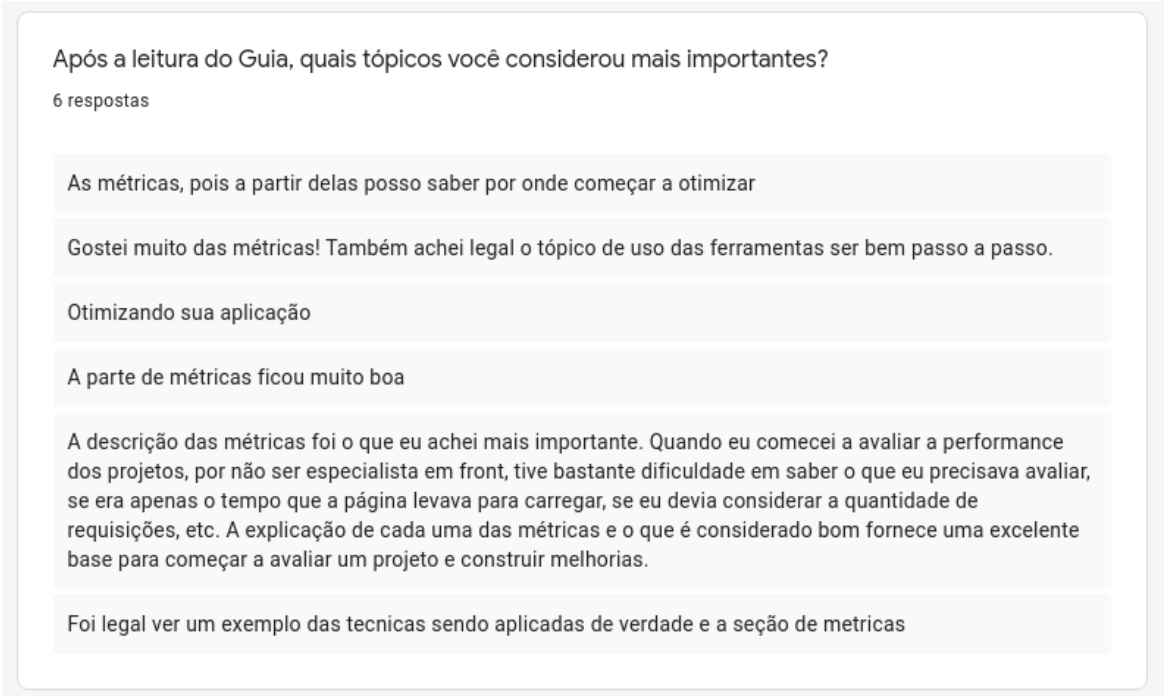
Fonte: Autora.

6.2.2 Plano de ação

A partir dos pontos coletados após a aplicação do questionário, juntamente com as análises disponibilizadas na seção anterior, foi possível levantar alguns pontos de melhoria e evolução para o Guia:

- Criação de um glossário para ajudar desenvolvedores inciantes a compreenderem termos chave que fazem parte do universo *front-end*, e

Figura 28 – Pergunta 10 do questionário



Fonte: Autora.

- Apresentar uma ferramenta na seção de diagnóstico que possa auxiliar o monitoramento do tamanho do *bundle* das aplicações.

6.2.3 Divulgação dos resultados

Após a implementação das melhorias divulgadas, conforme estabelecidas no plano de ação, têm-se as seguintes mudanças no Guia:

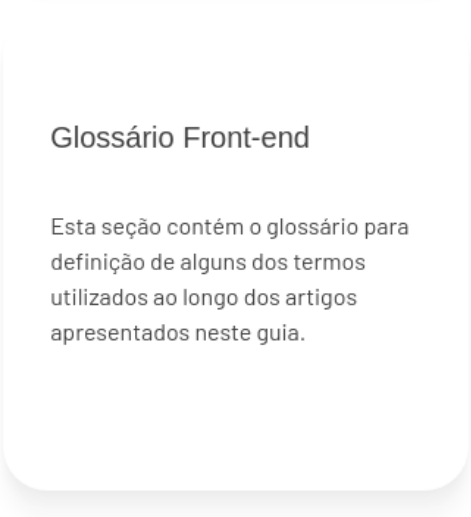
- Uma nova seção, que apresenta um glossário com alguns dos principais termos utilizados acerca de desempenho *front-end*:

- Um novo item, na seção 'Ferramentas de diagnóstico', que ajuda os desenvolvedores a monitorar o tamanho do *bundle* de suas aplicações que são entregues aos navegadores dos usuários.

6.3 Segundo Ciclo - Inspeção

O segundo ciclo de pesquisa ação foi realizado pela autora, com base nos resultados e *feedbacks* coletados no primeiro ciclo, bem como considerando que a mesma é desenvolvedora atuante na área de *front-end*, procurou realizar uma autocrítica sobre a versão "final" do Guia, com o objetivo de melhorar a experiência de leitura.

Figura 29 – Glossário



Fonte: Autora.

6.3.1 Análise e plano de ação

A autora conduziu uma inspeção no Guia após o encerramento do primeiro ciclo com o intuito de buscar e documentar possíveis falhas e ou pontos de melhoria ao longo da aplicação. Os pontos de melhoria notados estão relacionados à experiencia de leitura, dado que alguns conceitos podem ser mais facilmente compreendidos caso acompanhem exemplos reais:

- Disponibilizar um exemplo concreto de *Cumulative Layout Shift* dentro da aplicação. Um exemplo ilustrando a ocorrência do CLS pode facilitar a compreensão do leitor, dado que o CLS acontece quando o conteúdo que deveria ocupar determinado espaço na janela, por ainda não ter carregado completamente e não estar visível, é sobreposto pelo conteúdo já carregado abaixo dele. Entretanto, quando o conteúdo original é finalmente carregado, o mesmo acaba "saltando" em tela;

- Disponibilizar um exemplo de imagem no formato original sem compressão ao lado de sua versão com compressão num formato mais moderno e otimizado para a *web*, o WebP.

  Um exemplo que mostre lado a lado a diferença de qualidade entre ambas versões da mesma imagem pode incentivar outros desenvolvedores a submeter as imagens exibidas em seus *websites* a um processo de compressão, favorecendo o carregamento mais rápido das imagens, e

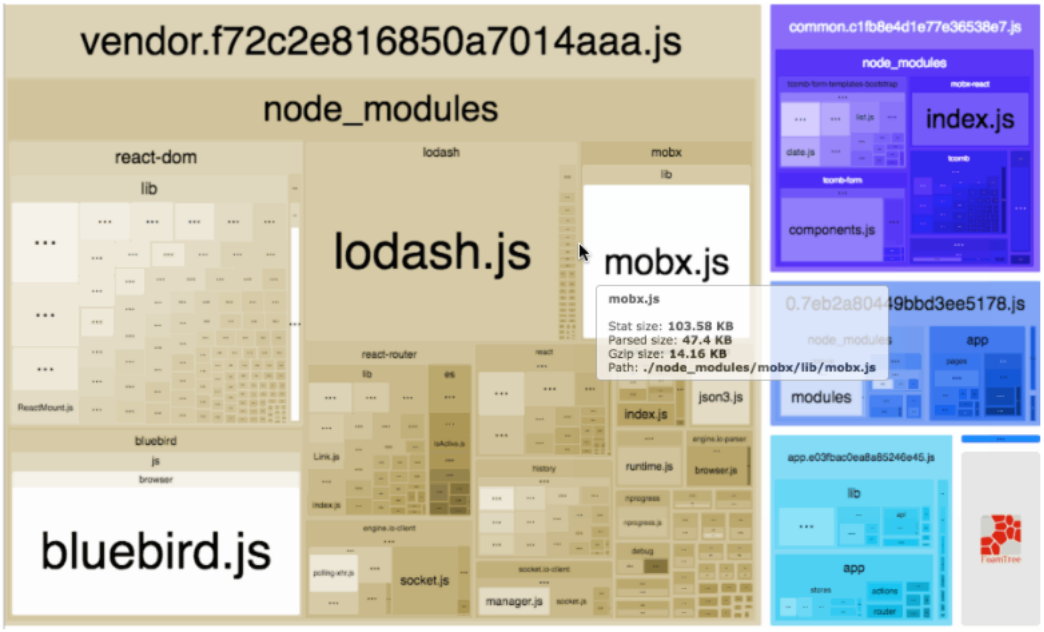
- Corrigir o *layout* de leitura em telas menores, pois o texto está ultrapassando, além da tela, causando rolagem horizontal, o que pode prejudicar a leitura dos textos do Guia em *smartphones*. A falha está representada na Figura 31:

Figura 30 – Subseção em ferramentas

Monitorando o tamanho da sua aplicação

Existem algumas ferramentas que podem auxiliar a monitorar o tamanho da aplicação, facilitando o diagnóstico das partes que mais entregam arquivos Javascript, por exemplo.

Ferramentas como Webpack Bundle Analyzer, Rollup Plugin Visualizer e parcel-plugin-bundle-visualiser geram um mapa interativo que exibe o tamanho dos arquivos contidos no *bundle*.



Fonte: Webpack Bundle Analyzer

Fonte: Autora.

6.3.2 Divulgação dos resultados

Após o diagnóstico e a definição das melhorias a serem implementadas no Guia, têm se as seguintes mudanças:

- Uma nova página, representada na Figura 32, que pode ser acessada através da seção de "Métricas", na parte de *Cumulative Layout Shift*, onde é possível ver um exemplo real do *layout* se deslocando na porção visível da janela.
- Também apresentada em uma nova página, representada na Figura 33, pode ser acessada através da seção "Otimizando sua aplicação", na parte de "imagens". É possível ver um exemplo real com duas imagens lado a lado, uma delas no formato tradicional e a outra no formato otimizado para a *web*.
- A Figura 34 apresenta a correção do *layout* de leitura para telas menores, facilitando o acesso de dispositivos móveis ao Guia.

Figura 31 – Texto incompleto por falha no *layout*

Fonte: Autora.

Figura 32 – *Live example* de CLS

Fonte: Autora.

Figura 33 – *Live example* que exibe imagens nos formatos .jpg e .webp

Fonte: Autora.

Figura 34 – Atualização de *layout* de leitura para *smartphones*

Fonte: Autora.

6.4 Visão Aplicada

O Guia contém exemplos práticos sobre o uso de algumas técnicas e abordagens, visando mitigar determinados problemas associados a desempenho em *front-end*. Nesse contexto, a autora procurou apoiar-se em práticas e estudos realizados em um caso real, realizado pela própria autora. O estudo pode ser acessado neste endereço.

6.4.1 Contexto

Como descrito na seção Contextualização no Capítulo 5, a autora do presente trabalho atuou no desenvolvimento do *front-end* de uma *startup* do ramo imobiliário. A ideia de refatorar a aplicação surgiu no momento em que o tempo necessário para carregar completamente uma página estava atingindo a marca de quase 20 segundos em computadores, que começou a causar desconforto até mesmo em alguns clientes, que relataram, via suporte técnico, sentir problemas de lentidão.

À vista disso, a autora conduziu um processo de refatoração que foi documentado em inglês e disponibilizado para a comunidade, onde relatou a experiência desde as medidas de diagnóstico, até as soluções empregadas para melhorar o desempenho da aplicação.

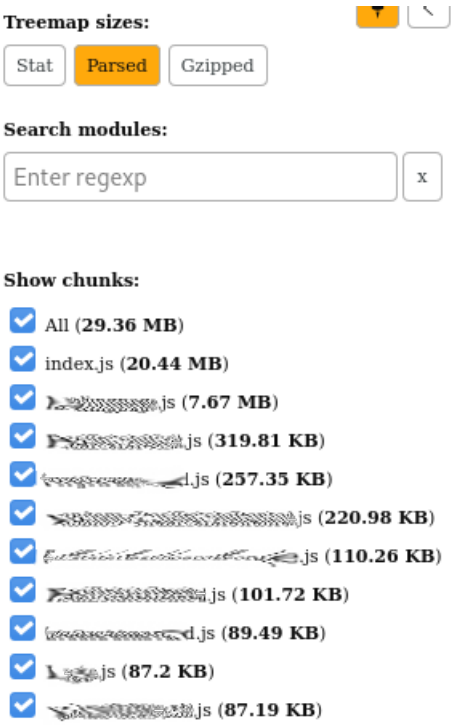
6.4.2 Problema

A aplicação cresceu muito rapidamente, e como era um MVP, por vezes de forma desordenada e sem boas práticas da engenharia de *software*, como a definição de um modelo arquitetural que possibilitasse escalabilidade, ou até mesmo a execução de rituais de *code review*.

A fim de executar um diagnóstico que pudesse dar visibilidade a respeito dos aspectos internos da aplicação que era desenvolvida, foi gerado um relatório capaz de exibir seu tamanho, o *bundle*, em detalhes, e foi possível perceber que o tamanho da aplicação era um problema: quase 30 MB de arquivos (entre *JavaScript*, HTML e CSS), eram enviados ao navegador do usuário. Isso evidenciou um problema já notado pela autora enquanto desenvolvedora: havia muito código *JavaScript* na aplicação, além de um alto número de código duplicado.

Adicionalmente, foi notado que a aplicação entregava todos os arquivos necessários para sua execução em no máximo 10 arquivos diferentes, o que evidenciava o não uso de técnicas como *lazy-loading*, técnica que consiste no *download* dos recursos *JavaScript*, HTML e CSS pelo navegador apenas quando uma página é acessada pelo usuário, ou seja, os arquivos são entregues sob demanda, economizando banda e memória (WAGNER; ANDREW, 2019). A Figura 35 apresenta o tamanho do *bundle* antes da refatoração.

Figura 35 – Tamanho do *bundle* pré refatoração

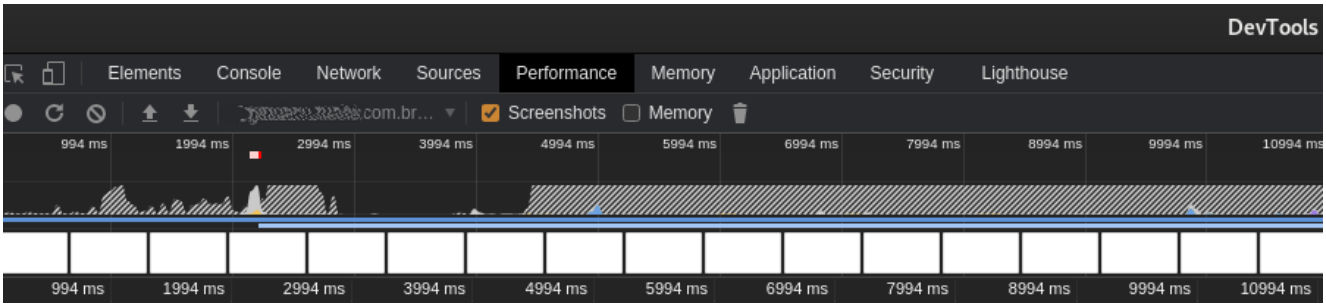


Fonte: Autora.

A Figura 36 representa a linha do tempo do processo de renderização da *landing page* da aplicação antes de qualquer processo de refatoração ser aplicado. A linha do tempo foi extraída da ferramenta *Developer Tools*, disponível no navegador do *Google*, o *Chrome*.

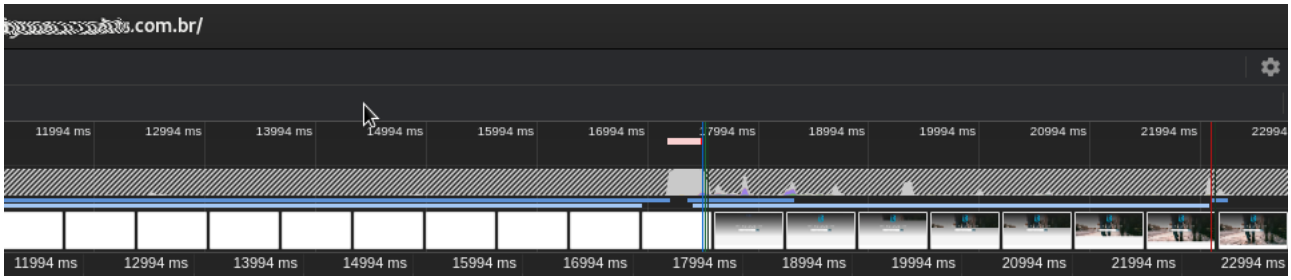
Nota-se que a aplicação, por entregar muitos arquivos, permanece na maioria dos *frames* exibindo uma página em branco. A Figura 37 representa a parte final do *frame*, onde também é possível observar o tempo gasto para o carregamento total da página: quase 23 segundos no navegador *Google Chrome*, versão *desktop*.

Figura 36 – Processo de renderização ilustrado através de *frames*, parte inicial



Fonte: Autora.

Figura 37 – Final do processo de renderização ilustrado através de *frames*

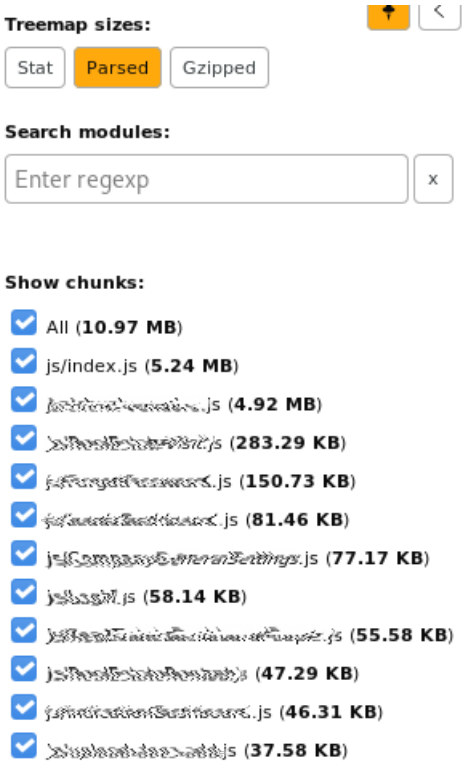


Fonte: Autora.

6.4.3 Solução

Com o objetivo de diminuir o tamanho e a quantidade de arquivos enviados ao cliente, parte das bibliotecas *JavaScript* instaladas por outros desenvolvedores, que eram responsáveis por manipular datas ou validar formulários, por exemplo, foram removidas da aplicação ou foram ou substituídas por alternativas menores. Muitos arquivos também não eram removidos ao não serem mais necessários em alguma parte da aplicação. Entretanto, ainda que não exibidos em tela, continuavam sendo baixados pelos navegadores dos usuários (PICOLO, 2020). A Figura 38 apresenta o tamanho do *bundle* após este processo de limpeza de código, onde é possível notar significativa redução para quase 11 MB.

Figura 38 – Tamanho da aplicação após primeiro ciclo de melhorias



Fonte: Autora.

Num segundo ciclo de melhorias, foi implementado *lazy loading*, ou carregamento preguiçoso, a nível de rotas da aplicação. Isso significa que apenas os arquivos necessários

para a construção da página em que o usuário está navegando ou solicitou serão baixados, o que economiza recursos do navegador, além de banda do usuário. Também foi implementado *tree-shaking* nas importações de funções de bibliotecas instaladas, o que ajuda a reduzir o tamanho do *bundle* da aplicação, pois não é mais necessário empacotar a biblioteca inteira, com todas as suas funções, apenas a função importada para utilização (PICOLO, 2020).

Além disso, as imagens exibidas na aplicação passaram por um processo de compressão com algum nível de perda (lossy) e conversão para WebP, o que reduziu seus tamanhos de 3 MB, em média, para 200 KB (PICOLO, 2020).

Como mencionado na seção 6.4.2 deste Capítulo, a aplicação não possuía *lazy loading* configurado. Por isso todos os arquivos necessários para a sua exibição completa, ainda que não utilizados, eram baixados e executados no navegador do usuário. Após configurado com sucesso, foi possível notar um aumento significativo dos *chunks* (ou pedaços) que compunham a aplicação, saltando de 10 antes da refatoração, para mais de 100 pedaços diferentes, que apresentavam tamanhos menores, como apresentado na Figura 39.

Após conduzir novo teste para apurar o tempo de renderização de uma páginas das áreas de clientes, nota-se que os resultados finais foram satisfatórios, como mostrado nas Figuras 40 e 41, onde o tempo necessário para renderizar a página por completo foi de 4.24 segundos no navegador *Google Chrome*, versão *desktop*.

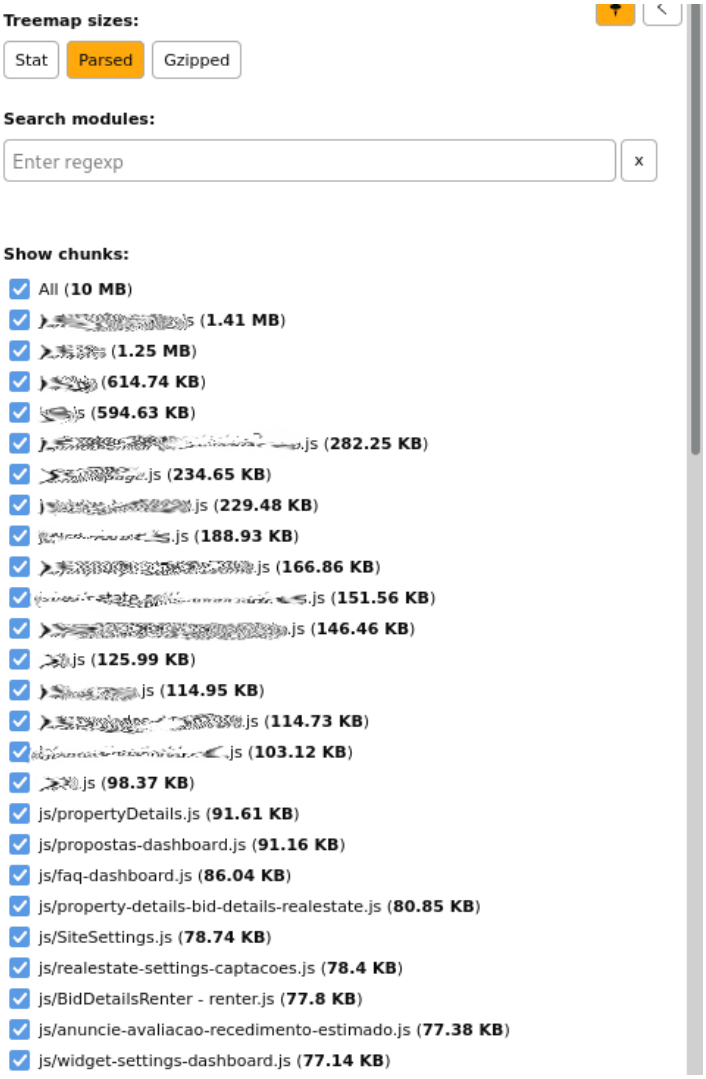
É importante ressaltar que este processo foi documentado e disponibilizado em formato de *blog* (PICOLO, 2020), pois poderia ajudar a comunidade a refletir sobre a importância das boas práticas da engenharia de *software* ao se construir uma aplicação *front-end*, além de incentivar a participação dos leitores. Foi possível, inclusive, perceber o quão importante é o uso de textos mais concisos e diretos, com exemplos práticos, pois conquista o público alvo e confere ao trabalho maior relevância.

6.5 Considerações finais do capítulo

Este Capítulo apresentou os resultados obtidos ao longo dos dois ciclos de pesquisa ação realizados. O primeiro ciclo contou com a participação de outros desenvolvedores para avaliar e validar o conteúdo do Guia, e o segundo ciclo proporcionou uma série de melhorias, principalmente focadas na experiência de leitura para facilitar o entendimento dos conceitos apresentados para o leitor.

O Capítulo traz ainda uma visão mais aplicada dos conceitos apresentados no Guia, através da experiência relatada pela autora do presente trabalho num artigo escrito e disponibilizado à comunidade, com textos e ações simples que podem ser tomadas a qualquer momento e que podem fazer a diferença no uso das aplicações para o usuário

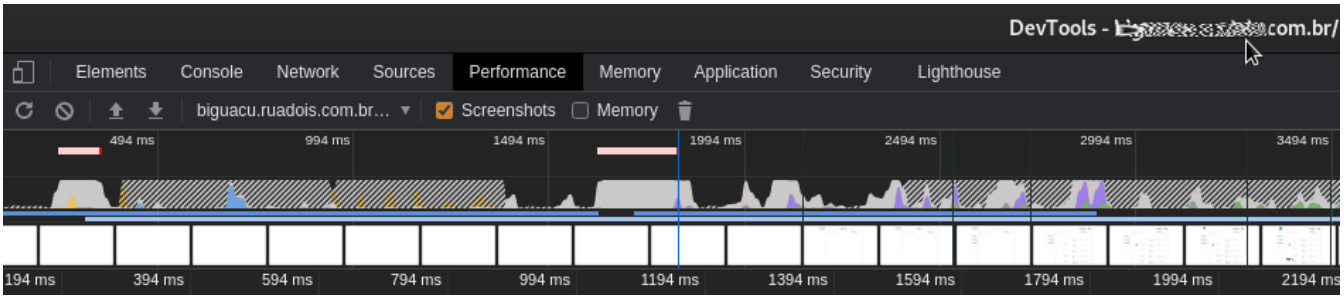
Figura 39 – Tamanho da aplicação após segundo ciclo de melhorias



Fonte: Autora.

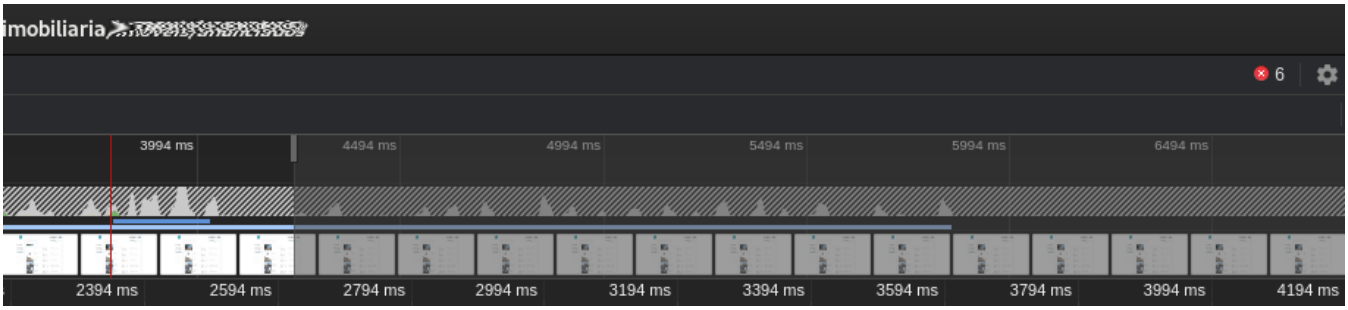
final.

Figura 40 – *Frame* do processo de renderização após segundo ciclo de melhorias



Fonte: Autora.

Figura 41 – Parte final do *frame* do processo de renderização após segundo ciclo de melhorias



Fonte: Autora.

7 Conclusão

Este capítulo apresenta as considerações finais a respeito deste Trabalho de Conclusão de Curso, onde, primeiramente, serão retomados os objetivos definidos no Capítulo de Introdução, com intuito de acordar o atingimento ou não dos mesmos para em seguida abordar suas competências e fragilidades. Por último, são apresentados os trabalhos futuros que podem vir a complementar e evoluir o Guia.

7.1 Objetivos alcançados

- Identificação das principais métricas relativas ao desempenho no *front-end*;

  Tal objetivo foi alcançado ainda na primeira parte deste trabalho, estando presente na seção 2.2 no Capítulo de Referencial Teórico.

- Identificação das principais técnicas relativas à otimização de desempenho no *front-end*;

  Tal objetivo foi atingido e validado nos ciclos de pesquisa ação, com as principais técnicas documentadas no Guia;

- Desenvolvimento de uma plataforma de código aberto, que confira uma base, se possível de referência, para os desenvolvedores brasileiros consultarem e melhorarem os indicadores de desempenho de suas aplicações.

  Tal objetivo foi atingido e validado também nos ciclos de pesquisa ação através da participação de desenvolvedores *front-end* que atuaram como respondentes do questionário disponibilizado no primeiro ciclo.

7.2 Competências do Guia

O Guia fornece uma base teórica que pode ser consultada por desenvolvedores *front-end* e pela comunidade em geral, a fim de conhecer e entender as principais técnicas além de poder conhecer como as métricas funcionam para avaliar uma aplicação em termos de desempenho. Também é possível acompanhar exemplos práticos que podem incentivar os desenvolvedores a dar o primeiro passo numa cultura de desenvolvimento que se preocupe com o desempenho das aplicações.

7.3 Fragilidades do Guia

Fragilidades evidenciadas ao longo dos ciclos de pesquisa ação por outros desenvolvedores envolvem a extensão do conteúdo que pode ser disponibilizado, como a integração entre as camadas *back-end* e *front-end*, desenvolvimento orientado à compatibilidade entre os navegadores existentes, e apresentação de outras ferramentas que possibilitem um monitoramento mais amplo, e por fim, documentação de casos de UI e UX focados em desempenho.

7.4 Trabalhos futuros

Para trabalhos futuros e possíveis contribuições da comunidade, visto que o Guia foi concebido para ser aberto e evoluído, podem ser destacados os seguintes pontos de evolução:

- Reunir e documentar os casos de sucesso que envolvem refatoração de aplicações de grande porte voltadas à redução do tempo de carregamento;
- Estender o número de técnicas do Guia, através da adição de tópicos acerca de CSS e Fontes, por exemplo, e
- Apresentar mais ferramentas que possibilitem um diagnóstico cada vez mais completo e preciso para os desenvolvedores.

Referências

ALQUDAH, M.; RAZALI, R. A comparison of scrum and kanban for identifying their selection factors. In: *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*. [S.l.: s.n.], 2017. p. 1–6. Citado 2 vezes nas páginas 44 e 46.

AN, D. *Find out how you stack up to new industry benchmarks for mobile page speed. "Think with Google-Mobile, Data & Measurement"*, 2018. Citado na página 24.

ART THERAPY. Color psychology: The emotional effects of colors. 2020. Disponível em: <http://www.arttherapyblog.com/online/color-psychology-psychologica-effects-of-colors>. Acesso em: 04 dez. 2020. Citado na página 49.

CHACON, S.; STRAUB, B. *Pro Git*. [S.l.]: apress, 2014. Citado na página 38.

DJIRDEH, H.; OSMANI, A.; BYNENS, M. *Browser-level image lazy-loading for the web*. 2020. Disponível em: <https://web.dev/browser-level-image-lazy-loading/>. Acesso em: 07 mar. 2021. Citado na página 35.

ERICSSON. The stress of streaming delays. Estocolmo, Suécia, p. 12, 2016. Citado na página 23.

FRIEDMAN, V. *Front-End Performance Checklist*. 2020. Disponível em: <https://www.smashingmagazine.com/2021/01/front-end-performance-2021-free-pdf-checklist/>. Acesso em: 3 dez. 2020. Citado 2 vezes nas páginas 35 e 48.

GARSIEL, T.; IRISH, P. *How browsers work: Behind the scenes of modern web browsers. Google Project, August*, 2011. Disponível em: <https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/>. Acesso em: 14 set. 2020. Citado 6 vezes nas páginas 27, 28, 29, 30, 31 e 35.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de pesquisa*. [S.l.]: Plageder, 2009. Citado 2 vezes nas páginas 39 e 40.

GIL, A. C. *Como elaborar projetos de pesquisa*. 6a edição. ed. [S.l.]: Editora Atlas, 2017. ISBN 978-85-97-01292-7. Citado 4 vezes nas páginas 40, 42, 43 e 45.

GOOGLE. Time to interactive. 2019. Disponível em: <https://web.dev/interactive/>. Acesso em: 13 out. 2020. Citado na página 34.

GOOGLE. The science behind web vitals. 2020. Disponível em: <https://blog.chromium.org/2020/05/the-science-behind-web-vitals.html>. Acesso em: 15 out. 2020. Citado 2 vezes nas páginas 31 e 35.

GOOGLE. Web vitals. 2020. Disponível em: <https://web.dev/vitals/>. Acesso em: 03 dez. 2020. Citado 2 vezes nas páginas 36 e 48.

GRIGORIK, I. *Optimize WebFont loading and rendering*. 2019. Disponível em: <https://web.dev/optimize-webfont-loading/>. Acesso em: 04 set. 2020. Citado na página 50.

GROSSKURTH, A.; GODFREY, M. W. *Architecture and evolution of the modern web browser. Preprint submitted to Elsevier Science*, v. 12, n. 26, p. 235–246, 2006. Citado 2 vezes nas páginas 27 e 28.

HOFFMAN, J. *The History of the Web*. 2017. Disponível em: <https://thehistoryoftheweb.com/timeline/>. Acesso em: 1 dez. 2020. Citado na página 51.

HOFFMANN, J. *The Decade-Long Path to Web Fonts*. 2017. Disponível em: <https://thehistoryoftheweb.com/web-fonts/>. Acesso em: 4 dez. 2020. Citado na página 50.

HTTP ARCHIVE. *Report: State of the Web*. 2020. Disponível em: <https://beta.httparchive.org/reports/state-of-the-web#bytesTotal>. Acesso em: 03 set. 2020. Citado na página 23.

HTTP ARCHIVE. *Web Almanac: HTTP Archive's annual state of the web report*. 2021. Disponível em: <https://almanac.httparchive.org/en/2020/>. Acesso em: 30 abr. 2021. Citado na página 56.

HéGARET, P. *What is the Document Object Model?* 2000. Disponível em: <https://www.w3.org/TR/DOM-Level-2-Core/introduction.html>. Acesso em: 29 set. 2020. Citado na página 29.

ILIEV, I.; DIMITROV, G. P. Front end optimization methods and their effect. In: *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. [S.l.: s.n.], 2014. p. 467–473. Citado 4 vezes nas páginas 24, 28, 29 e 43.

IMTIAZ, S. The psychology behind web design. 12 2016. Citado na página 49.

JACQUART, G. *SPA, SSG, SSR and JAMStack*. 2020. Disponível em: <https://levelup.gitconnected.com/spa-ssg-ssr-and-jamstack-a-front-end-acronyms-guide-6add9543f24>. Acesso em: 2 dez. 2020. Citado na página 57.

KOSAKA, M. *Inside look at modern web browser (part 2)*. 2018. Disponível em: <https://developers.google.com/web/updates/2018/09/inside-browser-part2>. Acesso em: 19 set. 2020. Citado 4 vezes nas páginas 28, 29, 30 e 31.

MARIA, J. S. *On web typography*. New York, NY: A Book Apart, 2014. Disponível em: <https://cds.cern.ch/record/1970235>. Citado na página 50.

MIHAJLIJA, M. *Performance budgets 101*. 2018. Disponível em: <https://web.dev/performance-budgets-101/#definition>. Acesso em: 06 set. 2020. Citado 3 vezes nas páginas 23, 24 e 44.

MIHAJLIJA, M. *Extract critical CSS*. 2019. Disponível em: <https://web.dev/extract-critical-css/>. Acesso em: 06 mar. 2021. Citado na página 35.

PAVIC, B.; ANSTEY, C.; WAGNER, J. *Why does speed matter?*: Performance is about user experience. 2019. Disponível em: <https://web.dev/why-speed-matters/#performance-is-about-user-experience>. Acesso em: 03 set. 2020. Citado na página 23.

PICOLO, M. My notes about conducting a massive refactor in a vue.js website. 2020. Disponível em: <https://dev.to/marianapicolo/my-notes-about-conducting-a-massive-refactor-in-a-vue-js-website-bo7>. Acesso em: 09 mai. 2021. Citado 2 vezes nas páginas 75 e 76.

ROCHA, Z. *Como Perder Pes no Browser*. 2013. Disponível em: <https://browserdiet.com/pt/>. Acesso em: 2 dez. 2020. Citado na página 48.

ROCHA, Z. *Why did I create Browser Diet?* 2013. Disponível em: <https://zenorocha.com/browser-diet>. Acesso em: 3 dez. 2020. Citado na página 48.

RUSSEL, A. *Can You Afford It?*: Real-world web performance budgets. 2017. Disponível em: <https://infrequently.org/2017/10/can-you-afford-it-real-world-web-performance-budgets/#content>. Acesso em: 10 set. 2020. Citado 2 vezes nas páginas 23 e 62.

SCHWABER, K. *Agile project management with Scrum*. [S.l.]: Microsoft press, 2004. Citado na página 44.

SHROFF, P. H.; CHAUDHARY, S. R. Critical rendering path optimizations to reduce the web page loading time. In: *2017 2nd International Conference for Convergence in Technology (I2CT)*. [S.l.: s.n.], 2017. p. 937–940. Citado 3 vezes nas páginas 23, 24 e 43.

WAGNER, J.; ANDREW, R. *Use lazy-loading to improve loading speed*. 2019. Disponível em: <https://web.dev/lazy-loading/>. Acesso em: 08 mai. 2021. Citado na página 73.

WALTON, P. *First Contentful Paint*. 2019. Disponível em: <https://web.dev/fcp/>. Acesso em: 09 out. 2020. Citado na página 32.

WALTON, P. *First Input Delay*. 2019. Disponível em: <https://web.dev/fid/>. Acesso em: 12 out. 2020. Citado na página 33.

WALTON, P. *Largest Contentful Paint*. 2019. Disponível em: <https://web.dev/lcp/>. Acesso em: 11 out. 2020. Citado 2 vezes nas páginas 32 e 33.

WALTON, P. *Time to Interactive*. 2019. Disponível em: <https://web.dev/tti/>. Acesso em: 12 out. 2020. Citado 2 vezes nas páginas 33 e 34.

WALTON, P. *Total Blocking Time*. 2019. Disponível em: <https://web.dev/tbt/>. Acesso em: 13 out. 2020. Citado na página 34.

WALTON, P. User-centric performance metrics. 2019. Disponível em: <https://web.dev/user-centric-performance-metrics/#user-centric-performance-metrics>. Acesso em: 11 out. 2020. Citado na página 31.

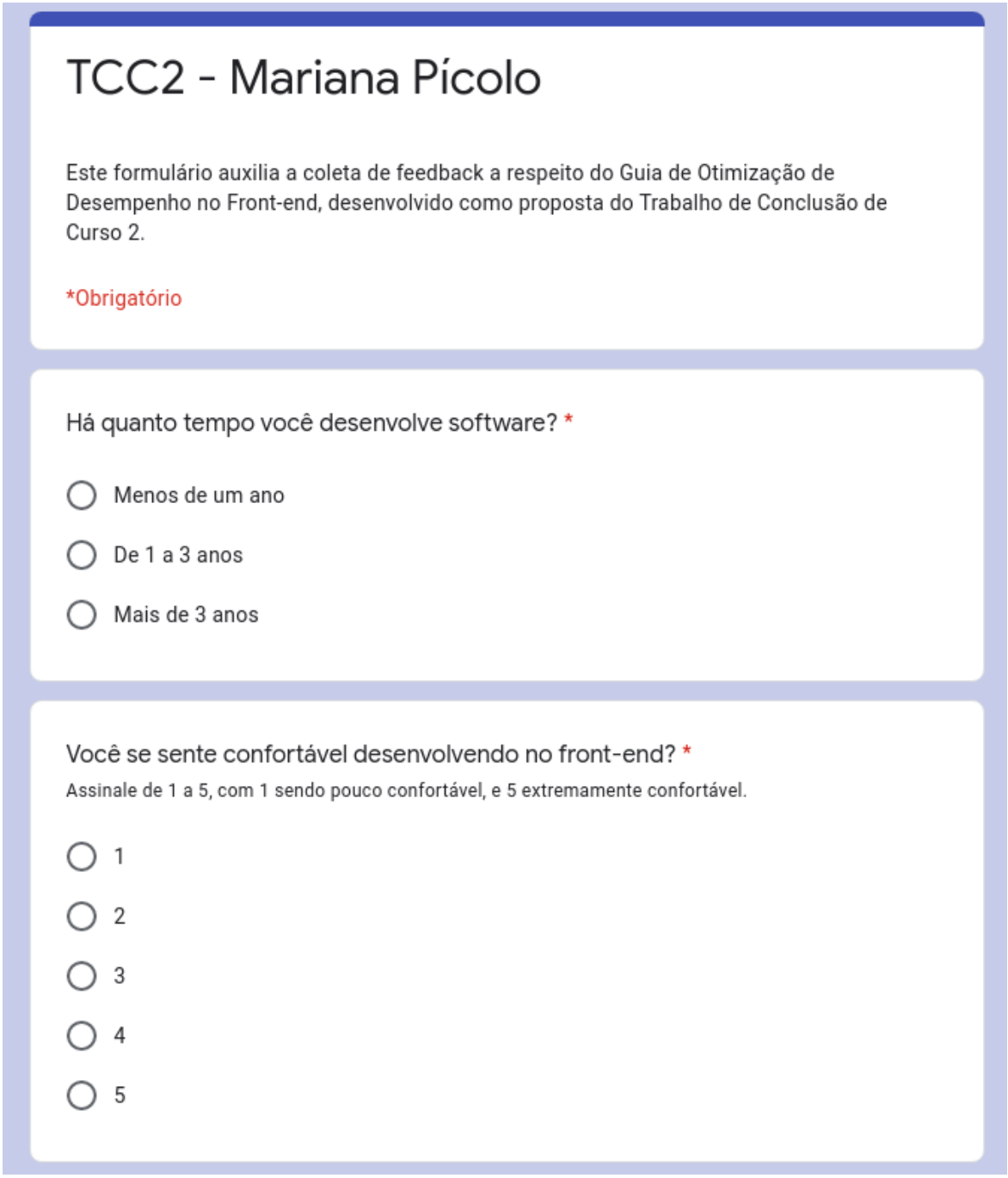
WALTON, P.; MIHAJLIJA, M. *Cumulative Layout Shift*. 2019. Disponível em: <https://web.dev/cls/>. Acesso em: 13 out. 2020. Citado na página 34.

WILLIAMS, L. Agile software development methodologies and practices. In: *Advances in computers*. [S.l.]: Elsevier, 2010. v. 80, p. 1–44. Citado na página 44.

Apêndices

APÊNDICE A – Primeiro questionário de avaliação do Guia

Figura 42 – Parte I do questionário.



Fonte: Autora.

Figura 43 – Parte II do questionário.

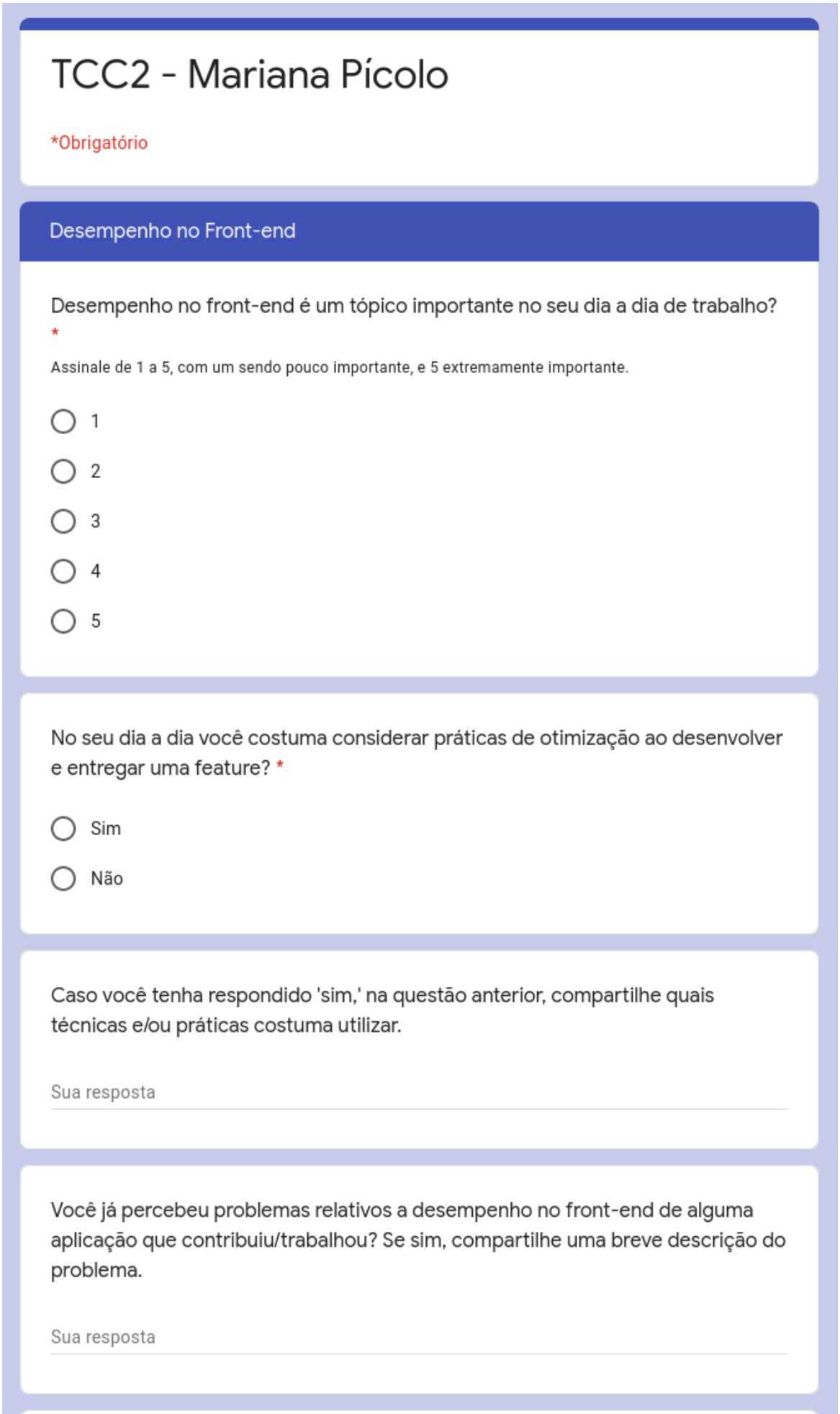
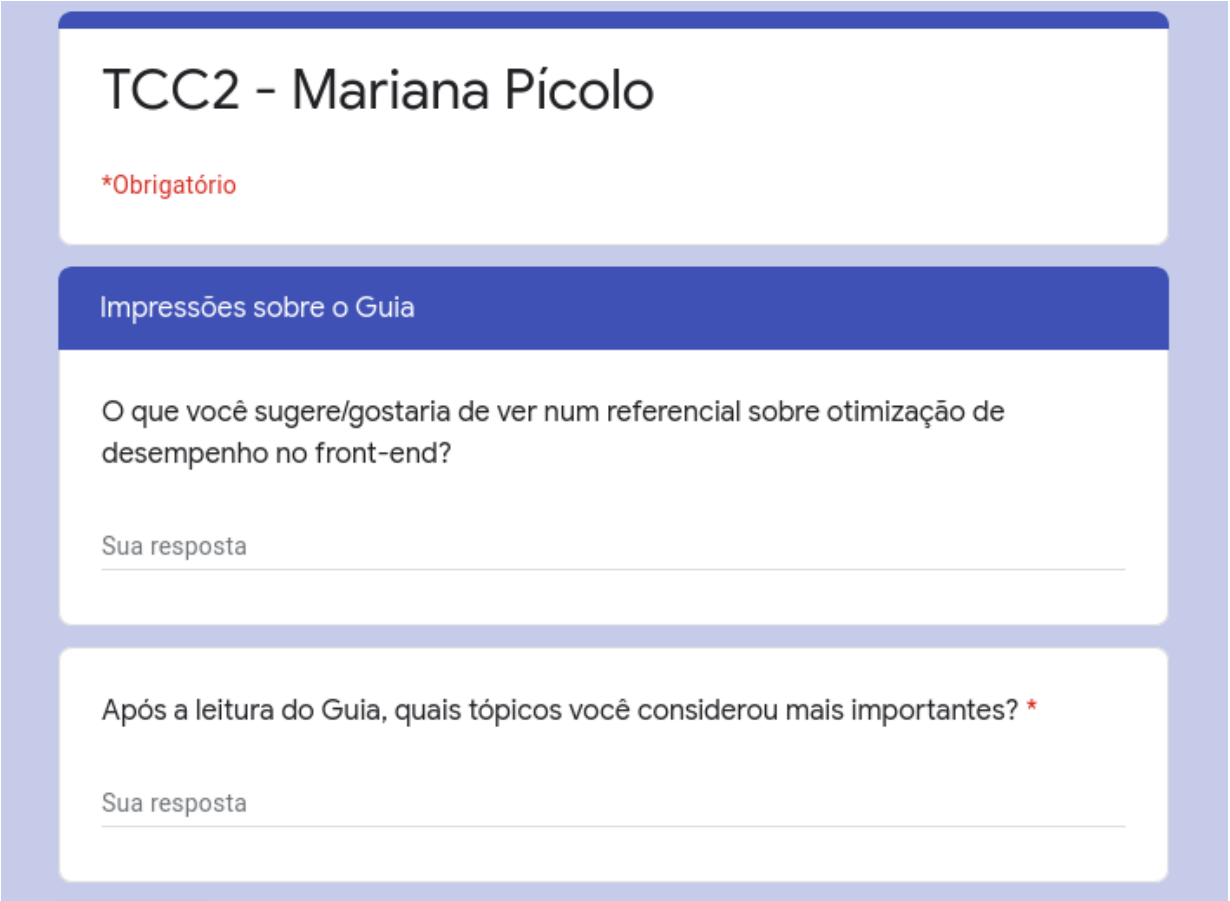


Figura 44 – Parte III do questionário.



Fonte: Autora.