Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Tabi

Extensão de navegador para visualização de abas

Ricardo Geraldes Tolesano

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisora: Profª. Drª. Kelly Rosa Braghetto

São Paulo
2022

A vida vale a pena quando você torce para ela não acabar.

— Clóvis de Barros Filho

Agradecimentos

Agradeço às minhas amigas Natália, Akemi, Bruna, Hei, Alice, Celine, e Cécile, que tornaram a vida muito mais interessante.

Agradeço à minha orientadora Kelly pela ajuda proporcionada neste trabalho.

Resumo

Ricardo Geraldes Tolesano. **Tabi:** ***Extensão de navegador para visualização de abas***. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2022.

Abas são uma funcionalidade comum nos navegadores mais populares atualmente. Inicialmente, este trabalho investiga os diversos problemas do design tradicional de abas e lista uma série de requisitos que uma boa solução deve satisfazer. Então, é realizada uma análise de algumas extensões e soluções existentes, e é feita uma compilação de algumas possibilidades de soluções para os problemas encontrados. Finalmente, o trabalho descreve o desenvolvimento da extensão de navegador Tabi que tenta resolver alguns desses problemas. A extensão permite categorizar abas como "importantes", "para leitura" e "já lidas", e mostra abas relacionadas por domínio e por similaridade de títulos em listas separadas. Permitir categorizar abas e mostrar abas importantes em uma lista própria parece ser útil para a realização de pesquisa exploratória. O modo como a funcionalidade de abas relacionadas foi implementada não rendeu resultados satisfatórios. Limitações na API dos navegadores não permitiram que o design planejado fosse desenvolvido. Diversas possibilidades futuras foram descritas. A extensão possui código aberto e foi publicada para o navegador Firefox.

Palavras-chave: Aba. Navegador. Design. UI. Extensão. Add-on. Pesquisa exploratória.

Abstract

Ricardo Geraldes Tolesano. **Tabi: *Browser extension for visualizing tabs***. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2022.

Tabs are a common feature in most popular browsers today. Initially, this work investigates the various problems of the traditional tab design and lists a series of requirements that a good solution must satisfy. Then, an analysis of some existing extensions and solutions is performed, and a compilation of some possible solutions for the problems is made. Finally, the work describes the development of the Tabi browser extension, which attempts to solve some of these problems. The extension allows one to categorize tabs as "important", "to read" and "already read", and shows related tabs by domain and title similarity in separate lists. Allowing tabs to be categorized and showing important tabs in a list of their own seems to be useful for conducting exploratory research. The way the related tabs functionality was implemented did not yield satisfactory results. Limitations in the browsers' API did not allow the planned design to be developed. Several future possibilities were described. The extension is open source and has been published for the Firefox browser.

Keywords: Tab. Browser. Design. UI. Extension. Add-on. Exploratory search.

Lista de figuras

Lista de tabelas

Sumário

Capítulo 1

Introdução

Abas são uma funcionalidade comum nos navegadores mais populares atualmente. Hahn *et al.* (2018) destacaram que abas servem principalmente duas funções para o usuário. A primeira delas é organizar e alternar entre múltiplas tarefas que o usuário está realizando no momento. Tarefas variam em duração, dificuldade e escopo, e podem ser, por exemplo, realizar um pagamento, resolver algo burocrático, cumprir com obrigações do trabalho, estudar, ouvir música, assistir vídeos, planejar uma viagem, acessar redes sociais, ou realizar uma pesquisa de algum assunto de interesse. Todas essas tarefas podem ocorrer ao mesmo tempo no navegador.

Mehrotra *et al.* (2016) estimaram, com base em dados provenientes de uma ferramenta de busca, que aproximadamente 80% dos usuários realizam duas ou mais tarefas por sessão de navegação. Ao longo de uma ou mais sessões, o usuário pode abrir diversas abas para cada tarefa, alternar entre elas, organizá-las e eventualmente fechá-las. Nesse sentido, pode-se dizer que não concluir tarefas ou não fechar abas após a conclusão das tarefas talvez sejam duas das principais causas do problema de acúmulo de abas, que será discutido posteriormente.

A segunda função desempenhada pelas abas é a de permitir que o usuário construa um modelo mental relacionado com a tarefa que está sendo realizada. Por exemplo, ao planejar uma viagem para outro país, inicialmente não se sabe muito sobre o assunto. Conforme o usuário pesquisa, ele pode comparar fontes de informação, realizar uma triagem de quais sites contêm informações relevantes, entender quais questões são mais relevantes e descobrir o que ainda precisa ser pesquisado antes de viajar.

Entretanto, seu design atual sofre de múltiplos defeitos. De forma geral, os usuários sentem dificuldade e confusão ao lidar com dezenas ou centenas de abas relacionadas a tarefas diferentes. O objetivo deste trabalho é investigar esses problemas e desenvolver uma solução na forma de uma extensão de navegador. A intenção principal não é que a extensão seja uma solução definitiva para os défices do design atual, mas sim explorar um novo design para visualização de abas, e permitir analisar como cada funcionalidade contribui para a resolução dos problemas.

Cada funcionalidade feita para a extensão teve como objetivo solucionar ou amenizar um ou mais problemas discutidos. Sobretudo, a necessidade de organização manual das

abas, e a falta de meios para que o usuário represente adequadamente o modelo mental da tarefa sendo realizada. A extensão permite categorizar abas como “importantes”, “para leitura” e “lidas”, além de exibir abas relacionadas à aba atual. A partir do desenvolvimento da extensão, concluiu-se que as APIs (Application Programming Interface) dos navegadores restringem consideravelmente o design e possíveis funcionalidades de uma extensão. Isso pode ser futuramente aperfeiçoado de forma a permitir desenvolver extensões mais funcionais.

O trabalho está dividido em três partes principais. A primeira parte descreve os problemas relacionados às abas de navegador, investiga suas causas e lista uma série de requisitos que uma solução deve satisfazer. Na segunda parte, é feita uma descrição e análise de alguns projetos que propuseram soluções para os problemas das abas ou que servem de inspiração para a elaboração da extensão. Algumas possíveis soluções são compiladas na Tabela 3.1. Finalmente, na terceira parte, é realizada uma descrição do desenvolvimento da extensão de navegador. É explicada a ideia por trás do projeto, sua estrutura, bem como as bibliotecas e arcabouços utilizados. Uma análise de como a extensão contribui na resolução dos problemas das abas é feita ao final do trabalho.

Capítulo 2

Caracterização do problema

O design tradicional de abas permite que o usuário represente seu modelo mental principalmente pela ordenação das abas. Alguns usuários empregam estratégias específicas de ordenação e disposição das abas no navegador, de forma que as abas melhor reflitam seu modelo mental. Um exemplo de estratégia é manter abas não lidas no final da lista e mover as abas com conteúdo mais relevante para o começo da lista. Além disso, é comum que certas abas sejam mantidas abertas para servir como lembrete para a realização de alguma tarefa, tal como responder um e-mail.

Nem sempre o usuário possui um modelo mental apropriado para a tarefa sendo realizada. Nesse sentido, o modelo mental pode ser construído conforme o usuário entende o domínio em questão. O processo de entender um domínio e construir um modelo mental é chamado de *sensemaking* na literatura e pode ser traduzido como "dar sentido". O tipo de pesquisa pela qual o usuário compreende o assunto e a tarefa e constrói seu modelo mental é chamada *exploratory search*, ou pesquisa exploratória. Pesquisas desse tipo requerem uma síntese e costumam ser mais complexas e envolver mais tentativas em comparação com pesquisas cujo objetivo é encontrar informações fatuais. Além disso, o processo de pesquisa exploratória é acompanhado de incerteza sobre quais informações e fontes serão relevantes no futuro, principalmente nos estágios iniciais de pesquisa (Hahn *et al.*, 2018).

2.1 Críticas ao design atual das abas

Um fenômeno comum é que usuários acumulem dezenas, centenas ou até milhares de abas ao longo de diversas sessões de navegação. Esse fenômeno é chamado de *tab hoarding* e pode ser traduzido como acúmulo de abas. Uma pesquisa feita por Chang, Hahn *et al.* (2021) concluiu que 30% dos participantes consideram ser acumuladores de abas. Uma das consequências desse acúmulo é algumas abas não estarem mais visíveis ou distinguíveis na tela e portanto nunca serem relembradas ou acessadas. O espaço da tela é limitado, e quando a quantidade de abas é grande, alguns navegadores optam por permitir rolagem das abas, enquanto outros tornam as abas cada vez menores, até que elas eventualmente se tornam muito pequenas e indistinguíveis (Figura 2.1). O problema de perder visibilidade das abas é informalmente conhecido como *black hole effect*, ou efeito

buraco negro. Além desse problema, organizar e navegar por uma grande quantidade de abas pode causar uma sobrecarga cognitiva e ser uma experiência frustrante e confusa para o usuário. Esse problema é chamado de *tab overload*, ou sobrecarga de abas (Chang, Kim *et al.*, 2021).

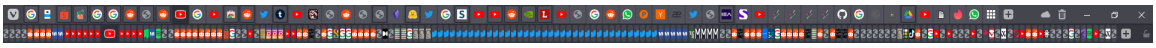

Figura 2.1: *Captura de tela de uma lista com mais de 800 abas abertas no navegador Vivaldi. Cada aba é consideravelmente pequena. Pode ser difícil gerenciar essa quantidade de abas. Fonte: autor.*

Chang, Hahn *et al.* (2021) descreveram duas forças que regem o problema do acúmulo de abas: a pressão para fechar abas e a pressão para manter abas abertas. As pressões para fechar abas dizem respeito a aumentar a organização, atenção e clareza mental, bem como reduzir o estresse e o uso de recursos computacionais tais como memória. Tem-se principalmente dois fatores colaborando para manter abas abertas. O primeiro se refere à necessidade de poder retomar uma tarefa que estava sendo feita sem perder o progresso e organização. O segundo se refere a poder visitar rapidamente sites acessados frequentemente.

Além dos problemas relacionados ao acúmulo de abas, tem-se problemas com o desempenho do design atual ao exercer certas funções. Hahn *et al.* (2018) argumentaram que abas possuírem múltiplos propósitos faz com que o design delas não tenha bom desempenho para nenhum deles. Sobretudo, a interface de abas poderia providenciar um suporte melhor para tarefas complexas tais como pesquisa exploratória. Nesse sentido, são detalhados três problemas no design de abas.

Primeiramente, abas são desconectadas de sua fonte ou "aba progenitora". Novas abas são costumeiramente abertas a partir de uma página de pesquisa ou por meio de links em alguma página. Entretanto, os navegadores mais populares não indicam explicitamente qual aba gerou qual. Assim, a relação entre as abas é indicada somente por proximidade ou presença no mesmo grupo de abas.

O segundo problema se refere à quebra da organização das abas ao longo do tempo. Isso ocorre conforme o usuário abre abas entre as já existentes, fecha abas ou as reordena. Tarefas complexas tais como pesquisa exploratória podem ser acompanhadas de uma mudança constante de prioridades, o que causa uma necessidade de reorganização e possivelmente uma quebra do modelo de organização anterior. A ordenação é uma das únicas formas pelas quais o contexto em que as abas foram abertas é indicado. Além disso, organizar abas manualmente é um dos únicos recursos oferecidos para que o usuário priorize abas e represente seu modelo mental. Ou seja, tanto informações contextuais quanto o modelo mental do usuário são afetados pela quebra de organização.

O terceiro problema diz respeito à falta de informações exibidas nas abas, principalmente no que diz respeito ao contexto em que a aba foi aberta. Geralmente, o navegador exibe somente o ícone, o título da página e o link de cada aba. Mais informações possibilitariam, por exemplo, que o usuário lembrasse qual intenção possuía ao abrir aquela aba e entendesse qual aba corresponde a cada tarefa.

Chang, Hahn *et al.* (2021) acrescentaram que o design das abas não foi feito para

suportar todas essas funcionalidades e destacaram que a disposição das abas em uma lista simples e linear é ineficiente para várias dessas tarefas e não permite que o usuário represente adequadamente o seu modelo mental.

2.2 Requisitos para um bom design de abas

Com base na literatura estudada, foi elaborada uma lista compilando requisitos que uma boa solução para o problema deve satisfazer:

R1 Permitir alternar entre fontes de informação e compará-las

R2 Auxiliar os diversos usos que usuários dão para abas, tais como utilizar as abas como lembretes

R3 Possuir informações suficientes para que o usuário identifique o contexto em que a aba foi aberta e seu propósito

R4 Permitir manter e gerenciar tarefas

R5 Permitir alternar entre tarefas ou contextos e retornar para uma tarefa ou contexto anterior rapidamente

R6 Permitir ao usuário criar um reflexo apropriado do seu modelo mental para a tarefa sendo realizada. Exemplos disso seriam proporcionar uma forma de indicar prioridade das abas, relevância, se a aba já foi lida ou não, mais formas de organização e poder marcar abas como completas

R7 Facilitar a realização de tarefas complexas tais como pesquisa exploratória

R8 Não gerar sobrecarga para realização de tarefas simples tais como abrir uma rede social ou ouvir música

R9 Reduzir a necessidade de organização manual, dado que ela é custosa para o usuário

R10 Facilitar o gerenciamento de grandes quantidade de abas, reduzindo a confusão do usuário

R11 Reduzir o efeito buraco negro

Alguns desses requisitos, tais como R1 e R4, já são satisfeitos pela interface tradicional, e é importante que continuem a ser satisfeitos em novos designs. No caso da elaboração de uma extensão de navegador, esses requisitos são naturalmente cumpridos, dado que é possível manter a interface tradicional de abas de forma independente da extensão.

Capítulo 3

Soluções e trabalhos relacionados

Chang, Hahn *et al.* (2021) propuseram algumas soluções para certos problemas relacionados ao acúmulo de abas. Usuários têm uma capacidade limitada para gerenciar e navegar por grandes quantidades de abas e janelas. Portanto, a interface de gerenciamento de abas deve auxiliar o usuário a focar sua atenção nas abas mais relevantes no momento. Além disso, uma possível solução para o efeito buraco negro seria mostrar na interface algumas abas que não foram recentemente acessadas, de forma que o usuário se lembre da existência delas.

Recentemente, navegadores têm oferecido mais recursos para organizar e visualizar abas. Alguns navegadores tais como Chrome (*Google Chrome - Download the Fast, Secure Browser from Google* 2017) e Microsoft Edge (*Microsoft Edge Features & Tips* 2022) têm possibilitado agrupar abas. De forma similar, certas extensões de navegador como, por exemplo, Toby (*Toby for Chrome* s.d.) e Workona (*Workona | The work organizer for the browser* 2022) também permitem agrupar abas, com a diferença que também permitem fechá-las, descarregá-las e, futuramente, retomar o acesso. Esse tipo de solução envolvendo agrupar abas reduz temporariamente a confusão do usuário ao lidar com grandes quantidades de abas, mas eventualmente, com a criação de muitos grupos, ainda resulta em sobrecarga de abas. Além disso, manualmente organizar abas em grupos e mantê-las adequadamente organizadas ao longo de tarefas complexas pode ser um desafio para os usuários (Chang, Kim *et al.*, 2021). Dentre os benefícios da funcionalidade de agrupamento de abas, pode-se citar que essa forma adicional de organização permite que o usuário melhor represente seu modelo mental e possibilita manter abas pertencentes ao mesmo contexto juntas.

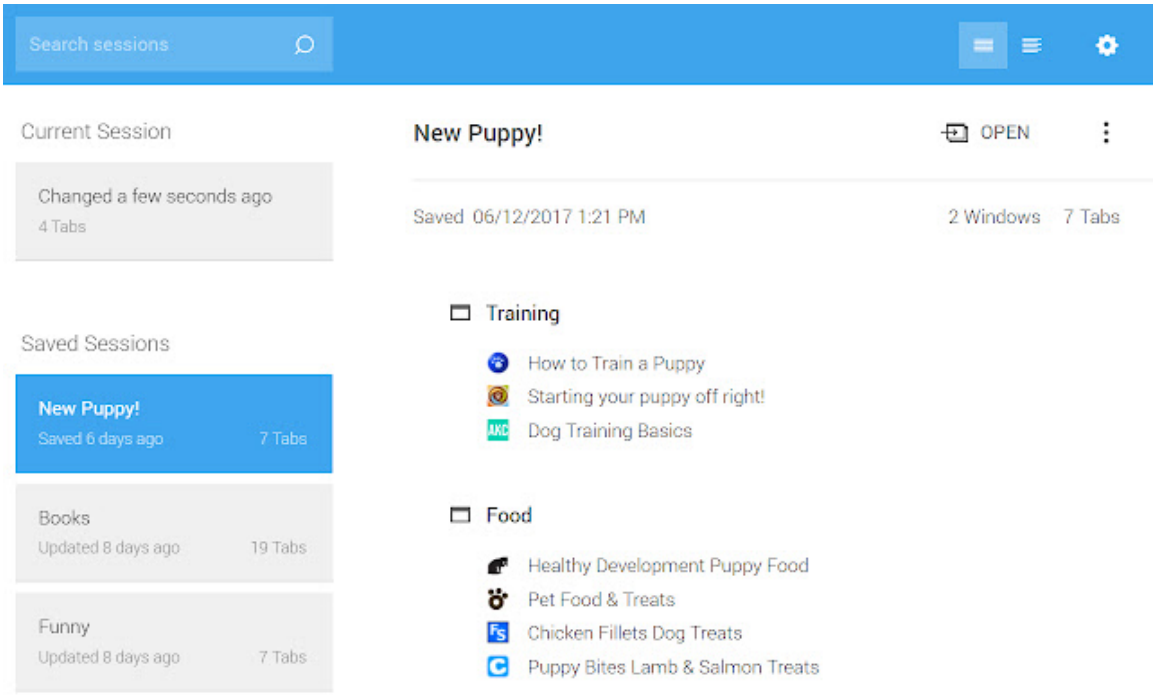

Figura 3.1: *Captura de tela da extensão Session Buddy. Fonte:* Session Buddy *s.d.*

Extensões tais como OneTab (*OneTab* 2022) e Session Buddy (*Session Buddy* s.d.) permitem fechar todas as abas e movê-las para uma lista. A partir da lista, as abas podem ser reabertas futuramente. De forma similar, navegadores costumam permitir também que usuários salvem todas as abas nos favoritos e as reabram todas de uma vez quando relevante. Esse tipo de solução diminui o consumo de memória do computador ao abrir diversas abas e permite focar somente nas abas que são relevantes no momento, reduzindo a confusão do usuário. Uma grande desvantagem dessa abordagem é esconder as abas atrás de uma interface da extensão, exacerbando o efeito buraco negro. Essas extensões também permitem gerenciar as abas em grupos, de forma similar às extensões Toby (*Toby for Chrome* s.d.) e Workona (*Workona | The work organizer for the browser* 2022) e, assim, têm os mesmos benefícios e sofrem dos mesmos problemas citados anteriormente. Na Figura 3.1, é possível observar a interface da extensão Session Buddy.

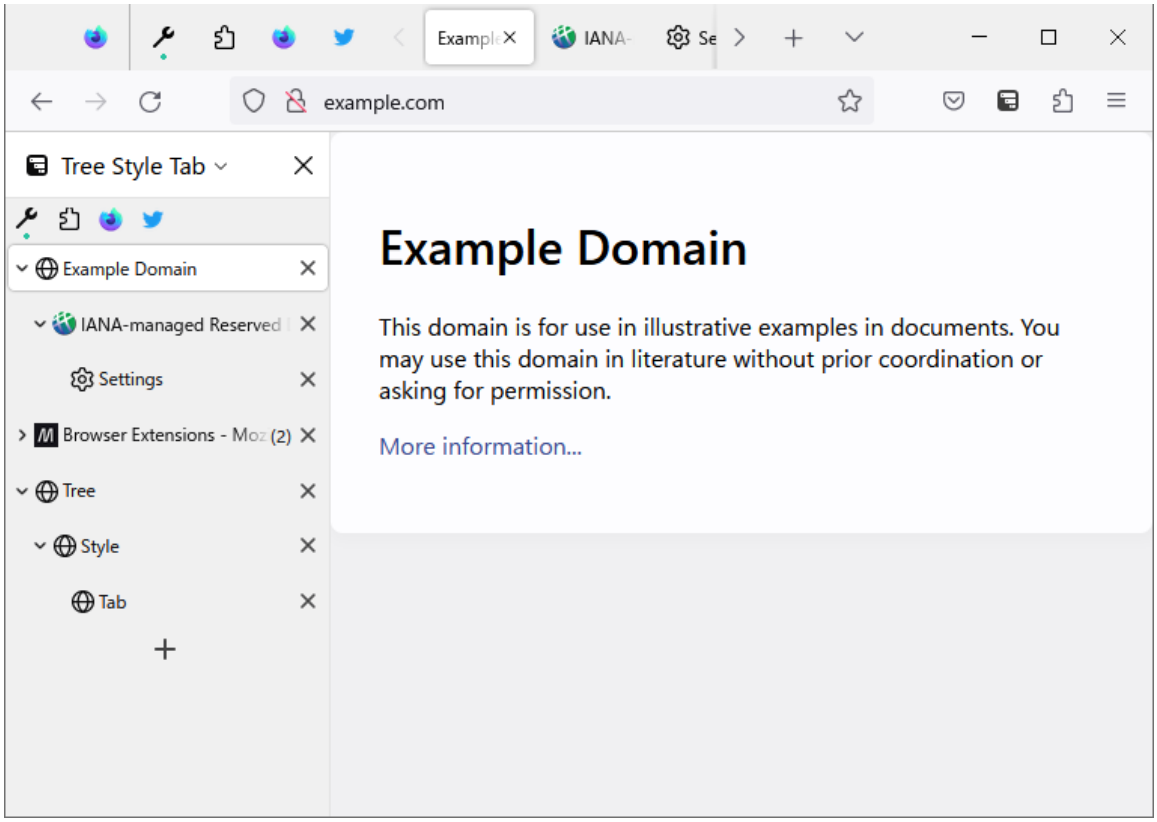

Figura 3.2: *Captura de tela da extensão Tree Style Tab. Fonte:* Tree Style Tab *2007.*

Algumas extensões tais como Tree Style Tab (*Tree Style Tab* 2007) e Tabs Outliner (*Tabs Outliner* 2022) exibem as abas em uma lista vertical e permite que elas sejam organizadas em uma hierarquia de múltiplos níveis, de forma parecida aos favoritos do navegador. As abas podem formar uma hierarquia automaticamente conforme são abertas, mas também podem ser manualmente organizadas pelo usuário. Por exemplo, ao abrir abas a partir de uma pesquisa, a pesquisa (aba progenitora) fica como raiz das outras abas. Essa funcionalidade permite discernir qual é a aba progenitora e mostrar o contexto em que cada aba foi aberta, além de reduzir a organização manual. Um possível problema é que a organização automática acontece conforme novas abas são abertas. Logo, um usuário novo da extensão não terá os benefícios da organização automática. Além disso, pode ocorrer perda de dados, ocasionando em perda da organização. Organizar em hierarquias é similar a organizar abas em grupos. Portanto, essas extensões têm os mesmos benefícios da funcionalidade de organização de abas em grupos e o mesmo problema de eventualmente resultar em sobrecarga de abas. Na Figura 3.2, é possível observar a interface da extensão Tree Style Tab.

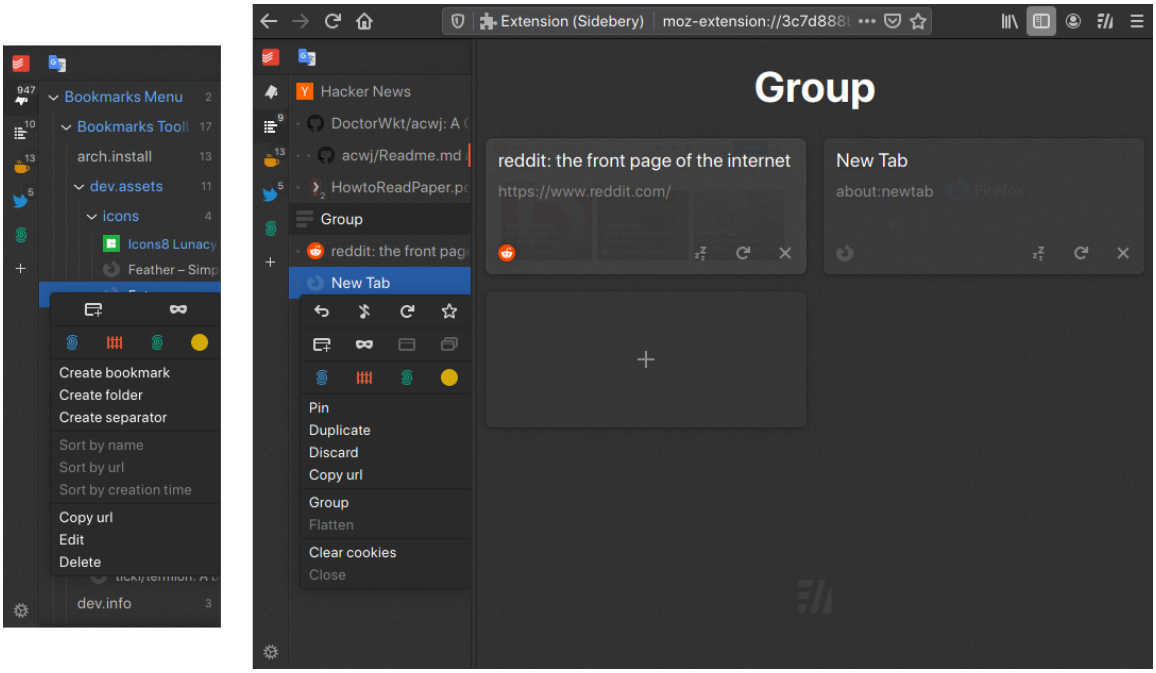

Figura 3.3: *Capturas de tela da extensão Sidebery. A primeira imagem mostra a interface de gerenciamento de favoritos, e a segunda imagem mostra a interface de gerenciamento de abas. Os botões na lateral indicam os diversos grupos de abas que podem ser acessados. Fonte:* Sidebery *2018*.

A extensão Sidebery para o navegador Firefox permite diversas funcionalidades para gerenciar abas (*Sidebery* 2018) (Figura 3.3). Uma delas diz respeito a agrupar abas de forma automática com base em seus *links*, com regras definidas pelo usuário. Por exemplo, abas de uma rede social específica podem automaticamente ser adicionadas a um grupo. A extensão *Cluster - Window & Tab Manager* (2022) possui uma funcionalidade similar. Essa abordagem pode reduzir a necessidade de organização manual. Entretanto, ela aparenta ser útil somente em casos particulares, dado que não discrimina o contexto e desfaz a organização natural das abas. Por exemplo, com o uso dessa funcionalidade, abas de uma rede social sempre serão abertas dentro de um grupo, mesmo pertencendo a tarefas completamente diferentes. Além disso, essa funcionalidade requer que o usuário escolha especificamente os sites que deseja incluir no grupo. Portanto, a organização não é totalmente automatizada. A extensão também permite organizar abas em grupos manualmente, assim o usuário pode melhor representar seu modelo mental. Somente um grupo é visualizado por vez, o que potencialmente reduz a confusão do usuário ao lidar com diversas abas e permite focar somente nas abas relevantes à tarefa atual. Entretanto, isso pode exacerbar o efeito buraco negro, dado que abas de outros grupos ficam fora do campo de visão do usuário. A extensão também organiza abas automaticamente em uma hierarquia conforme novas abas são abertas, funcionalidade cujos benefícios e problemas já foram citados.

Liu *et al.* (2012), Abela e Staff (2016) e Chang, Kim *et al.* (2021) propuseram formas de organizar abas automaticamente por meio de inteligência artificial. Essa é uma forma interessante de reduzir a necessidade de organização manual das abas. Em uma extensão, alguns modelos de inteligência artificial podem ser aplicados mesmo para usuários novos ou em caso de perda de dados. Por exemplo, a extensão elaborada por Chang, Kim *et al.*

(2021) sugere automaticamente grupos de abas para o usuário. Entretanto, resta avaliar a eficiência das soluções citadas na prática e para um volume maior de usuários.

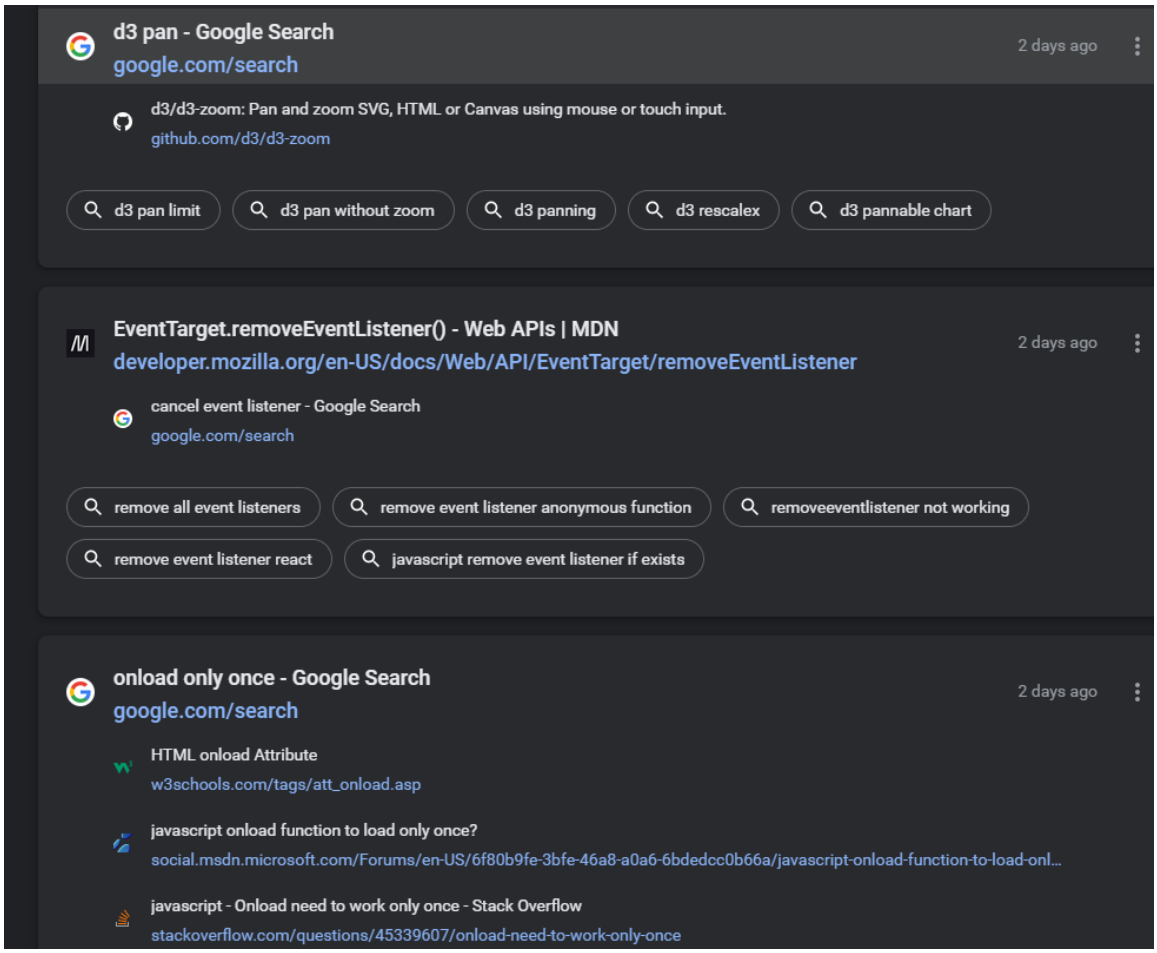

Figura 3.4: *Captura de tela da funcionalidade Journeys, disponível no navegador Google Chrome. Fonte: Autor.*

Em 2022, o navegador Google Chrome introduziu uma funcionalidade apelidada como *Journeys*, nome que poderia ser traduzido como "jornadas" (Yushkina, 2022). Ela consiste em um histórico dos sites acessados, exceto que, ao invés de exibir uma lista linear única tal como no design tradicional do histórico, os itens são automaticamente agrupados por tarefa (Figura 3.4). O agrupamento aparenta ser feito com base em título do site, *link*, momento de acesso e a partir de qual pesquisa ou site aquele *link* foi acessado. Ao digitar na barra de endereços, o navegador pode sugerir retomar uma jornada com tema semelhante às tarefas sendo realizadas na sessão atual. *Sites* abertos para uma tarefa anterior podem ser reabertos, e também é possível pesquisar por jornadas anteriores. A funcionalidade refere-se principalmente ao histórico e não às abas, mas a possibilidade de retomar jornadas anteriores sem necessitar manter as abas abertas poderia reduzir a quantidade de abas que os usuários acumulam. Além disso, o design e as ideias utilizadas poderiam servir como inspiração no desenvolvimento de uma extensão para gerenciamento de abas.

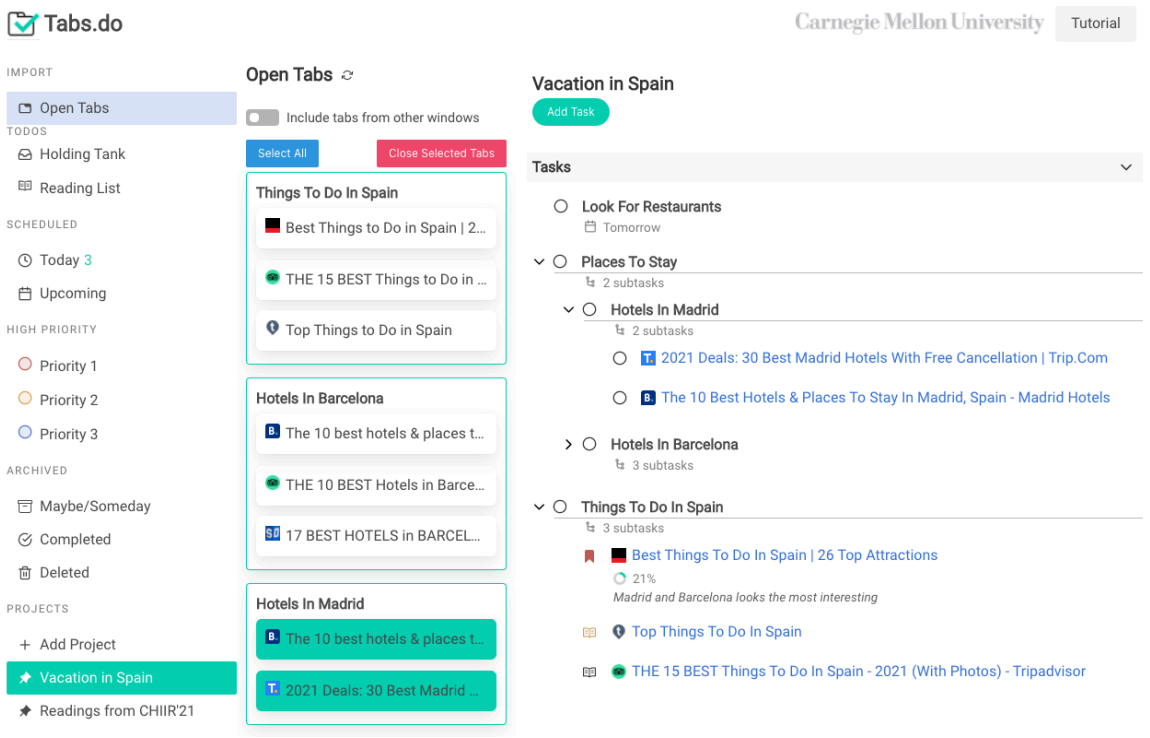

Figura 3.5: *Captura de tela da extensão Tabs.do. Fonte: Chang, Kim et al., 2021.*

Chang, Kim *et al.* (2021) descreveram o desenvolvimento da extensão de navegador Tabs.do para gerenciamento de abas. Ela possui uma interface similar a um aplicativo de gerenciamento de tarefas (Figura 3.5).

Até Novembro de 2022, não se havia indício de que a extensão estivesse disponível para o público. Dessa forma, não foi possível testar as funcionalidades. Sobretudo, a eficiência da inteligência artificial utilizada para agrupar as abas não pôde ser averiguada. Entretanto, pode ser realizada uma análise geral e crítica da extensão com base no que foi escrito no artigo (Chang, Kim *et al.*, 2021), nas imagens e nos vídeos disponíveis (Chang, 2021b; Chang, 2021a).

A extensão combina várias funcionalidades disponíveis em outras extensões já mencionadas, tais como organização de abas em uma hierarquia exibida por meio de uma lista vertical, permitir organizar abas em projetos que são visualizados um por vez, e possibilitar deletar abas e conjuntos de abas mas mantê-los dentro da extensão. Sobre essa última funcionalidade, Chang, Kim *et al.* (2021) descreveram que permitiu que usuários se sentissem menos apegados às abas ao fechá-las, possivelmente reduzindo a sobrecarga de abas. Essas funcionalidades provavelmente possuem os mesmos problemas e benefícios mencionados para outras extensões.

Tabs.do possibilita mais formas de organização para o usuário. Por exemplo, é possível adicionar prioridades e prazos às abas, permitindo que o usuário melhor represente seu modelo mental. Outra funcionalidade diz respeito a adicionar anotações a abas ou grupos de abas. Por meio dela, o usuário é capaz de escrever, por exemplo, qual intenção possuía para essas abas ou qual tarefa deveria ser realizada na aba.

Abas precisam ser salvas na extensão para usufruir dos benefícios. Ou seja, abas podem ser mantidas fora da extensão e adicionadas à extensão somente quando se deseja organizá-las. Isso pode diminuir a confusão do usuário ao organizar centenas de abas, bem como reduzir o apego a certas abas, dado que elas estão salvas na extensão e podem ser fechadas. Entretanto, essa decisão de design tem potencial para introduzir uma sobrecarga inicial desnecessária no uso da extensão.

A interface em forma de aplicativo de tarefas parece ser intuitiva e similar a algo que boa parte dos usuários deve conhecer. Esse design favorece mudanças na forma como os usuários lidam com suas abas. Por exemplo, alguns usuários optam por escrever tarefas e subtarefas antes da abertura de abas (Chang, Kim *et al.*, 2021). Gerenciar tarefas realizadas no navegador dentro do próprio navegador e mostrar as abas correspondentes a essa tarefa pode ser mais vantajoso que utilizar outro programa separado para gerenciamento de tarefas.

Chang, Kim *et al.* (2021) concluíram que a extensão foi efetiva em auxiliar o usuário a representar seu modelo mental da tarefa sendo realizada, em permitir troca rápida de contexto e em permitir que o usuário foque na tarefa atual sem ser distraído pelas abas relacionadas a outras tarefas.

Figura 3.6: *Comparação entre uma lista tradicional de abas em um navegador para dispositivos iPhone (primeira imagem) com a interface do aplicativo Bento Browser. Fonte: Hahn et al., 2018.*

Hahn *et al.* (2018) desenvolveram um navegador para dispositivos iPhone com enfoque em auxiliar o usuário em tarefas de pesquisa (Figura 3.6). O aplicativo possui uma interface de abas similar a um aplicativo gerenciador de tarefas. Abas são organizadas em tarefas e subtarefas e podem ser marcadas como importantes, para leitura e concluídas. Diversos dos benefícios e problemas relacionados ao uso desse tipo de interface no Tabs.do também estão presentes no Bento Browser.

Um diferencial do aplicativo é manter as abas e tarefas mais recentes no topo da lista. Assim, retomar tarefas recentes e provavelmente mais relevantes pode ser feito rapidamente, mesmo quando há grandes quantidades de abas. Outra funcionalidade mantém

um histórico de pesquisas realizadas e quais abas foram abertas a partir delas, permitindo que o usuário identifique o contexto em que as abas foram abertas.

Hahn *et al.* (2018) concluíram que o aplicativo facilitou que os usuários realizassem tarefas complexas de pesquisa. Tanto o Bento Browser quanto o Tabs.do podem servir de inspiração para a elaboração de interfaces de abas não convencionais.

3.1 Possíveis soluções para os problemas do design tradicional de abas

A partir da lista de requisitos que uma solução deve satisfazer (Seção 2.2) e de trabalhos relacionados, foi elaborada a Tabela 3.1 compilando as possibilidades observadas para solucionar os problemas do design tradicional de abas, de forma a satisfazer os requisitos listados anteriormente.

Rótulo	Descrição	Pode satisfazer os requisitos
S1	Mostrar mais informações e contexto sobre as abas, tais como aba progenitora e abas relacionadas	R3, possivelmente R11
S2	Oferecer mais formas de organização e categorização	R2, R3, R4, R5, R6, R7, R10
S3	Diminuir a necessidade de organização manual por meio de um algoritmo ou inteligência artificial	R3, R9, R10
S4	Utilizar uma outra forma de visualização das abas, diferente de uma lista única, horizontal e linear	Todos
S5	Guiar a atenção do usuário para abas relevantes, por exemplo, reduzindo ou aumentando sua opacidade ou tamanho da fonte, ou trazendo abas mais recentes e relevantes para o topo da lista	R2, R10
S6	Mostrar abas antigas de forma aleatória ou com uso de um algoritmo, de modo a reduzir a chance do usuário perdê-las de vista	R11

Tabela 3.1: *Possibilidades para solucionar os problemas do design tradicional de abas e alguns dos requisitos que podem ser satisfeitos por cada tipo de solução.*

Capítulo 4

Desenvolvimento de uma extensão de navegador

Este capítulo descreve o desenvolvimento de uma extensão de navegador chamada Tabi, que visa explorar soluções para os problemas do design tradicional de abas. Ela está disponível para o navegador Firefox.

4.1 Como é realizado o desenvolvimento de uma extensão de navegador

O código de uma extensão de navegador é muito parecido com o de uma página web. São utilizados primariamente HTML, CSS e JavaScript para elaboração da interface e comportamento da extensão.

Os navegadores baseados em Chromium e o navegador Firefox possuem uma API de extensões (*JavaScript APIs - Mozilla | MDN* 2022). Essa API permite obter dados e interagir com os navegadores de diversas formas. Por exemplo, é possível obter dados de abas, criá-las ou removê-las. É possível também interagir com favoritos, janelas, histórico, definir qual página da extensão é aberta em uma nova aba, dentre outras coisas. As APIs de ambos os navegadores são similares, mas existem algumas divergências que necessitam de adaptações no código para que a extensão funcione em ambos os navegadores. Além disso, algumas funcionalidades podem estar presentes em uma API mas não na outra. Por exemplo, no Firefox é possível mostrar a extensão no painel lateral, mas na API dos navegadores Chromium isso não é possível de forma oficial.

A API de navegadores também providencia um armazenamento de dados no formato JSON. O armazenamento pode ser local ou sincronizado por meio de um servidor do navegador. Dessa forma, fazendo uso desse armazenamento, não é necessário que a extensão inclua um banco de dados ou necessite de um servidor para sincronização dos dados, desde que o tamanho do arquivo de dados não ultrapasse um dado limite.

Um arquivo chamado `manifest.json` define detalhes sobre extensão, quais permissões são utilizadas, qual o caminho dos arquivos de ícones da extensão e alguns outros

detalhes para que a extensão possa ser carregada pelo navegador.

Após escrever o código e configurar o `manifest.json`, basta empacotar os arquivos em um formato definido pelo navegador para que a extensão possa ser utilizada por outros usuários. No caso do navegador Firefox, o formato utilizado é o ZIP. Também é possível acessar a extensão temporariamente sem empacotá-la ou publicá-la, o que auxilia no desenvolvimento do código.

Os navegadores Firefox e Google Chrome disponibilizam um catálogo oficial de extensões (*Extensions –Add-ons for Firefox (en-US)* 2018; *Chrome Web Store* 2019). Usuários podem instalar extensões do catálogo rapidamente e de forma gratuita, e o catálogo permite pesquisar e descobrir novas extensões. Qualquer pessoa pode submeter uma extensão para esses catálogos. Para que a extensão seja publicada, ela passa por um processo de aprovação. A extensão Tabi foi rapidamente aprovada no catálogo de extensões do Firefox (*Tabi - Visualize Tabs* 2022).

4.2 Design e funcionalidades da extensão

Após a leitura de artigos sobre o design de abas, foi dado início ao processo de desenvolvimento de uma extensão de navegador. Diversas soluções possíveis foram consideradas, e foi feita uma análise preliminar sobre quais problemas cada solução resolveria e quais seriam as limitações de cada solução.

Finalmente, optou-se por um design baseado em múltiplas listas de abas. O design que foi inicialmente desenvolvido ocupava boa parte da tela e seria mostrado quando o usuário utilizasse um atalho ou clicasse no ícone da extensão. Na Figura 4.1, é possível observar esse design inicial. Para isso ser feito com boa performance, seria necessário que a interface estivesse sempre carregada em segundo plano de forma persistente. Isso não pode ser realizado por conta de limitações na API dos navegadores. Por esse motivo, a extensão foi contida no painel lateral do navegador e algumas funcionalidades foram abandonadas por conta do limite de espaço. A descrição do design da extensão será feita com base na implementação final, e não na proposta inicial ou no design implementado inicialmente.

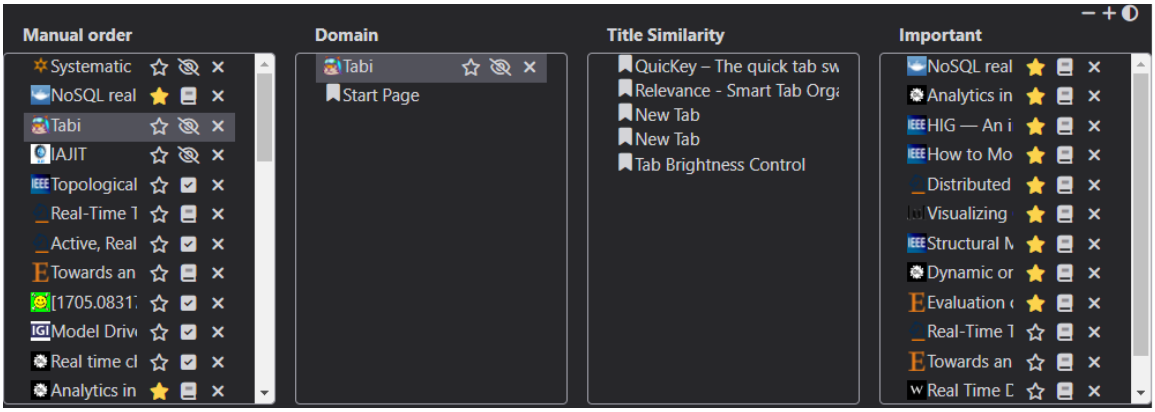

Figura 4.1: *Captura de tela do design inicial da extensão. Esse design utilizava grande parte da tela. Fonte: autor.*

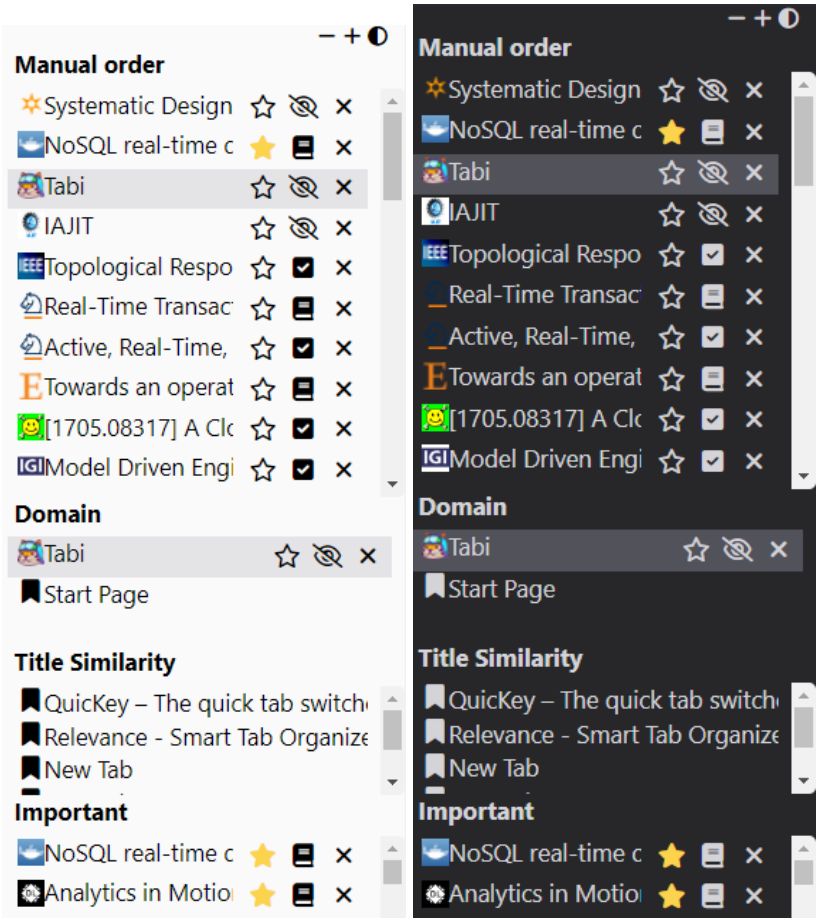

Figura 4.2: *Capturas de tela do design final da extensão nos modos claro e escuro. Fonte: autor.*

No design final da extensão (Figura 4.2), uma aba pode ser mostrada em mais de uma lista, e cada lista serve um propósito específico. Algumas listas também mostram favoritos relacionados, e não somente abas. A motivação por trás disso é facilitar que usuários entendam o propósito dessas listas, dado que também possuir favoritos aumenta as chances das listas serem preenchidas e não estarem vazias. As listas implementadas exibem itens da seguinte forma:

- Abas em ordem manual, de forma similar a como os navegadores exibem as abas;
- Abas e favoritos com o mesmo domínio da aba atual. Por exemplo, se a aba atual é de uma rede social, essa lista mostra todas as abas e favoritos dessa mesma rede social;
- Abas e favoritos com título similar ao da aba atual. Por exemplo, se a aba atual contém a palavra “receita”, então essa lista deverá mostrar outras abas e favoritos contendo a palavra “receita”;
- Abas marcadas como importantes.

Mostrar abas do mesmo domínio e abas com títulos similares tem a intenção de mostrar abas relacionadas sem necessidade de organização manual pelo usuário. O algoritmo

de similaridade de título utilizado foi implementado com base no coeficiente de Sørensen-Dice para similaridade de textos (Kondrak *et al.*, 2003), definido como:

$$s = \frac{2n_t}{n_x + n_y} \tag{4.1}$$

onde n_x é o número de bigramas em um texto x, n_y é o número de bigramas em um texto y, e n_t é o número de bigramas encontrados em ambos os textos. Um bigrama é uma sequência de duas letras vizinhas. Por exemplo, os bigramas da palavra "noite" são "no", "oi", "it", "te".

A extensão possui duas formas de categorizar abas. A primeira se dá por um botão em forma de estrela que permite marcar abas como importantes. A segunda consiste em um botão com três estados. Inicialmente, este botão indica que a aba não foi lida. Após o usuário clicar, o botão marca a aba para leitura. Após mais um clique, o botão marca a aba como lida.

A extensão também permite alternar entre os modos claro e escuro, bem como aumentar e diminuir o tamanho da fonte. Isso foi implementado logo na primeira versão por ser considerado essencial para a usabilidade da extensão.

Uma das funcionalidades planejadas para a extensão era mostrar mais contexto sobre as abas por meio da aba progenitora. Por exemplo, a aba progenitora poderia ser indicada, e abas abertas a partir da mesma aba progenitora seriam mostradas em uma lista, como forma de exibição de abas relacionadas. Isso também permitiria outras possibilidades de algoritmos para obtenção de abas relacionadas.

A API do Firefox disponibiliza um dado sobre as abas chamado `openerTabId`. Esse dado diz respeito ao identificador (id) da aba progenitora de uma dada aba. Na prática, foi observado que esse valor estava quase sempre vazio, principalmente para as abas mais antigas. Apesar disso, a extensão poderia armazenar as abas progenitoras sem depender da API. Entretanto, optou-se por não fazer isso. O motivo é que a extensão foi planejada para ser o mais determinística possível. Ou seja, as principais funcionalidades da extensão deveriam funcionar adequadamente para novos usuários ou em caso de perda de dados. Implementar uma funcionalidade baseada nas abas progenitoras é uma possibilidade futura.

Na falta de dados sobre a aba progenitora, uma tentativa foi mostrar abas abertas no mesmo dia da aba atual. O dado indicando a data em que uma aba foi aberta está disponível pela API do navegador. Entretanto, esse dado não estava presente para as abas que foram abertas há muito tempo. Por esse motivo, essa possibilidade também foi descartada.

As funcionalidades descritas acima utilizam as soluções S1, S2, S3 e S4. A versão desenvolvida da extensão visa satisfazer principalmente os requisitos R3, R6, R7, R9. As funcionalidades S5 e S6 foram planejadas para satisfazer R2, R10, R11, mas não foram implementadas. R1, R4, R8 são satisfeitos naturalmente pela interface de abas tradicional. Não houve foco em satisfazer R5.

4.3 Detalhes técnicos

Inicialmente, a intenção era disponibilizar a extensão tanto para o navegador Firefox quanto para navegadores baseados em Chromium, tais como Microsoft Edge e Google Chrome. Entretanto, a API do Chromium não possui uma forma de adicionar um painel lateral. Por esse motivo, a extensão foi publicada somente para Firefox. Entretanto, a extensão ainda pode ser usada em certos navegadores baseados em Chromium que possuem um painel lateral, como por exemplo o navegador Vivaldi (*Vivaldi Browser Features | Everything' s an option* 2015). Para que isso funcione, a página da extensão deve ser adicionada a esse painel lateral. Existem alguns detalhes de aparência que não ficam corretos ao se fazer isso.

Dentre as ferramentas utilizadas na extensão, tem-se em destaque Svelte e Tailwind.

Svelte é um arcabouço *front-end* para criação de interfaces web interativas (*Svelte • Cybernetically enhanced web apps* 2022). É uma alternativa para outros arcabouços populares tais como React e Vue. Uma das grandes vantagens do Svelte é a alta performance da aplicação. Ao desenvolver extensões para navegadores, a performance tem grande importância. Por exemplo, o usuário não deve esperar 1 segundo toda vez que interagir com suas abas. Outra possível vantagem do Svelte é a facilidade de escrita do código em relação a arcabouços tais como React. A pesquisa *Stack Overflow Developer Survey 2022* (2022) revelou que o Svelte é o segundo arcabouço web mais adorado pela comunidade.

Tailwind é um arcabouço para estilos CSS (*Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* 2020). Ele disponibiliza diversas classes que são adicionadas diretamente aos elementos em HTML para produzir o efeito desejado. O arcabouço possibilita manter a mesma aparência da página em diversos navegadores. Isso inicialmente era importante para o desenvolvimento da extensão, dado que ela estava sendo desenvolvida para múltiplos navegadores. Outra vantagem é que o Tailwind oferece diversos padrões pré-definidos de tamanho, espaçamento e cores. Além disso, o arcabouço pode ser considerado mais simples e legível de escrever que estilos CSS.

A arquitetura utilizada para o projeto pode ser observada na Figura 4.3. Ela consiste principalmente de duas classes. A classe `ExtensionStorage` gerencia o armazenamento da extensão por meio da API do navegador. Ela permite que outras classes possam obter e atualizar o armazenamento da extensão sem perda de dados, além de inicializar os valores padrão da extensão. A classe `BrowserMediator` faz uma ponte entre o Svelte e o Firefox. Ela possui diversas funções para obter dados sobre abas e favoritos de uma forma que pode ser visualizada facilmente por meio do arcabouço *front-end* utilizado. Além disso, possui funções que obtêm e atualizam dados do armazenamento da extensão por meio da classe `ExtensionStorage`.

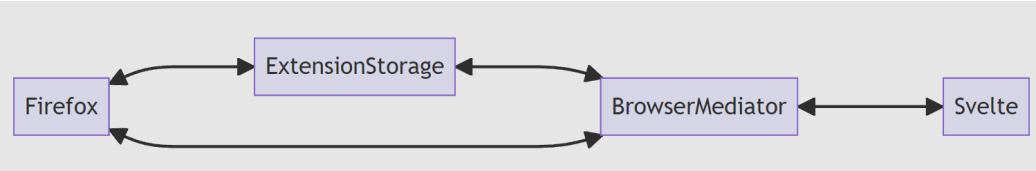

Figura 4.3: *Diagrama da arquitetura do projeto. Fonte: autor.*

O processo de desenvolvimento da extensão requereu muitas tomadas de decisão sobre o design, sobre quais funcionalidades seriam implementadas e sobre como contornar as limitações e divergências das APIs de extensões dos navegadores. Houve uma preocupação em relação a se o usuário entenderia o propósito da extensão e como utilizá-la, dado que o design poderia não ser familiar. Além disso, houve um cuidado em manter o código legível e limpo. Logo, desenvolver a extensão necessitou de decisões sobre a arquitetura do projeto e sobre a organização do código. Outra dificuldade na elaboração da extensão foi a falta de familiaridade com as ferramentas e bibliotecas utilizadas.

4.4 Publicação da extensão

A extensão foi publicada na loja de extensões do Firefox, de forma que qualquer usuário do navegador possa facilmente instalar a extensão de forma gratuita.

O projeto possui código aberto e está disponível na plataforma GitHub. O código utiliza a licença MIT, que permite modificação, distribuição, uso pessoal e uso comercial, mas que não prevê nenhuma garantia ou responsabilidade por parte do autor.

O nome escolhido para a extensão foi Tabi. Esse nome faz referência à palavra *tab* em inglês, que significa abas, e à palavra 旅 em japonês, que é pronunciada como "tabi" e pode ser traduzida como jornada.

Um ícone para a extensão foi gerado por meio da inteligência artificial DALL-E 2 (OpenAI, 2022). Ele representa um gato malhado indo em uma jornada. A imagem é um trocadilho entre o nome da extensão e "tabby cat", que pode ser traduzido do inglês como "gato malhado". O ícone pode ser observado na Figura 4.4.

Figura 4.4: *Ícone da extensão Tabi. Fonte: autor.*

Foi realizada uma divulgação da extensão nos sites Reddit (*Tabi - New vertical tabs*

add-on that also shows related tabs 2022) e Hacker News (*Tabi – Visualize related tabs on Firefox, made with Svelte | Hacker News* 2022), no grupo do Telegram do Bacharelado de Ciência da Computação do Instituto de Matemática e Estatística da Universidade de São Paulo, e para alguns conhecidos do autor. Entretanto, até o final de 2022, haviam poucos usuários da extensão. Por esse motivo, não foi possível obter comentários com melhorias e elogios à extensão.

Capítulo 5

Análise dos resultados da extensão

5.1 Relato de utilização da extensão

A seguir, será feito um relato pessoal de utilização da extensão elaborado pelo autor, de forma a elucidar uma das formas como ela pode ser usada. Os acontecimentos se referem ao dia 2 de Dezembro de 2022, meses após a publicação da extensão, em 10 de Outubro de 2022. O relato foi prontamente registrado logo após o ocorrido. Entre a publicação da extensão e os acontecimentos, o autor fez pouco uso da extensão. Logo, não havia mais uma certa familiaridade com a aplicação. Pode-se dizer que o autor nesse momento passou a ver a extensão tal como um usuário, e menos como um desenvolvedor. O navegador utilizado foi o Vivaldi. A extensão não oferece suporte oficial para esse navegador, mas pode ser utilizada adicionando a página da extensão ao painel lateral. Ao fazer isso, a extensão funciona corretamente, mas possui alguns problemas relacionados à aparência.

Foi realizado um trabalho de faculdade que consiste na elaboração de um artigo científico sobre sistemas de banco de dados em tempo real.

Inicialmente, não se sabia muito sobre o tema. Após diversas tentativas de pesquisa por artigos da área, foi encontrada a página https://dblp.org/search?q=real+time+database contendo referências a diversas publicações sobre o tema.

A partir dessa página, foram abertas dezenas de abas com artigos que pareciam relevantes. Nessa etapa, a falta de conhecimento sobre o tema não permitia discernir adequadamente quais artigos seriam mais importantes para a realização do trabalho. Além disso, foi experienciada uma sobrecarga de informação, dado que haviam muitos artigos abrangendo conceitos complexos.

Após abertas as páginas contendo informações sobre os artigos escolhidos, iniciou-se a verificação de cada um deles, auxiliada pela extensão. Artigos com mais citações e que pareciam conter informações gerais que poderiam ser utilizadas no trabalho foram marcados como importantes. Após alguns minutos, percebeu-se que marcar esses artigos como “para leitura” faria mais sentido, e então isso foi realizado. Artigos muito específicos

e com poucas citações foram marcados como lidos. Para alguns casos, havia uma dúvida sobre o artigo ser relevante ou não, e por esse motivo a aba não foi marcada. Após uma segunda verificação, essas abas não marcadas foram constatadas como irrelevantes e, portanto, marcadas como lidas. Na Figura 5.1, é possível observar a extensão após essa etapa da pesquisa.

Dezenas de abas haviam sido marcadas para leitura, mas o trabalho não requeria tantas fontes de informações. Logo, a intenção era escolher quais artigos teriam prioridade e seriam lidos primeiro para elaboração do trabalho. A extensão mostra todas as abas importantes e marcadas para leitura em uma lista específica. Isso permitiu ter uma visão geral dos artigos relevantes e escolher quais seriam priorizados. Os artigos mais relevantes dentre os marcados para leitura foram então marcados como importantes (Figura 5.2). Nessa etapa, havia um entendimento maior sobre a área a ser estudada e sobre qual tipo de informação poderia ser incluída no trabalho.

Após escolhidas as abas mais importantes, era desejado copiar seus *links*, bem como os *links* das abas para leitura, de forma a poder compartilhar com outros membros do grupo do projeto. A extensão não dava suporte para selecionar múltiplas abas, bem como realizar certas operações tal como copiar seus *links*, ordená-las manualmente ou movê-las para outra janela. Por esse motivo, optou-se por copiar os *links* manualmente.

Esse relato pode ser enquadrado como um caso de pesquisa exploratória. A extensão parece auxiliar o usuário nessa tarefa por permitir categorizar as abas e por possuir uma lista separada somente com abas relevantes. Entretanto, além da falta de suporte para certas operações envolvendo abas e para múltiplas abas, o autor observou na prática algumas limitações e possibilidades de melhoria.

Percebeu-se que o botão “importante” foi utilizado para selecionar as abas mais importantes, mas também poderia cumprir a função de marcar abas que servem como lembretes ou são urgentes. Idealmente, abas urgentes e realmente importantes deveriam ser facilmente distinguíveis de outras abas. Uma possibilidade seria permitir níveis de prioridade, utilizando cores diferentes ou símbolos diferentes. Para alternar entre prioridades, o usuário poderia clicar mais de uma vez no botão, utilizar um atalho ou abrir um menu que permite escolher a prioridade. Assim, por exemplo, abas urgentes ou que servem como lembretes poderiam ser marcadas com prioridade 1, possuindo cor amarela, enquanto as abas correspondentes aos artigos poderiam ser marcadas com prioridade 4, possuindo a cor azul.

Houve incerteza do usuário sobre qual botão utilizar para categorizar a aba em caso de dúvida. Uma possibilidade interessante seria permitir que o usuário adicione anotações às abas. Abas com anotações seriam marcadas com um visual específico, por exemplo um ícone, e a informação poderia ser lida e modificada ao clicar no ícone ou na aba. Outra possibilidade é, se diversas categorias aparentarem ser necessárias, seria possível permitir que o usuário configure o comportamento dos botões, adicionando mais categorias e ícones para um mesmo botão. Por exemplo, o usuário poderia adicionar um ícone representando “em dúvida”.

Outro problema encontrado foi que abas marcadas como lidas são difíceis de distinguir de abas marcadas para leitura. Isso poderia ser resolvido reduzindo o brilho de abas

marcadas como lidas.

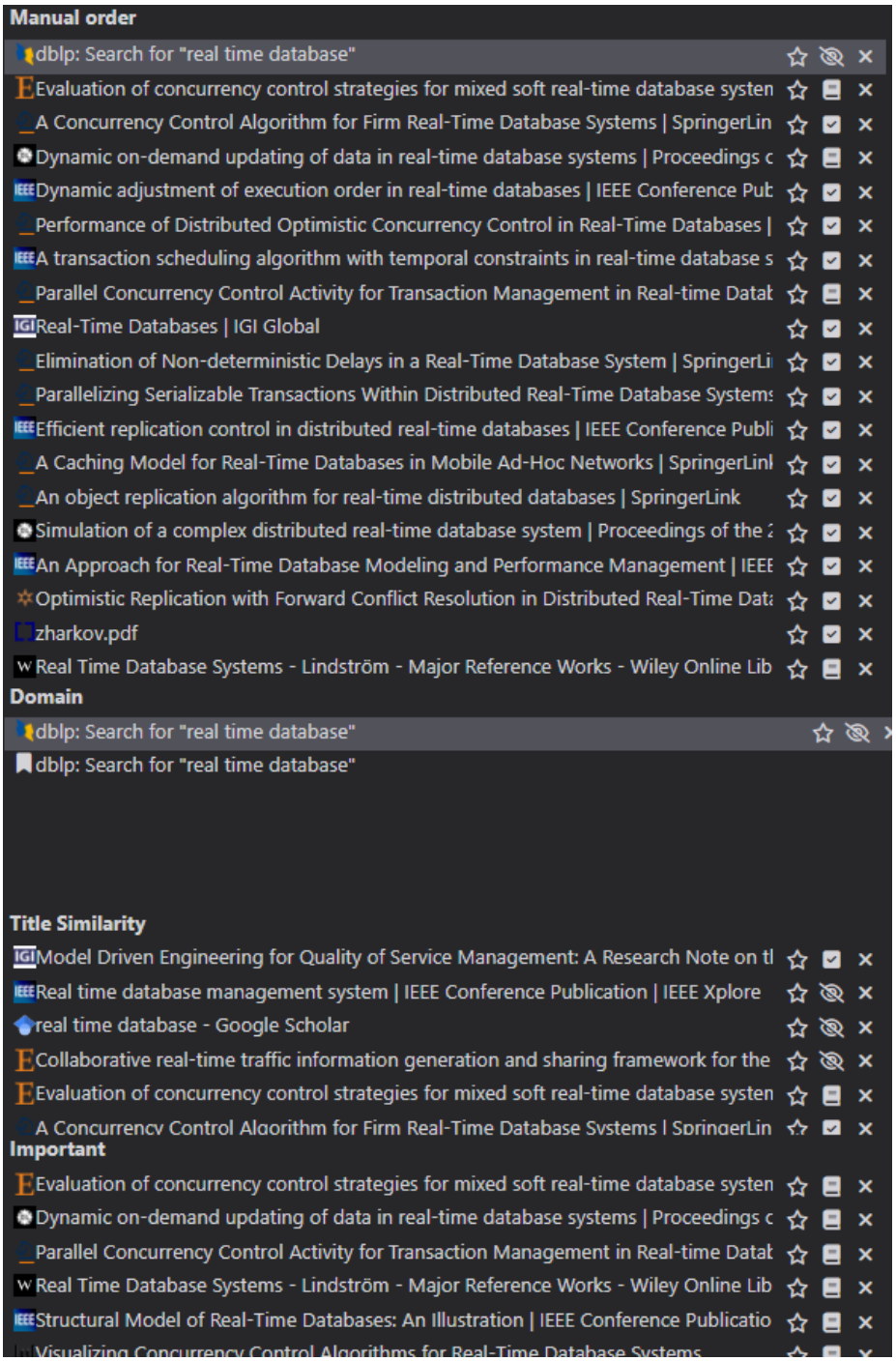

Figura 5.1: *Captura de tela da extensão durante realização de pesquisa exploratória. Nessa etapa, algumas abas foram marcadas para leitura e outras abas foram concluídas. Fonte: autor.*

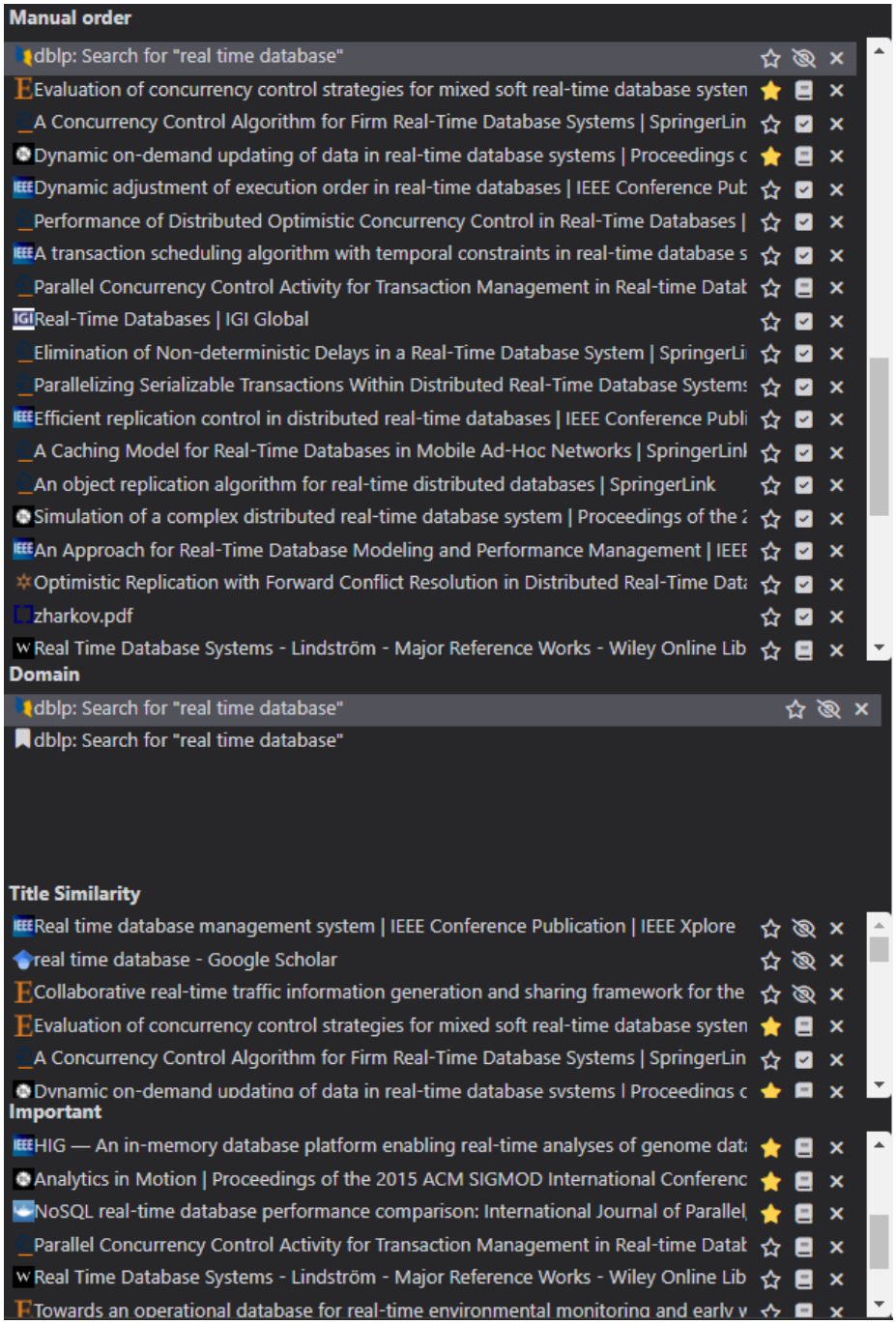

Figura 5.2: *Captura de tela da extensão durante realização de pesquisa exploratória. Nessa etapa, algumas abas anteriormente marcadas para leitura foram marcadas como importantes. Para isso, foi utilizada a lista de abas importantes, que também exibe abas para leitura. Fonte: autor.*

5.2 Resultados e dificuldades

Sobretudo, a maior dificuldade encontrada foi relacionada a limitações na API dos navegadores. Não foi possível implementar o design planejado com boa usabilidade e performance. Manter a extensão no painel lateral impactou a usabilidade da extensão. Além disso, a API não permitiu uma forma de obter abas abertas no mesmo contexto, seja

por aba progenitora ou por data de abertura.

Para obter a similaridade de título entre as abas, foi necessário pesquisar sobre o assunto e considerar diversos algoritmos e soluções. Apesar disso, o resultado obtido não foi satisfatório. Outro algoritmo ou solução pronta poderia ser utilizado. Também poderia ser desenvolvida uma inteligência artificial para obter abas relacionadas.

Não se obteve uma conclusão apropriada sobre a eficácia da lista que exibe abas do mesmo domínio. Ela parece ser útil em alguns casos, tais como navegar por todas as abas de uma rede social específica. Entretanto, ela utiliza um espaço do painel lateral que poderia ser ocupado pelas outras listas. A funcionalidade talvez seria mais eficiente no design inicial da extensão, dado que ocuparia menos do espaço total da página, e permitiria o usuário acessar rapidamente abas do mesmo domínio quando necessário. Observou-se que essa funcionalidade poderia ser substituída por uma barra de pesquisa que permita pesquisar abas pelo domínio. Pesquisar por abas é uma funcionalidade que foi observada em alguns navegadores tais como Vivaldi (*Vivaldi Browser Features | Everything' s an option* 2015). Alternativamente, as listas de abas relacionadas por domínio e abas relacionadas por títulos poderiam ser fundidas em uma nova lista, utilizando um novo algoritmo de similaridade que leva em conta ambos os dados.

Portanto, de forma geral, a extensão deixou a desejar em suas tentativas de mostrar mais informações e contexto sobre as abas e diminuir a necessidade de organização manual. Entretanto, permitir mais formas de organização das abas e mostrar abas marcadas como importantes em uma lista própria parece ter sido efetivo para que o usuário recrie seu modelo mental, e parece auxiliar na tarefa de pesquisa exploratória. Desconsiderando as listas de abas relacionadas, o design final ficou muito similar a uma lista de abas verticais, que é uma funcionalidade presente em alguns navegadores tais como o Vivaldi (*Vivaldi Browser Features | Everything' s an option* 2015) e Microsoft Edge (*Microsoft Edge Features & Tips* 2022).

Logo, a extensão se propôs a satisfazer os requisitos R3 (indicar contexto das abas), R6 (permitir criar um reflexo do modelo mental do usuário) , R7 (facilitar pesquisa exploratória), R9 (reduzir organização manual) por meio das soluções S1 (mostrar abas relacionadas), S2 (oferecer mais formas de organização), S3 (utilizar algoritmo para relacionar abas automaticamente), S4 (utilizar um design diferente para visualização). Ela falhou em satisfazer os requisitos R3 e R9, mas parece ter tido um resultado satisfatório para os requisitos R6 e R7 por meio da solução S3. As soluções S1, S2 e S4 não tiveram uma implementação que gerou resultados satisfatórios. As soluções S5 (reduzir opacidade de abas marcadas como ”lidas”) e S6 (mostrar abas antigas de forma aleatória) não foram implementadas na versão final, e poderiam satisfazer outros requisitos. S1 poderia ter satisfeito o requisito R11 (reduzir o efeito buraco negro), mas isso não foi observado na prática, dado que a implementação de S1 não foi satisfatória.

Capítulo 6

Considerações finais

Após uma análise dos diversos problemas do design atual de abas e soluções propostas por trabalhos anteriores, foi desenvolvida uma extensão de navegador que tenta abordar alguns dos problemas estudados. A extensão, chamada Tabi, permite categorizar abas como "importantes", "para leitura" e "lidas". Além disso, possui listas adicionais que exibem abas e favoritos relacionados à aba atual com base no domínio e na similaridade de títulos.

Tabi foi publicada para o navegador Firefox, fazendo uso da API disponibilizada pelo navegador para obtenção de informações sobre abas e favoritos. A extensão foi desenvolvida de forma similar a uma página web, fazendo uso de HTML, CSS e JavaScript. Para isso, foi utilizado o arcabouço *front-end* Svelte (*Svelte • Cybernetically enhanced web apps* 2022), que garantiu uma boa performance para a aplicação, e o arcabouço Tailwind (*Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* 2020), que facilitou desenvolver a aparência da extensão.

As funcionalidades de categorização de abas implementadas se demonstraram úteis para realização de pesquisa exploratória, além de auxiliar o usuário a melhor representar seu modelo mental da tarefa sendo realizada.

Entretanto, vários dos requisitos listados para uma boa solução não foram satisfeitos pela extensão. Refinar as soluções presentes, bem como implementar as soluções restantes de forma a satisfazer mais requisitos, são tarefas que poderiam ser realizadas no futuro.

Algumas possíveis melhorias para a extensão seriam:

- Reduzir a opacidade de abas já lidas;
- Mostrar abas antigas aleatoriamente para que o usuário não as perca de vista;
- Utilizar outro algoritmo para mostrar abas relacionadas;
- Permitir adicionar anotações às abas;
- Implementar funcionalidades disponíveis na interface tradicional de abas, tais como ordenação manual de abas e seleção de múltiplas abas.

Além disso, seria interessante explorar outros designs não convencionais. Por exemplo, as abas poderiam ser visualizadas como nós de um grafo. Os arcos indicariam relacionamentos entre as abas, por exemplo, domínio e aba progenitora.

A extensão foi divulgada em algumas plataformas, mas quase não obteve usuários. Por esse motivo, não foi possível obter retorno de usuários para avaliação da extensão e implementação de melhorias. Um passo importante para avaliar a solução proposta seria utilizar um questionário padronizado, tal como o *System Usability Scale* (Brooke, 1995).

As principais dificuldades encontradas se referem às limitações das APIs dos navegadores para criação do design inicial para a extensão e para a obtenção de informações sobre o contexto das abas. Logo, uma possível proposta seria que APIs de navegadores permitam modificar a interface do usuário. Um exemplo disso seria permitir que uma página da extensão esteja sempre carregada e possa sobrepor a aba atual a partir de um comando do usuário. Isso possibilitaria outros designs ao desenvolver extensões de navegador. Outra proposta seria permitir obter mais dados sobre o contexto das abas por meio da API. Por exemplo, o navegador poderia garantir que o dado sobre a aba progenitora esteja sempre presente na propriedade `openerTabId`. Outra opção seria permitir acesso aos dados sobre o contexto em que cada aba foi aberta utilizados pela funcionalidade Journey (Yushkina, 2022) no navegador Google Chrome.

Referências

[Abela e Staff 2016] Charlie Abela e Chris Staff. "Behaviour mining for automatic task-keeping and visualisations for task-refinding". In: *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*. 2016, pp. 23–32 (citado na pg. 10).

[Brooke 1995] John Brooke. "SUS: a quick and dirty usability scale". *Usability Eval. Ind.* 189 (nov. de 1995) (citado na pg. 30).

[Chang 2021a] Joseph Chee Chang. *Tabs.do UIST 30s Preview*. YouTube, ago. de 2021. url: https://www.youtube.com/watch?v=he--Ly0UQ-4 (acesso em 21/12/2022) (citado na pg. 12).

[Chang 2021b] Joseph Chee Chang. *Tabs.do UIST Virtual Presentation V4*. YouTube, set. de 2021. url: https://www.youtube.com/watch?v=ZwbVzDRFbGs (acesso em 21/12/2022) (citado na pg. 12).

[Chang, Hahn *et al.* 2021] Joseph Chee Chang, Nathan Hahn *et al.* "When the tab comes due: challenges in the cost structure of browser tab usage". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–15 (citado nas pgs. 3, 4, 7).

[Chang, Kim *et al.* 2021] Joseph Chee Chang, Yongsung Kim *et al.* "Tabs. do: Task-Centric Browser Tab Management". In: *The 34th Annual ACM Symposium on User Interface Software and Technology*. 2021, pp. 663–676 (citado nas pgs. 4, 7, 10, 12, 13).

[*Chrome Web Store* 2019] *Chrome Web Store*. Google.com, 2019. url: https://chrome.google.com/webstore/category/extensions?hl=en (citado na pg. 16).

[*Cluster - Window & Tab Manager* 2022] *Cluster - Window & Tab Manager*. chrome.google.com. url: https://chrome.google.com/webstore/detail/cluster-window-tab-manage/aadahadfdmiibmdhfmpbeeebejmjnkef?hl=pt-BR&gl=US&authuser=1 (acesso em 21/12/2022) (citado na pg. 10).

[*Extensions –Add-ons for Firefox (en-US)* 2018] *Extensions –Add-ons for Firefox (en-US)*. Mozilla.org, 2018. url: https://addons.mozilla.org/en-US/firefox/extensions/ (acesso em 21/12/2022) (citado na pg. 16).

[*Google Chrome - Download the Fast, Secure Browser from Google* 2017] *Google Chrome - Download the Fast, Secure Browser from Google*. Google.com, 2017. URL: https://www.google.com/chrome/ (acesso em 21/12/2022) (citado na pg. 7).

[Hahn *et al.* 2018] Nathan Hahn, Joseph Chee Chang e Aniket Kittur. "Bento browser: complex mobile search without tabs". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–12 (citado nas pgs. 1, 3, 4, 13, 14).

[*JavaScript APIs - Mozilla | MDN* 2022] *JavaScript APIs - Mozilla | MDN*. Mozilla.org, set. de 2022. URL: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API (acesso em 21/12/2022) (citado na pg. 15).

[Kondrak *et al.* 2003] Grzegorz Kondrak, Daniel Marcu e Kevin Knight. "Cognates can improve statistical translation models". In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–Short Papers - Volume 2*. NAACL-Short '03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 46–48. DOI: 10.3115/1073483.1073499. URL: https://doi.org/10.3115/1073483.1073499 (citado na pg. 18).

[Liu *et al.* 2012] Jie Liu, Chun Yu, Wenchang Xu e Yuanchun Shi. "Clustering web pages to facilitate revisitation on mobile devices". In: *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. 2012, pp. 249–252 (citado na pg. 10).

[Mehrotra *et al.* 2016] Rishabh Mehrotra, Prasanta Bhattacharya e Emine Yilmaz. "Characterizing users' multi-tasking behavior in web search". In: *Proceedings of the 2016 ACM on conference on human information interaction and retrieval*. 2016, pp. 297–300 (citado na pg. 1).

[*Microsoft Edge Features & Tips* 2022] *Microsoft Edge Features & Tips*. Microsoft.com, 2022. URL: https://www.microsoft.com/en-us/edge/features?form=MT00D9 (acesso em 21/12/2022) (citado nas pgs. 7, 27).

[*OneTab* 2022] *OneTab*. chrome.google.com. URL: https://chrome.google.com/webstore/detail/onetab/chphlpgkkbolifaimnlloiipkdnihall?hl=pt-BR (acesso em 23/12/2022) (citado na pg. 8).

[OpenAI 2022] OpenAI. *DALL·E 2*. OpenAI, abr. de 2022. URL: https://openai.com/dall-e-2/ (acesso em 21/12/2022) (citado na pg. 20).

[*Session Buddy* s.d.] *Session Buddy*. chrome.google.com. URL: https://chrome.google.com/webstore/detail/session-buddy/edacconmaakjimmfgnblocblbcdcpbko?hl=en (citado na pg. 8).

[*Sidebery* 2018] *Sidebery*. Mozilla.org, set. de 2018. URL: https://addons.mozilla.org/en-US/firefox/addon/sidebery/ (acesso em 21/12/2022) (citado na pg. 10).

[*Stack Overflow Developer Survey 2022* 2022] *Stack Overflow Developer Survey 2022*. Stack Overflow, 2022. URL: https://survey.stackoverflow.co/2022/#technology-most-loved-dreaded-and-wanted (citado na pg. 19).

[*Svelte • Cybernetically enhanced web apps* 2022] *Svelte • Cybernetically enhanced web apps*. Svelte.dev, 2022. URL: https://svelte.dev/ (acesso em 21/12/2022) (citado nas pgs. 19, 29).

[*Tabi - New vertical tabs add-on that also shows related tabs* 2022] *Tabi - New vertical tabs add-on that also shows related tabs*. reddit, out. de 2022. URL: https://www.reddit.com/r/firefox/comments/y35gpp/tabi_new_vertical_tabs_addon_that_also_shows/ (acesso em 21/12/2022) (citado na pg. 20).

[*Tabi - Visualize Tabs* 2022] *Tabi - Visualize Tabs*. Mozilla.org, out. de 2022. URL: https://addons.mozilla.org/en-US/firefox/addon/tabi-visualize-tabs/ (acesso em 21/12/2022) (citado na pg. 16).

[*Tabi – Visualize related tabs on Firefox, made with Svelte | Hacker News* 2022] *Tabi – Visualize related tabs on Firefox, made with Svelte | Hacker News*. Ycombinator.com, 2022. URL: https://news.ycombinator.com/item?id=33194385 (acesso em 21/12/2022) (citado na pg. 21).

[*Tabs Outliner* 2022] *Tabs Outliner*. chrome.google.com. URL: https://chrome.google.com/webstore/detail/tabs-outliner/eggkanocgddhmamlbiijnphhppkpkmkl?hl=en (acesso em 21/12/2022) (citado na pg. 9).

[*Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* 2020] *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* Tailwindcss.com, nov. de 2020. URL: https://tailwindcss.com/ (acesso em 21/12/2022) (citado nas pgs. 19, 29).

[*Toby for Chrome* s.d.] *Toby for Chrome*. chrome.google.com. URL: https://chrome.google.com/webstore/detail/toby-for-chrome/hddnkoipeenegfoeaoibdmnaalmgkpip?hl=en (citado nas pgs. 7, 8).

[*Tree Style Tab* 2007] *Tree Style Tab*. Mozilla.org, out. de 2007. URL: https://addons.mozilla.org/en-US/firefox/addon/tree-style-tab/ (acesso em 21/12/2022) (citado na pg. 9).

[*Vivaldi Browser Features | Everything' s an option* 2015] *Vivaldi Browser Features | Everything' s an option*. Vivaldi Browser, 2015. URL: https://vivaldi.com/features/ (acesso em 21/12/2022) (citado nas pgs. 19, 27).

[*Workona | The work organizer for the browser* 2022] *Workona | The work organizer for the browser*. Workona, 2022. URL: https://workona.com/ (acesso em 21/12/2022) (citado nas pgs. 7, 8).

[Yushkina 2022] Yana Yushkina. *Finding answers gets better with Chrome*. Google, fev. de 2022. URL: https://blog.google/products/chrome/finding-answers-gets-better-chrome/ (acesso em 23/10/2022) (citado nas pgs. 11, 30).