UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GIULIA BONASPETTI MARTINS

Melhoria da experiência de usuário no Tracim

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Profa. Dra. Renata de Matos Galante

Porto Alegre
2022

AGRADECIMENTOS

Antes de tudo, gostaria de agradecer profundamente a toda a empresa que me recebeu tão bem com todo seu acolhimento, pela oportunidade de poder e igualmente agradeço a todas as pessoas que contribuíram para o sucesso do meu trabalho.

Em particular, gostaria de agradecer a Damien Accorsi, gerente de empresa, que sempre esteve muito disponível e pronto para me ajudar, pela confiança e pelo tempo que me concedeu.

Gostaria também de agradecer aos meu tutores e orientadores durante todo o processo Côme Huguiès, Renata de Matos Galante e Sébastien Pittion pela disponibilidade, pelos conselhos e por todo o apoio. Finalmente, gostaria de agradecer por sua disponibilidade e trabalho em equipe a todos integrantes da Algoo: Basile Legal, Bastien Sevajol, Célia Margotteau, Charline Rageade, Guénaël Muller, Guillaume Metzger, Marie Bégué, Mathis Perrier, Myriam Megnigbeto, Nickol Bryson, Philippe Accorsi, Raphaël Isla, Raphaël Jakse e Sébastien Grignard.

Agradeço igualmente a todos os participantes dos testes.

RESUMO

Experiência do usuário diz respeito a usabilidade e qualidade da experiência que um usuário tem ao usar um *software*. O objetivo desse trabalho é melhorar a experiência do usuário no Tracim, desenvolvido pela empresa francesa Algoo. Tracim é um *software* sob licença livre que visa facilitar a gestão e o trabalho em equipe. Para tal fim, foi realizado o desenvolvimento *frontend* para correções de problemas já existentes e seis novas funcionalidades: busca, compartilhamento de arquivos, identificador, atualização em tempo real (*Tracim Live Messages*), notificações e menções. Todas as funcionalidades usam tecnologias recentes como React.js e Redux. Após a implementação, foram feitos testes de usuário com duas versões de Tracim para comparar e analisar se a experiência aumentou de fato. A análise é feita através de um formulário aplicado a cada participante e por uma lista de tarefas que foram cronometradas em cada versão. Por fim, são mencionados pontos de melhoria no desenvolvimento e nos testes de usuário para possíveis trabalhos futuros, esses pontos foram identificados durante o testes.

Palavras-chave: Experiência de Usuário. UX. Tracim. Javascript. Open Source. Software Livre. Frontend.

Improving the user experience in Tracim

ABSTRACT

User experience refers to the usability and quality of experience a user has when using software. The goal of this work is to improve the user experience in Tracim, developed by the French company Algoo. Tracim is a software under free license that aims to facilitate team management and work. To this end, frontend development was made to fix existing problems and to add six new features: search, file sharing, username, real time updates (Tracim Live Messages), notifications and mentions. All features use recent technologies like React.js and Redux. After implementation, we conducted user tests with two versions of Tracim to compare and analyze if the experience has indeed improved. The analysis is done through a form applied to each participant and a list of tasks that were timed in each version. Finally, points for improvement in the development and user testing are mentioned for possible future work, which were identified during the testing.

Keywords: User Experience. UX. Tracim. Javascript. Open Source. Free Software. Frontend..

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface: *interface* de Programação de Aplicações
CSS	Cascading Style Sheets
DOM	Document Object Model: Modelo de Documento por Objetos
HTML	HyperText Markup Language: Linguagem de Marcação de Hipertexto
HTTP	Hypertext Transfer Protocol
ID	Identificador
ISO	International Organization for Standardization: Organização Internacional de Normalização
REST	Representational State Transfer: Transferência Representacional de Estado
RGPD	Regulamento Geral sobre a Proteção de Dados
TAB	Tabulação
TLM	Tracim Live Message
URL	Uniform Resource Locator: Localizador Uniforme de Recursos
UX	User Experience: Experiência de usuário

SUMÁRIO

1 INTRODUÇÃO

Algoo (ALGOO, 2021a) é uma empresa especializada em aplicações *web* personalizada e produção e contribuição de *softwares* livres e *open source*. Ela existe há cerca de 6 anos e é baseada em Moirans, na França. É o principal desenvolvedor e contribuinte do Tracim (ALGOO, 2021b), o *software* é uma das principais razões da criação da empresa. Além de todos os valores ligados ao desenvolvimento livre, Algoo preza muito pela diversidade. Possuindo uma equipe com diferentes idades, gêneros e nacionalidades. O mascote da empresa foi escolhido justamente por ser um animal que vive em comunidade, um suricato. Algoo possui uma equipe técnica multidisciplinar com mais de 15 anos de experiência na área de desenvolvimento de *software*, do *backend* ao *frontend*. O desenvolvimento *backend* é principalmente (mas não exclusivamente) em *Python*, com *Django* / *Flask* / *Turbogears*, e o desenvolvimento *frontend* em *AngulasJS* / *Javascript* / *React*.

Parte do trabalho aqui descrito foi realizada durante a pandemia da COVID-19. Neste contexto de trabalho remoto obrigatório para quase todas as empresas na França, Algoo pensou sobre a melhor maneira de colaborar e facilitar o trabalho de todos, a partir desta reflexão nasceu o serviço de videoconferência Visiocall (ALGOO, 2021c).

A videoconferência é uma ferramenta essencial para poder trabalhar remotamente de forma eficiente e reativa, a vantagem da Visiocall é que ela não requer a compra de uma licença nem a criação de uma conta pessoal, é gratuita e tem uma politica clara e transparente sobre os dados pessoais dos usuários. É uma distribuição de uma instância de Jitsi Meet (JITSI, 2022), compatível com o RGPD, hospedada na França e baseada exclusivamente em *software* livre. Foi então a ferramenta usada na empresa durante o período de trabalho remoto, além do próprio Tracim.

Algoo acredita que uma experiência de usuário bem sucedida é o resultado de uma mistura inteligente de ergonomia (simplicidade, conforto, produtividade), riqueza funcional e estética. Sendo assim adapta as aplicações desenvolvidas ao perfil dos usuários alvo. A experiência de usuário, também conhecida por *UX* (NORMAN; MILLER; HENDERSON, 1995), é um conceito de usabilidade do *software*. Refere-se à qualidade da experiência que o usuário tem quando utiliza uma plataforma.

Este trabalho mostra uma perspectiva geral, falando tanto sobre as análises subjetivas e objetivas da *UX* quanto sobre detalhes técnicos do *software*, incluindo as tecnologias que foram utilizadas e a arquitetura de código, e de seus desafios de implementação. Os

resultados apresentados foram desenvolvidos durante um estágio realizado na empresa Algoo.

Sendo assim, o objetivo principal foi melhorar a experiência do usuário no *software* desenvolvido pela empresa, Tracim (ALGOO, 2021b). Várias soluções diferentes foram analisadas e desenvolvidas para que, ao final, o conjunto dessas soluções pudesse melhorar significativamente o *software*. Sendo elas: correção de bugs, busca de conteúdos, compartilhamento de arquivos, identificador, atualizações em tempo real, notificações e menções.

O trabalho se concentra nas decisões de concepção e no desenvolvimento *frontend*, mesmo se todas as funcionalidades tiveram o apoio de uma equipe *backend* para o seu desenvolvimento. O desenvolvimento *frontend* refere-se à *interface* gráfica do projeto e as possíveis interações com o usuário.

O restante do texto está organizado da seguinte forma. O capítulo 2 versa sobre a fundamentação teórica da experiência de usuário e do desenvolvimento além das tecnologias utilizadas. O capítulo 3 apresenta em detalhe o *software* Tracim. O capítulo 4 aborda todos os desenvolvimentos realizados. E por fim, o capítulo 5 apresenta os testes feitos com usuários para validar a melhoria de experiência e seus resultados.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo apresenta toda a fundamentação teórica utilizada durante o trabalho, que pode ser dividida em dois principais pilares: fundamentação necessária para a concepção da experiência de usuário e para o desenvolvimento de *software*. Cada parte será apresentada, respectivamente, nas seções que seguem.

2.1 Experiência de usuário

A experiência de usuário é um conceito de usabilidade do *software*. Refere-se à qualidade da experiência que o usuário tem quando utiliza uma plataforma. O termo *User Experience* foi popularizado por Donald Norman em meados dos anos 90, onde ele buscava estender o estudo da interação homem-produto além da eficácia incluindo fatores emocionais e comportamentais (NORMAN; MILLER; HENDERSON, 1995).

A primeira vez que a ISO publicou uma norma relacionada foi em 1999, a ISO 13407 (STANDARDIZATION, 1999), que foi cancelada e substituída pela ISO 9241-210. Atualmente ela esta na segunda edição (primeira publicada em 2010 (STANDARDIZATION, 2010) e segunda em 2019 (STANDARDIZATION, 2019)) e fornece alguns requisitos e recomendações.

Contudo, a norma não possui uma cobertura detalhada dos métodos, somente fornece uma visão geral de ergonomia e da modelização com foco no ser humano. Na própria introdução, os autores comentam que as boas práticas podem variar muito segundo a complexidade do produto e seus objetivos. Nessa norma, além dos fatores emocionais e comportamentais que citamos anteriormente, nós podemos incluir na experiência do usuário as crenças, preferências, realizações, percepções, respostas físicas e psicológicas que ocorrem durante o uso do produto.

Entretanto, o tópico ainda é bem subjetivo e as medidas do tipo “quantos cliques foram necessários” ou “quanto tempo o usuário demorou para realizar a tarefa” ajudam mas seguidamente possuem menos valor do que o retorno direto de como o usuário se sentiu.

Sendo assim, são utilizados três fatores principais para os testes de *UX*:

- análise da eficácia (usuário conseguiu ou não realizar a tarefa);
- análise de eficiência/performance (fatores mensuráveis como o número de cliques

ou o tempo para realizar a tarefa);

- análise da satisfações e/ou frustrações do usuário.

A ISO 9241-210 também lista três fatores que influenciam na experiência: o sistema, o usuário e o contexto de uso. Sistema é o conjunto de um ou mais elementos com o qual o usuário interage para atingir um propósito específico, pode ser um *hardware*, um *software*, um serviço ou até mesmo uma pessoa. Usuário é o individuo com um sistema buscando alcançar um objetivo. Por fim, contexto de uso é a combinação do ambiente (técnico, físico, social, cultural e organizacional), dos recursos disponíveis e dos usuário com as tarefas e seus objetivos.

Um exemplo desses fatores é a experiência de um caixa de um supermercado:

- sistema: o caixa (tanto a parte física, quanto a parte digital), o leitor de códigos, os produtos que serão comprados, o código de barras de cada produto, etc;
- usuário: o operador de caixa;
- contexto de uso: o quanto o operador esta confortável com o sistema (ambiente técnico), se ele possui uma cadeira para não ficar cansado (ambiente físico), a simpatia do cliente à frente (ambiente social), todos os produtos estão registrados e pesados (recursos disponíveis), etc.

No capítulo 5 detalharemos os testes realizados para esse trabalho, retomando os três fatores principais de testes e os três fatores de influência. Todavia, antecipando, foram utilizados um formulário personalizado e a cronometragem do tempo de realização de cada tarefa.

2.2 Desenvolvimento de *software*

Nessa seção é apresentada a fundamentação teórica para o desenvolvimento de *software* e as respectivas tecnologias utilizadas. Começando pela gestão do código na subseção 2.2.1, após falando do código em si na subseção 2.2.2 e por último explicando os testes de desenvolvimento na subseção 2.2.3.

2.2.1 GIT

O GIT é um *software* de gerenciamento de versões descentralizado. Na Algoo, GitHub[1] é o serviço de hospedagem e gerenciamento de desenvolvimento *web* usado. Trabalhando com diversas *branches*, *merge requests*, *issues* e *labels*. Como o Tracim é *open source*, todo desenvolvimento desse trabalho pode ser visto no GitHub @gbonaspetti (GITHUB, 2022a).

2.2.2 Programação *frontend*

Para facilitar o desenvolvimento, Tracim usa algumas ferramentas de programação. A principal delas é a biblioteca JavaScript **React.js** (META, Plataforms, 2022) que facilita a programação de aplicações de página única e nos permite dividir o código em componentes dependentes do estado. O código é portanto escrito em JSX, que é uma extensão da linguagem JavaScript, parecendo e estruturando o código muito semelhante ao HTML. Os componentes têm uma função de renderização, chamada *render* que se encarrega de exibir a parte visual.

Para a programação CSS usamos o pré-processador **Stylus** (STYLUS, 2022) que nos permite criar variáveis e hierarquias claras e fáceis de usar em nosso CSS, e também tivemos o suporte da estrutura **Bootstrap** (BOOTSTRAP, 2022) que simplifica algumas coisas mais complexas e padroniza outras. Por exemplo, Tracim possui um cabeçalho com vários botões, ao diminuir a tela é necessário agrupar esse botões para não causar uma barra de rolagem horizontal. Bootstrap padroniza os tamanhos de tela em *xs* (*extra small* - extra pequena), *sm* (*small* - pequena), *md* (*medium* - média), *lg* (*large* - grande), *xl* (*extra large* - extra grande) e possui uma funcionalidade de barra de navegação, chamada *navbar*, em que podemos definir facilmente em qual tamanho de tela deseja-se unir os botões um menu. Na Figura 2.1 podemos ver o cabeçalho em uma tela média com todos os botões amostra, enquanto na Figura 2.2 temos a versão em uma tela extra pequena, onde vemos um único botão de menu no lugar e na Figura 2.3 temos a versão com o menu aberto.

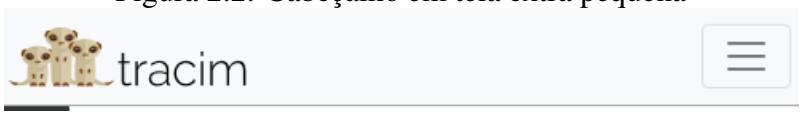
Para simplificar a comunicação com o *backend* e variáveis que seriam acessadas por múltiplos componentes, usamos a biblioteca *open source* **Redux** (ABRAMOV, Dan, 2022), que faz a gestão do estado da aplicação *web*, e o *framework* de *backend* **Swagger**

[1] https://github.com/

Figura 2.1: Cabeçalho em tela média

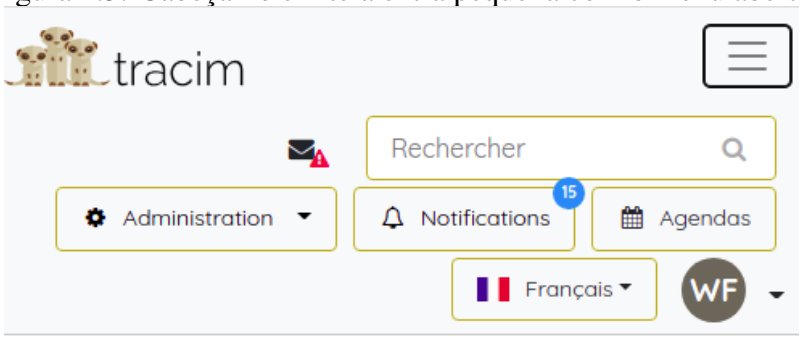
Fonte: o autor

Figura 2.2: Cabeçalho em tela extra pequena



Fonte: o autor

Figura 2.3: Cabeçalho em tela extra pequena com o menu aberto



Fonte: o autor

UI (SMARTBEAR, 2021), que nos ajuda a projetar e visualizar as possíveis chamadas para a API de *backend* que podem ser feitas em nossa aplicação.

2.2.3 Testes de desenvolvimento

O *frontend* Tracim tem dois tipos de testes: testes de integração e testes unitários.

Para os testes de integração usamos **Cypress** (IO, Cypress, 2022), que consiste em um *software* livre, um simulador de testes instalado localmente e um serviço de painel de instrumentos para gravação dos testes. Isto torna mais rápido, mais fácil e mais confiável a realização de testes.

Com esta ferramenta, pode-se simular um usuário passando por todas as características do Tracim para ajudar a ver que uma nova característica estava funcionando bem e não causando impacto nas outras.

Os testes unitários são feitos usando **Mocha** (OPENJS, 2022), que é um *framework* executado sobre o Node.js e no navegador, o que torna os testes assíncronos mais fáceis. Apesar dessa facilidade de assincronia, esse tipo de teste possui suas dificuldades, pois todas as requisições ao *backend* e seus respectivos dados devem ser simulados, o que

nem sempre é trivial.

Independentemente de quaisquer dificuldades, os testes de desenvolvimento são de extrema importância para garantir uma boa experiência do usuário, visto que precisamos ter certeza de que tudo funcionará como desejamos e que ao acrescentar novas funcionalidades tudo funciona como o usuário está acostumado.

3 TRACIM

O objetivo desse capítulo é apresentar com mais detalhes o *software* Tracim. Começando pela sua historia e uma visão geral na seção 3.1, seguindo para as licenças de *software* na seção 3.2, depois as contribuições externas na seção 3.3, abordando após na seção 3.4 o público alvo observado e finalizando com a arquitetura do código *frontend* na seção 3.5.

3.1 Visão Geral

Tracim (ALGOO, 2021b) é um *software* livre e 100% *open source*. Uma plataforma *extranet* e *intranet* para trocas e trabalho em equipe que se adapta às necessidades do usuário e visa facilitar a colaboração. O *software* é simples e intuitivo, traduzido em 4 idiomas diferentes (em breve mais). As informações e trocas são centralizadas em um único lugar e disponíveis 24 horas por dia.

Seu nome vem de *TRACability IMproved* e tem como quatro principais eixos de evolução: colaboração de equipe, gestão do conhecimento, animação de comunidades e produção de documentos. Para conquistar tais eixos, possui muitas funcionalidades, tais como:

- histórico de versões para cada documento;
- atividades recentes;
- separação em espaços;
- agenda compartilhada;
- notificações;
- etc.

O *software* é apresentado na forma de uma aplicação *web* que também oferece *interface*s CalDav (CALCONNECT, 2022) e WebDav (WHITEHEAD, 2010) para uma ótima integração.

3.2 A licença do Tracim

A filosofia do *software* livre é muito importante para a empresa e Tracim não possui apenas uma licença, mas várias:

- GPLv3 (FREE SOFTWARE FOUNDATION, 2007a): presente na agenda que usamos, pois ela depende do CalDav (CALCONNECT, 2022) que usa esta licença;
- LGPL (FREE SOFTWARE FOUNDATION, 2007b): em parte do código *frontend*;
- MIT (OPEN SOURCE INITIATIVE, 2022): em todo o código *backend* e em parte o *frontend*.

A licença GPLv3 (Licença Pública Geral GNU versão 3) é uma licença *copyleft* enquanto MIT (Licença criada pelo Instituto de Tecnologia de Massachusetts) é uma licença permissiva, LGPL (Licença Pública Geral Menor GNU) seria um meio termo entre as duas. A principal diferença entre elas é que para as licenças GPL todos trabalhos derivados só podem ser distribuídos se utilizarem a mesma licença. Em contrapartida LGPL permite vincular o programa a código não-LGPL sem revogar a licença, incluindo *softwares* proprietários, desde que estejam disponíveis em bibliotecas. Assim, torna-se possível para um programador que deseja fazer *software* proprietário utilizar algumas das ferramentas do mundo livre sem forçar seu *software* a ser livre também.

Por fim, a MIT que tem como única exigência manter o aviso de *copyright* e uma cópia da licença em todas as cópias do *software*.

3.3 Contribuições

O repositório Tracim está no GitHub (GITHUB, 2022b) e hoje conta com 28 colaboradores que são - ou foram - em sua maioria funcionários da Algoo. As contribuições são mais raras pelas seguintes razões:

- Tracim é um projeto grande e é necessário ter conhecimento técnico para mudá-lo;
- o objetivo de aumentar o número de colaboradores no Tracim não é uma prioridade da empresa no momento, mas sim desenvolver uma comunidade de usuários.

Boa parte das nossas contribuições realmente externas à empresa são feitas na tradução da aplicação, pois utilizamos a plataforma Weblate (ČIHAŘ, Michal, 2012-2022) que permite a qualquer pessoas contribuir, não restringindo apenas à desenvolvedores.

3.4 Público alvo

Trabalhando na área da experiência de usuário é muito importante conhecer o público alvo do *software*. Entretanto, tal especificação nunca foi feita para o Tracim, sendo assim, nesse trabalho, reunimos as principais características ditas pelo responsável produto e implícitas na *interface* para guiar-nos.

Ao realizar a concepção de *interface* é preciso se concentrar em três perfis: o administrador de sistemas, o gestor da instância e o usuário de base. Eles serão apresentados com mais detalhes na subseções a seguir, respectivamente.

3.4.1 Administrador de sistemas

O administrador de sistemas é o responsável pela configuração da instância. Por mais que não seja necessário levar esse perfil em conta para a *interface* visual, ele deve ser considerado quando criamos a "*interface*" dos comandos. Isso significa que, toda a documentação e coerência de comandos que podem ser usados para a configuração.

O administrador de sistema é um perfil que trabalha com a informática e que conhece todos o jargões, *interface*s desenhadas para ele devem ser técnicas e diretas. Ele precisa ter conhecimento de todas as opções disponíveis, porém a opção por *default* deve ser a opção mais utilizada para otimizar seu tempo.

3.4.2 Gestor da instância

O gestor da instância é o responsável pela gestão dos usuários e dos espaços dentro do Tracim. É um perfil minimamente confortável com a informática, mesmo se normalmente não trabalha diretamente com ela.

Para a *interface* da gestão nos preocupamos menos com a estética e mais com a praticidade. A *interface* deve ser simples e conter todas informações necessárias para as tarefas recorrentes. Opções mais avançadas e/ou pouco utilizadas não precisam ser tão evidentes. Contudo, a opção por *default* deve ser a opção mais cautelosa para priorizar a segurança.

3.4.3 Usuário de base

Para o restante das funcionalidades do Tracim, o perfil pensado não está acostumado com informática. Sendo assim, a *interface* deve ser minimalista, atraente e sobretudo intuitiva. Deseja-se que o usuário tenha vontade de usar a plataforma.

A opção por *default* deve ser pensada de acordo com a gravidade em caso de erro. Para funcionalidades com maiores riscos escolhe-se a opção mais segura, enquanto para as de menores riscos escolhe-se a opção mais utilizada.

3.5 A arquitetura do *frontend* do Tracim

Neste trabalho, alguns termos mais específicos da arquitetura do *frontend* Tracim serão mencionados, portanto, esta subseção explicará brevemente como o código está organizado.

Primeiro de tudo, o *frontend* é dividido em três partes principais: os *apps*, o aplicativo principal do *frontend* e a biblioteca *frontend_lib*.

Os *apps* são aplicações para fazer e gerenciar funcionalidades específicas. Todos eles são aplicações React autônomas que têm seus próprios repositórios e podem ser livremente adicionados ou removidos do Tracim. Eles atuam como *plugins* e são de dois tipos: *appFeature* e *appFullScreen*.

- *appFeature* são aplicações ligadas a um tipo de conteúdo da plataforma, elas compartilham um design semelhante com a ideia de uma janela em cima da aplicação;
- *appFullScreen* são aplicações relacionadas à parte de gerenciamento da plataforma e de seus usuários, tendo um design que dá a impressão de estar totalmente integrado dentro do Tracim.

O aplicativo principal *frontend* é quem gerencia os principais elementos do Tracim, como a barra lateral e o cabeçalho, assim como a gestão das aplicações. É a única aplicação que utiliza Redux.

A biblioteca *frontend_lib* é um conjunto de componentes e funções que podem ser importados por qualquer aplicação, tendo assim os componentes mais genéricos possíveis.

4 MELHORIA DA EXPERIÊNCIA DE USUÁRIO

Nesse capítulo é apresentado a metodologia utilizada, seção 4.1, e o desenvolvimento feito, seção 4.2, para melhorar a experiência de usuário no Tracim. Ambos tratam da correção de problemas já existentes e das funcionalidades busca, compartilhamento de arquivos, identificador, *Tracim Live Messages*, notificações e menções.

4.1 Levantamento de requisitos e metodologia

Durante as fases de resolução de *bugs* e melhorias, alguns dos problemas resolvidos já haviam sido detectados pela equipe Algoo, visto que eles utilizam o *software* na sua comunicação *extranet*/*intranet*. Tais problemas foram avaliados e eventualmente tratados de acordo com sua importância. No decorrer do trabalho, foi possível detectar e corrigir novos *bugs* e melhorias que ainda não haviam sido identificados.

Outro momento em que a resolução de *bugs* é importante e que somos confrontados diretamente com decisões de melhoria de experiência de usuário são as fases de integração. Após fazer testes intensivos em equipe, é necessário decidir quais *bugs* resolver e suas prioridades.

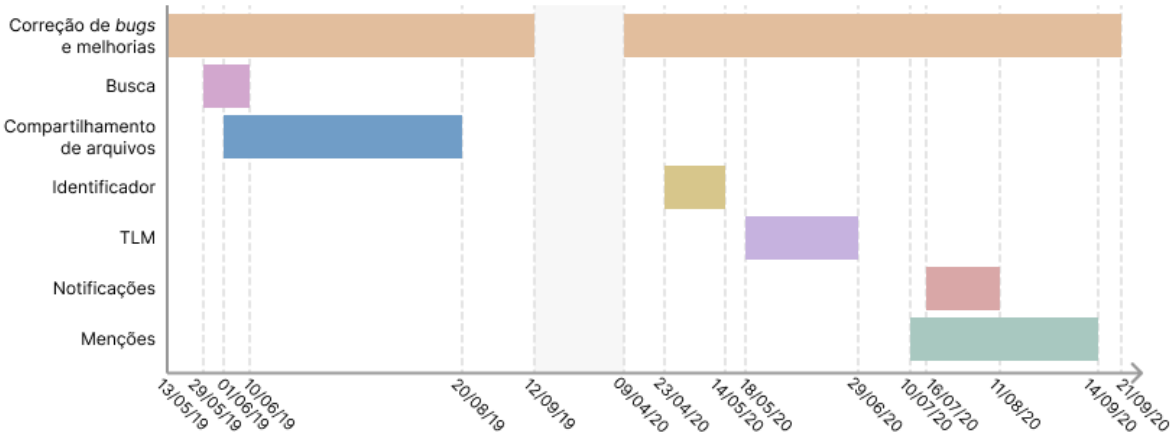
Para as funcionalidades que notamos que faltavam no *software*, o desenvolvimento é organizado com a famosa expressão "dividir para conquistar" em mente. Temos o problema em programação base, i.e. tudo que é necessário para uma primeira versão super simples, e vários *issues* de melhoria.

Esses desenvolvimentos de melhoria podem ser implementados em qualquer ordem, então são priorizados para garantir que mudanças mais drásticas para a experiência de usuário sejam feitas primeiro. Além do mais, cada *issue* deve ser o mais simples e direto possível para permitir a realização em paralelo.

Na Figura 4.1, podemos ver uma simplificação da divisão do tempo. O trabalho aconteceu em dois momentos: de maio à setembro de 2019 e de abril à setembro de 2020. Correções de *bugs* e melhorias ocorreram ao longo de todo o trabalho, com mais ou menos intensidade segundo o período. As funcionalidades de busca e compartilhamento de arquivos foram realizadas no primeiro momento e as demais no segundo.

Apesar da Figura 4.1 dar uma ideia de divisão do tempo, é importante ressaltar que isso não representa a dificuldade e complexidade de cada desenvolvimento. Tanto o nível do desenvolvedor no momento dado quanto a facilidade com a arquitetura do código

Figura 4.1: Divisão do tempo entre desenvolvimentos



Fonte: o autor

são fatores chave que influenciam no tempo.

Além disso, cada funcionalidade possuía uma equipe diferente, enquanto o compartilhamento de arquivos foi feito por apenas duas pessoas (uma no desenvolvimento *frontend* e outra no *backend*), os TLMs foram feitos por seis desenvolvedores diferentes. Igualmente, a Figura 4.1 leva em conta unicamente o tempo de desenvolvimento. Apesar do trabalho ter sido realizado no contexto de um estágio, a concepção e especificação foi feita em equipe antes de começar a programação em si, com diversas reuniões e tomada de decisões. Cabe ressaltar que a autora participou ativamente no processo de tomada de decisão.

4.2 Desenvolvimento

Esta seção descreve com um pouco mais em detalhe sobre os desenvolvimentos *frontend* realizados nesse trabalho. Ela está dividida em 7 subseções: correção de *bugs* e melhorias (4.2.1), busca (4.2.2), compartilhamento de arquivos (4.2.3), identificador (4.2.4), mensagens em tempo real (4.2.5), notificações (4.2.6) e menções (4.2.7). Sendo a primeira um pouco mais geral, incluindo alguns exemplos, e as demais são funcionalidades específicas realizadas.

4.2.1 Correção de *bugs* e melhorias

Tracim é um grande projeto que já dispõe de alguns anos, então possui muitos *bugs* já detectados que ainda não foram resolvidos e melhorias para serem feitas. Para garantir uma boa experiência do usuário, antes de acrescentar novas funcionalidades, é necessário que a aplicação funcione bem sem elas. Com isto em mente e para obter um primeiro contato com o código Tracim, a primeira coisa que foi feita nesse trabalho foi corrigir alguns *bugs* de *UX* e melhorias que apresentam uma evolução na experiência do usuário também.

Não entraremos em detalhe em cada *bug* corrigido, pois foram em torno de 152 *issues* (entre *bugs* e funcionalidades) desenvolvidos durante essa fase. Primeiro lidando com alguns *bugs* simples como uma tradução incorreta ou um problema de design responsivo. Após, percorrendo a aplicação Tracim para ver possíveis problemas de *UX* não detectados e classificá-los. Em seguida, retornando à correção, porém se concentrando na importância dos *bugs* e melhorias a partir de uma perspectiva de impacto *UX* e não em sua dificuldade de desenvolvimento.

Para resolver os *bugs* e melhorias, usamos o seguinte fluxo de trabalho:

- buscar se o sujeito já existe nos *issues* do repositório GitHub do Tracim;
    - se existir: atualizar o *issue*, se necessário, e atribuir-se à ele;
    - se não existir: criar o *issue*, colocá-lo no projeto certo e atribuir-se à ele.
- seguir o quadro *Kanban* do projeto no GitHub de acordo com o estado atual do desenvolvimento (*To Do - In Progress - To/under review - Need test on Branch - Under test on branch - Need test on develop - To close - Done*).

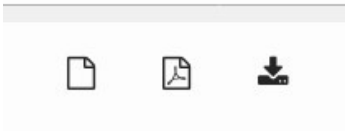
Nos itens a seguir, a título de exemplo, serão apresentados alguns dos *bugs* tratados e melhorias aplicadas. Destaca-se também com qual público alvo em mente o trabalho foi feito.

4.2.1.1 Cor de ícone incoerente

Um dos primeiros *bugs* corrigidos foi bem simples. Na *interface* havia uma regressão que fazia com que alguns dos botões que antes eram cinza escuro, como podemos ver na Figura 4.2, agora fossem azuis como na Figura 4.3 (ACCORSI, Philippe, 2019).

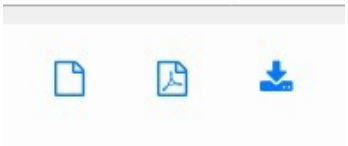
O diagnostico do problema foi que os botões eram elementos HTML do tipo ân-

Figura 4.2: Botões do arquivo antes do *bug*



Fonte: https://github.com/tracim/tracim/issues/1712

Figura 4.3: Botões do arquivo depois do *bug*



Fonte: https://github.com/tracim/tracim/issues/1712

cora, ou seja `<a>`, e a regra CSS que dava a cor escura para todos elementos desse tipo havia sido apagada para permitir *links* de outras cores através de classes. Sendo assim, a cor usada era a cor padrão para esse tipo de elemento na estrutura Bootstrap.

Para corrigi-lo bastou adicionar a regra `color` com a cor certa da fonte na classe específica desses botões.

Em termos de experiência de usuário, a correção desse *bug* permite uma *interface* coerente, de forma que um usuário de base não precise refletir muito do porque esses ícones são azuis e se ele pode clicar neles. Um usuário que esta acostumado com *links* em textos na *web*, por exemplo, poderia imaginar que ao clicar ele seria redirecionado para outra página.

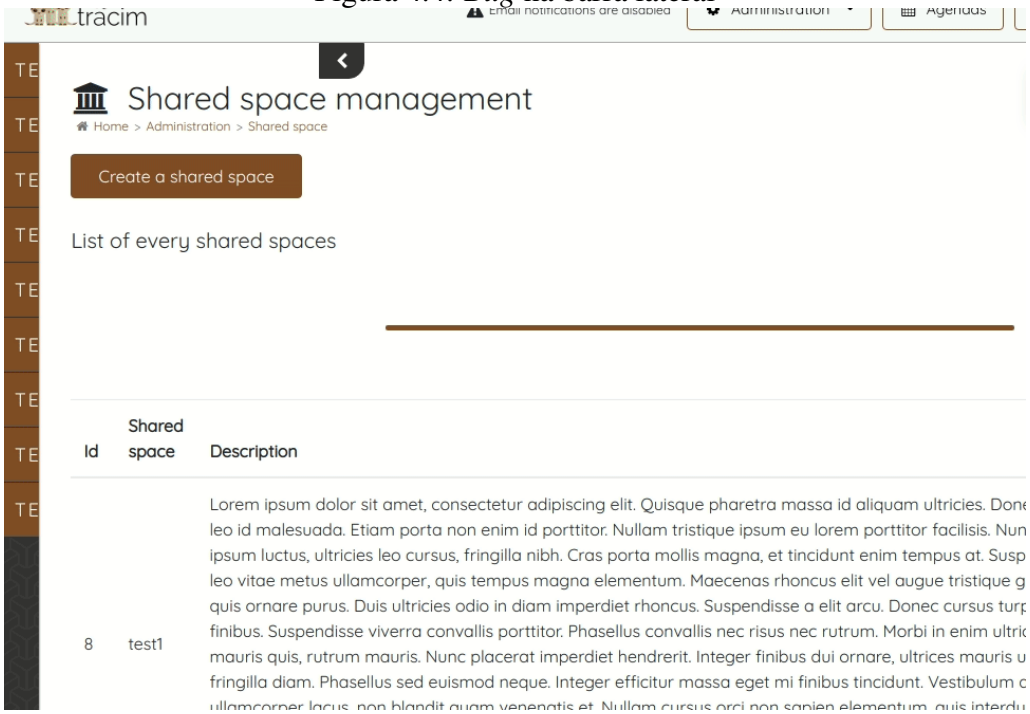
4.2.1.2 Problema na barra lateral quando o espaço tem uma grande descrição

Ao adicionar uma grande descrição a um espaço, um gestor de instância havia muita dificuldade de navegação pois sua barra lateral desaparecia em meio a tela de administração (MARTINS, Giulia B., 2019c), como podemos ver na Figura 4.4. Essa dificuldade piorava significativamente a experiência. Além do mais, ter toda a descrição exposta na página de administração não é útil para o gestor e pode aumentar o tempo que o usuário leva para realizar tarefas simples, pois adiciona mais barra de rolagem e mais textos na tela.

Para resolver esse problema e voltar para um *layout* como o apresentado na Figura 4.5, foi adicionado uma classe que permite customizações CSS a cada coluna da tabela da página de administração.

No arquivo de estilo, o texto foi limitado à somente uma linha, foi definido uma

Figura 4.4: *Bug* na barra lateral



Fonte: https://user-images.githubusercontent.com/35422692/58108073-6eebf800-7beb-11e9-85af-6f8a6bae7592.gif

porcentagem de largura da tabela que cada coluna ocuparia e, caso o texto exceda essa largura, ele é cortado e reticências são colocadas no lugar do restante. O texto continua visível na *tooltip* da descrição.

4.2.1.3 Gestão da tecla Tab na edição de documentos

Durante a edição de um documento, se um usuário apertasse a tecla Tab, o foco mudava para o botão anular (SEVAJOL, Bastien; HUGUIÈS, Côme, 2019). Isso fez com que alguns usuários perdessem todas modificações que haviam feito até lá.

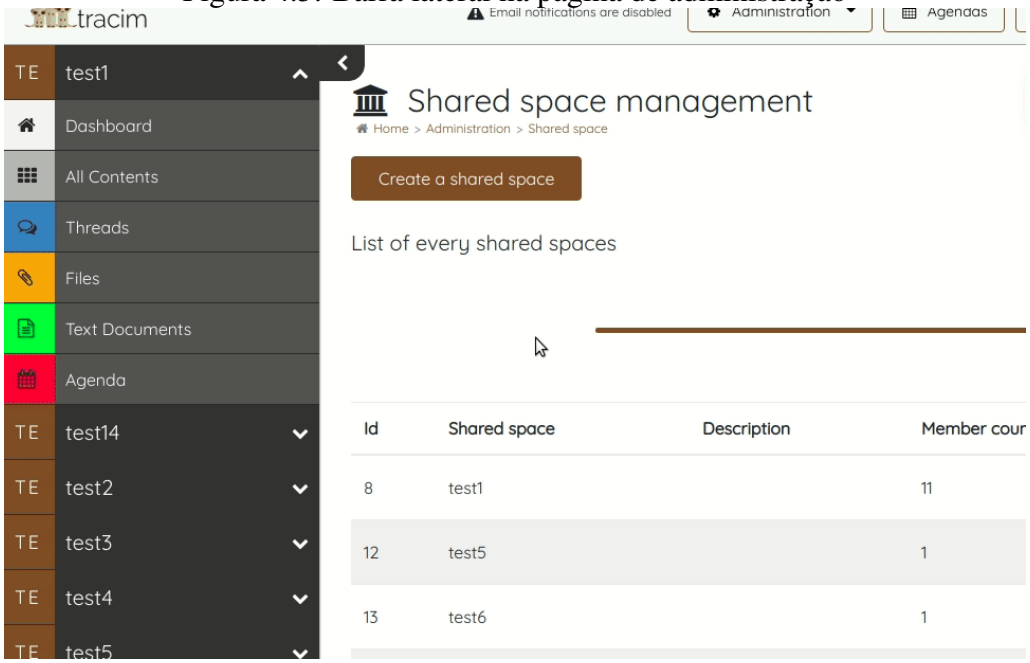
A solução para o problema foi adicionar atributos `tabindex` aos botões para determinar a ordem em que eles serão focalizados.

4.2.1.4 Tornar o menu dos aplicativos totalmente clicável

Cada aplicativo é dividido em duas partes: o conteúdo e o histórico de versões. Tais partes são separadas por um menu cinza, como vemos na Figura 4.6. Esse histórico pode ser escondido caso o usuário queira mais espaço para ver o conteúdo.

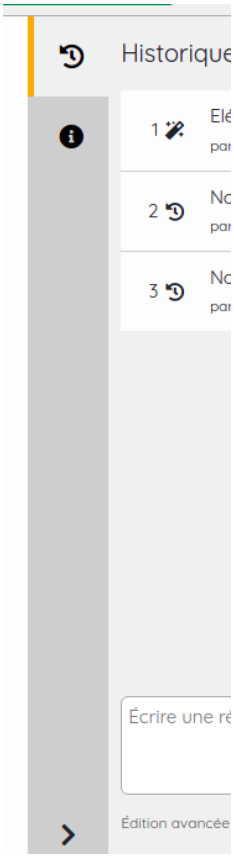
Antes, a única forma de esconder era clicando no ícone de seta no canto inferior do menu. Para melhorar essa experiência parar um usuário de base, tornou-se todo o menu

Figura 4.5: Barra lateral na página de administração



Fonte: https://user-images.githubusercontent.com/35422692/58108073-6eebf800-7beb-11e9-85af-6f8a6bae7592.gif

Figura 4.6: Menu interno dos aplicativos



Fonte: o autor

clicável (MARTINS, Giulia B., 2019b).

4.2.1.5 Ocultar a função de arquivamento

Ao deletar um conteúdo no Tracim, ele é marcado como deletado, é escondido da *interface*, porém não é de fato apagado da base de dados, podendo assim restaurar o conteúdo a qualquer momento. Outra funcionalidade era a de arquivar o conteúdo, ele era então marcado como arquivado, escondido da *interface* e podia ser restaurado.

Como esse breve resumo mostrou, ambas funcionalidades eram extremamente parecidas e isso causava confusão no nossos usuários de base. Como melhoria, decidiu-se ocultar a possibilidade de arquivamento (MARTINS, Giulia B., 2019a).

Foi escolhido o arquivamento ao invés da supressão, pois o usuário tende a se sentir mais seguro podendo apagar coisas numa plataforma em que ele faz publicações. Também, foi um pedido do responsável do produto que a funcionalidade fosse apenas ocultada e não apagada, pois ele planejava reativá-la assim que fosse possível filtrar por documentos arquivados.

4.2.2 Busca

No livro "Não me faça pensar" (KRUG, 2014), referência para experiência do usuário, Steven Kurg cita que uma das características obrigatórias na *web* é a busca.

Uma instância de Tracim pode ter muitos documentos, mas não havia uma maneira rápida e mais automática de encontrá-los. Sendo assim, foi implementada uma busca simples e uma busca avançada usando ElasticSearch (ELASTICSEARCH, 2022) no *backend*.

Inicialmente a única diferença entre as duas buscas era no lado do servidor que, ao usar ElasticSearch, buscava pelo termo dentro do conteúdo e não só no nome e que possuía uma ponderação dos resultados para mostrar os mais relevantes no início.

Algum tempo mais tarde, em março de 2021, a parte visual também mudou. A busca avançada passou a ter mais detalhes sobre cada resultado e foi adicionada a possibilidade de filtrá-los. Contudo, vale ressaltar que os testes feitos nesse trabalho são sob a versão antes dessa mudança e utilizam a busca simples.

A quantidade de conteúdos presentes no banco de dados das instâncias de teste não justificaria usar uma busca avançada. Nas tarefas cuja busca poderia ser relevante havia

no máximo três itens de resultado e a ponderação não mudaria muita coisa na experiência do usuário.

No momento do desenvolvimento *frontend*, a parte de *design* já havia sido feita e apresentada através de protótipos de *interface* e a parte de *backend* também já havia sido desenvolvida. Foram desenvolvidos:

- campo de busca;
- página de resultados com paginação;
- busca via parâmetros na URL;
- testes funcionais.

Citando mais em detalhe, a busca é realizada na aplicação principal *frontend* e portanto utiliza Redux para salvar as seguintes informações:

- *currentNumberPage*: o número atual de páginas carregadas;
- *numberResultsByPage*: a quantidade de elementos por página;
- *currentNumberSearchResults*: a quantidade elementos exibidos;
- *searchedKeywords*: os termos buscados;
- *resultsList*: a lista de resultados.

E para atualizar essas informações, nós temos sete ações possíveis:

- *resetSearch*: reinicializa as informações do Redux para "apagar" a antiga busca;
- *setSearchResultsList*: define uma nova lista de resultados;
- *appendSearchResultsList*: adiciona novos elementos na lista de resultados já existente;
- *setCurrentNumberSearchResults*: define uma nova quantidade elementos exibidos;
- *setSearchedKeywords*: define novos termos buscados;
- *setNumberResultsByPage*: define uma nova quantidade de elementos por página;
- *setCurrentNumberPage*: define um novo número atual de páginas carregadas.

Para pedir dados ao *backend*, nós utilizamos uma API REST com os seguintes dados:

- *show_archived*: para saber se o *backend* deve enviar conteúdos arquivados;
- *show_deleted*: para saber se o *backend* deve enviar conteúdos deletados;
- *show_active*: para saber se o *backend* deve enviar conteúdos ativos;
- *content_types*: para definir quais tipos de conteúdo serão recebidos;

- *search_string*: os termos pesquisados;
- *page_nb*: o número atual da página;
- *size*: o número de elementos desejados por página.

Como no momento do desenvolvimento os conceitos de conteúdo arquivado e conteúdo não-ativo não eram explorados pelo *frontend*, esses dados foram sempre 0 (não mostra conteúdo arquivados) e 1 (mostra todos conteúdos ativos). Além disso, não havia filtros para a pesquisa, então o dado relacionado a arquivos deletados era sempre 0 e o dado relacionado aos tipos de conteúdo era igual a todos os tipos disponíveis na instancia. O número de elementos por página também é fixo e definido por uma variável de configuração.

Existem duas formas de realizar uma busca, através do campo mostrado na Figura 4.7 ou através da URL. No campo de busca, a cada nova busca é enviado ao *backend* uma requisição passando os termos fornecidos pelo usuário e o número de página 1, pois deduz-se que é uma nova pesquisa a cada vez que esse campo é utilizado.

Figura 4.7: Campo de busca

Fonte: o autor

Para a busca via URL foi criada uma rota específica com React Router (REMIX, 2022) para acessar diretamente a página de resultados desejada. Essa opção foi outra forma de melhorar a experiência, assim o usuário poderia compartilhar exatamente o que teria buscado ao copiar e colar sua URL para outro usuário.

Para esse modo, todos os dados que são enviados para o backend e foram citados anteriormente estão na URL, até mesmo os fixos. Isso é feito atualizando a URL a cada ação do usuário, por exemplo, se realizamos uma busca por "teste" usando o campo de busca na instância `giulia.tracim.fr`, que está configurada para haver apenas discussões como conteúdo (*thread*) e 10 elementos por página, nós seremos redirecionados para a URL:

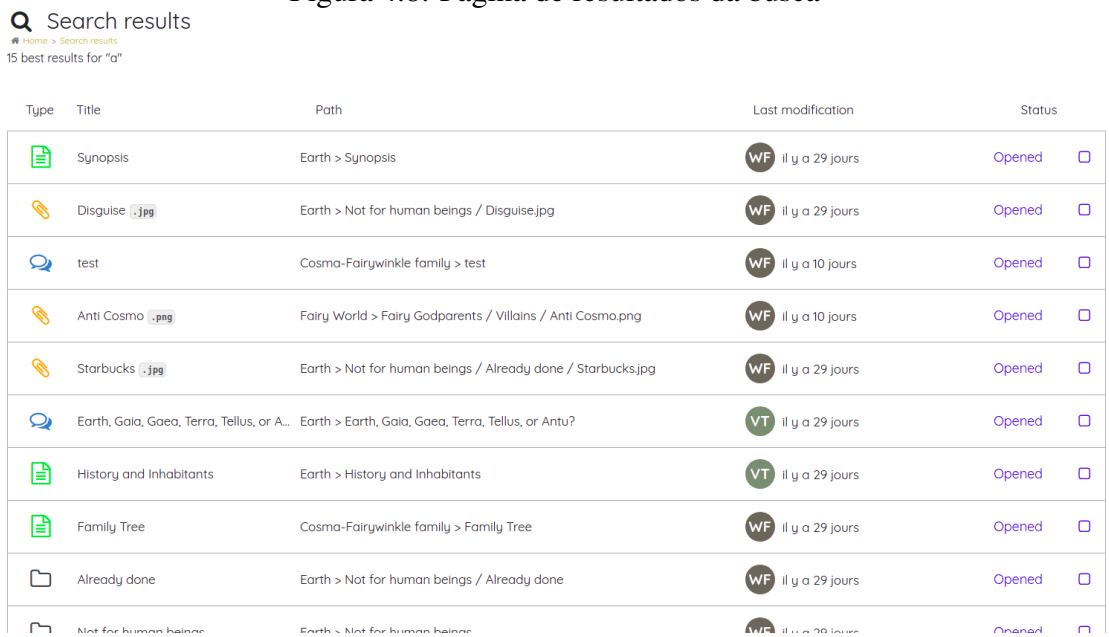
`https://giulia.tracim.fr/ui/search-result?act=1&arc=0&del=0&nr=15&p=1&q=teste&t=thread`

Ao clicar no botão para ver mais elementos, outra requisição é enviada com os mesmos dados e o número de página incrementado. Tal mudança também é refletida na URL.

Assim que uma resposta do *backend* é recebida, as respectivas ações do Redux são chamadas com os dados pertinentes.

O *design* da *interface* foi concebido para ser o mais próximo do que o usuário está acostumado em outras plataformas, mantendo a coerência com o estilo do restante de Tracim. A página de resultados contém o ícone tipo do conteúdo, o nome, o caminho para encontrá-lo, o status, quem fez a última modificação e quando, como mostrado na Figura 4.8. Clicar em um conteúdo redireciona para a URL do mesmo.

Figura 4.8: Página de resultados da busca



Fonte: o autor

4.2.3 Compartilhamento de arquivos

Como o foco principal do Tracim é o compartilhamento de conhecimentos, era lógico e necessário ter uma forma de enviar e receber arquivos de qualquer pessoa, mesmo se ela não possuísse uma conta na plataforma.

A ideia inicial era fazer uma opção de compartilhar arquivos tanto com usuários Tracim quanto com *links* públicos, mas devido à urgência de ter esta funcionalidade, decidimos implementar somente a opção de compartilhamento (*upload* e *download*) através de um *link*, protegido ou não por senha, enviado por *e-mail*. Os arquivos de *upload* são carregados em uma pasta dedicada no respectivo espaço onde o *link* foi gerado.

Para garantir mais conforto e segurança, os membros do espaço podem interrom-

per o acesso ao *download* e/ou ao *upload* a qualquer momento. Na parte de desenvolvimento, havia cerca de nove novos componentes:

- a tela de acesso de quem recebe o *link* para o *download*;
- a tela de acesso de quem recebe o *link* para *upload*;
- a tela de de finalização de um *upload*;
- a nova opção para controlar o *upload* de um arquivo, onde cada *link* é um componente React em si, também utilizado no *upload*, de modo que é fácil manter um padrão de *design*;
- a opção de criar um novo *link* de *upload* para um arquivo para o usuário do Tracim;
- a opção para controlar *upload*s na pasta de compartilhamento de um espaço;
- a opção de criar um novo *link* a partir do *upload* para um espaço.

Por uma questão de tempo e facilidade de implementação, foi decidido que inicialmente só teríamos a funcionalidade para conteúdos do tipo arquivo. Então, foi adicionado um item no menu interno da aplicação correspondente, como mostrado na Figura 4.9. Esse menu é gerado através de uma lista que contém o nome do menu, seu ícone e o componente que deve ser mostrado quando o menu está ativo.

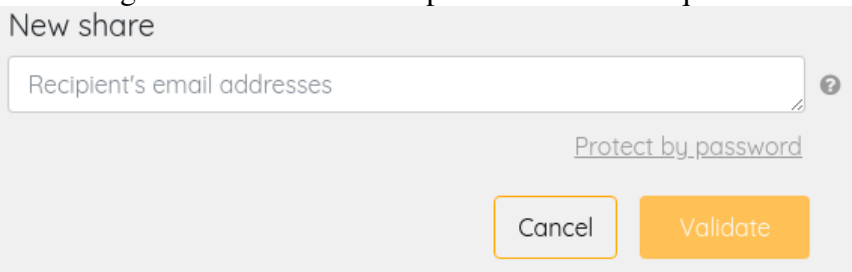
Figura 4.9: Entrada de compartilhamento no menu

Fonte: o autor

Para a criação de um *link* foi adicionado um campo de entrada texto onde um ou mais *e-mail*s devem ser colocados, como podemos ver na Figura 4.10. Esse campo é verificado assim que o usuário tenta validar o formulário.

Primeiramente divide-se o texto de acordo com os separadores vírgula (,), espaço ( ), ponto-e-vírgula (;) ou nova linha (\n). Após cada parte é testada para ver se possuem três parte em um modelo ____@____.____. Caso algum dos *e-mail*s adicionados não seja

Figura 4.10: Zona de compartilhamento de arquivos



Fonte: o autor

valido, uma mensagem de erro aparece.

Se os *e-mail*s são validos, uma requisição POST é feita para o *backend* enviando a lista de *e-mail*s e a senha, caso exista. A resposta dessa requisição é o novo *link* gerado.

4.2.4 Identificador (*username*)

Um dos objetivos para o futuro do Tracim era tornar possível fazer menções em documentos e comentários, para isso foi necessário criar um identificador único para cada usuário. Dentre as discussões sobre este tema, destacam-se:

1. Como a maioria dos usuários do Tracim são de língua francesa, como traduzir *username*?

   A principal razão para esta pergunta é que *username* é geralmente traduzido para nome de usuário, entretanto, em Tracim temos o termo "nome completo" que corresponde a um nome público do usuário que não é necessariamente único. Assim, pesquisamos como a diferença era tratada em outras plataformas, fizemos a pergunta no site LinuxFR e decidimos usar o termo "identificador".

2. O usuário decide sobre seu próprio identificador ou é automático? Se o usuário defini-lo, ele tem o direito de mudá-lo sempre que quiser?

   Foi decidido que o usuário pode escolher seu identificador e, com base no que é feito no mercado hoje, achamos que era uma boa ideia dar ao usuário a liberdade de mudá-lo sempre que ele quiser.

3. Como definir o identificador para os usuários que já possuem uma conta no Tracim?

   Se o identificador for *NULL* (na resposta da API), decidimos sugerir ao usuário que defina seu identificador no momento da conexão com uma *popup* que pode ser ignorada. Para que o *popup* não apareça sempre para um usuário que decidiu deixar

o identificador vazio, nós utilizamos um *cookie* que diz se o usuário X decidiu deixar seu identificador nulo ou se ele ainda não se conectou.

4. Quais são as regras para a criação de um identificador?
   Definimos que um identificador deve ter entre 3 e 255 caracteres, somente caracteres `azAZ09-_` são permitidos, não pode conter espaços brancos e deve ser único entre os usuários da instância.

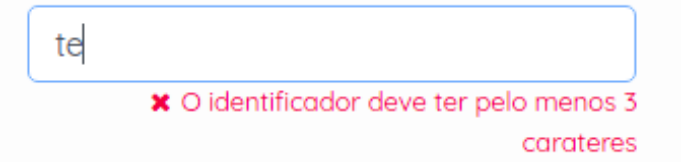
No que diz respeito à *interface*, foi decidido que onde quer que o identificador seja exibido como informação do usuário, ele deve ser precedido por um @, porque este será o atalho usado para fazer uma menção e porque é uma prática comum na *internet*.

Entretanto, quando o usuário define seu identificador, todos os *placeholders* e etiquetas de formulário não devem incluir o @. O objetivo desta medida era encorajar o usuário a não usar o @ no início de seu identificador ao escolhê-lo, já que o @ não é um caractere válido.

Outra decisão tomada pensando em melhorar a experiência do usuário foi adicionar um *feedback* em tempo real ao definir o identificador. Como podemos ver na Figura 4.11, ao escrever o identificador, o usurário sabe se ele está de acordo com as normas de tamanho, caracteres permitidos, palavras reservadas ou até mesmo se o identificador está disponível ou se já foi escolhido por outro usuário.

Esse retorno vem do *backend*, então a cada letra digitada o *frontend* envia uma solicitação para a API perguntado se o identificador é válido.

Figura 4.11: *Feedback* em tempo real para o identificador



Fonte: o autor

Ao total, 22 componentes (10 telas) foram modificados e/ou adicionados para realizar esta funcionalidade.

4.2.5 TLM (*Tracim Live Message*)

Até 2020, Tracim não tinha nenhum tipo de atualização em tempo real, ou seja, se um usuário tivesse um documento aberto e outro usuário fizesse um comentário na

linha cronológica desse mesmo documento, o primeiro usuário teria que fechar e abrir novamente o documento ou atualizar o navegador para ver o comentário. Como um dos principais objetivos do Tracim é facilitar a colaboração, isto pode ser um problema. Foi por isso que os TLMs (*Tracim Live Messages*) foram desenvolvidos.

Um problema foi que quando iniciamos os *issues*, não enriquecemos e validamos a especificação e o desenvolvimento acabou por ficar mais próximo do que o que o desenvolvedor entendia ser necessário fazer do que o que estava realmente escrito na descrição do *issue*. No final, houve algumas diferenças entre o que foi feito e as expectativas do gestor do projeto, o que resultou em alguns pedaços de código que tiveram de ser refeitos. Todo este problema acabou por conduzir a reflexões sobre as dificuldades de comunicação que encontramos durante o trabalho remoto e sobre a redação e gestão das especificações do projeto.

Focando um pouco mais na parte técnica dessa funcionalidade, foi decidido que o *backend* enviaria mensagens via SSE (*Server-Sent Events*), estas conteriam o tipo da entidade, o tipo do evento principal, possivelmente um subtipo da entidade e os campos necessários, como mostra a Tabela 4.1, onde C = *created*, M = *modified*, D = *deleted* e U = *undeleted*.

Tabela 4.1: Tabela dos tipos de TLM

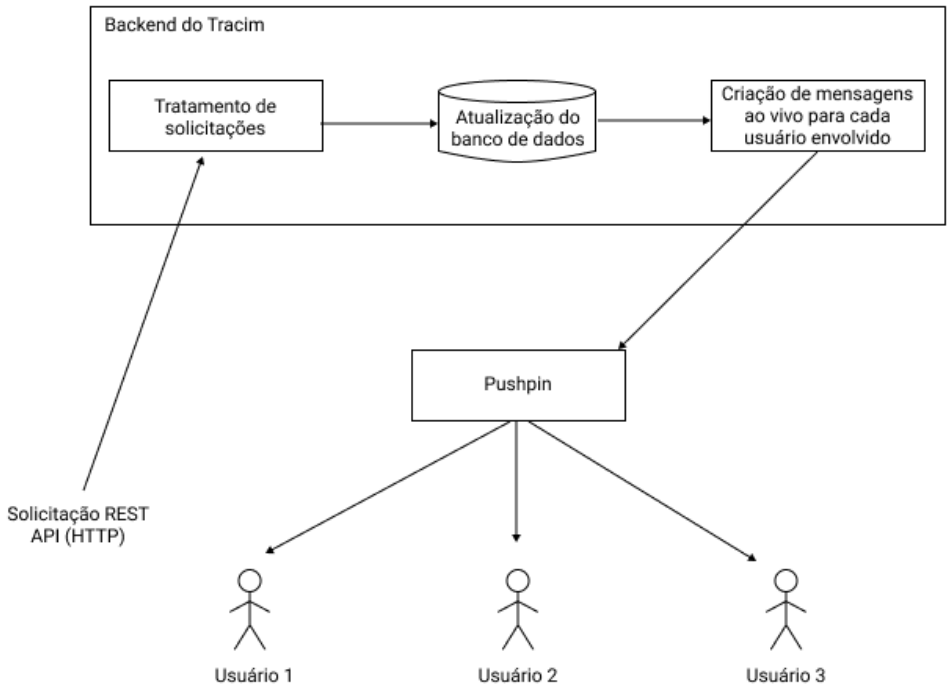
entity type	core event types	sub type	fields
user	C/M/D/U		author, user
workspace	C/M/D/U		author, workspace
workspace_member	C/M/D		author, user, workspace, member
content	C/M/D/U	file/thread/ html-document/ kanban/folder/ comment	author, workspace, content
mention	C		author, workspace, content, mention
reaction	C/D		author, workspace, content, reaction, user
workspace_subscription	C/M/D		author, workspace, subscription, user
tag	C/M/D		author, workspace, tag
content_tag	C/D		author, workspace, tag, content
user_call	C/M/D		author, user_call, user

Na Figura 4.12 podemos ver uma representação da arquitetura da funcionalidade.

Os usuários, ao se conectarem, abrem um canal de comunicação com o servidor proxy Pushpin (FANOUT, 2022), que tem como principal objetivo manter esse canal aberto até que o usuário feche sua sessão. Internamente, ele usa um sistema de *heartbeat* para garantir que o usuário ainda está conectado e não ouve algum problema como uma queda de conexão.

Além de compartilhar em modo *broadcast* para o *frontend* de todos os usuários conectados, o *backend* estoca todas informações sobre os TLMs no banco de dados para poder utilizar futuramente em outras funcionalidades, como o muro de notificações descrito na seção 4.2.6.

Figura 4.12: Arquitetura *Tracim Live Messages*



Fonte: Simplificação do esquema apresentado em https://fosdem.org/2022/schedule/event/collabtracim/

O *frontend* implementa um ouvinte que espera pela chegada de um novo TLM, o que desencadeia o envio de um evento personalizado JavaScript chamado *TracimLiveMessage* com o tipo `entity_type.core_event_type` ou `entity_type.sub_type.core_event_type` e os respectivos campos de dados.

Inicialmente, o nosso objetivo era que cada aplicação Tracim tivesse um manipulador para cada evento *TracimLiveMessage* que influenciasse o seu comportamento ou *interface*. Mas durante a implementação, encontramos problemas na aplicação principal de *frontend* que tinha dois componentes processando a mesma mensagem TLM. As vezes os componentes estavam sozinhos (ou seja, apenas um deles tinha sido executado), e por

isso tinha que processar a mensagem, mas as vezes eles eram executados em conjunto, o que resultava no processamento da mesma mensagem duas vezes.

Para lidar com esta seção crítica, criamos um componente React global à aplicação chamado *ReduxTlmDispatcher.jsx*, que escuta todos os TLMS e executa as atualizações no Redux segundo o que for necessário. Os demais componentes apenas olham para as atualizações globais do Redux e não para os TLMs.

Como foi necessária muita refatoração de código para obter a melhor forma possível de receber e lidar com estes eventos, aproveitamos a oportunidade para utilizar o mesmo mecanismo para os outros eventos personalizados JavaScript que já tínhamos no Tracim e fizemos esta refatoração também. Ao todo 16 componentes foram modificados e/ou adicionados para alcançar esta funcionalidade.

No início dos testes de integração, o gestor do projeto viu a diferença entre o que esperava da funcionalidade e o que tinha sido desenvolvido. No desenvolvimento, garantimos que todas as atualizações de conteúdo eram 100% automáticas, ou seja, assim que um usuário alterava um documento de texto, o texto era atualizado ao mesmo tempo para todos. Contudo, o gestor do projeto indicou que isto poderia ser muito confuso para o usuário, pois poderia acontecer que o texto mudasse completamente enquanto alguém o lesse. Sendo assim, foi feito um processo de esclarecimento do comportamento e a funcionalidade foi reimplementada com uma melhor experiência de usuário.

4.2.6 Notificações

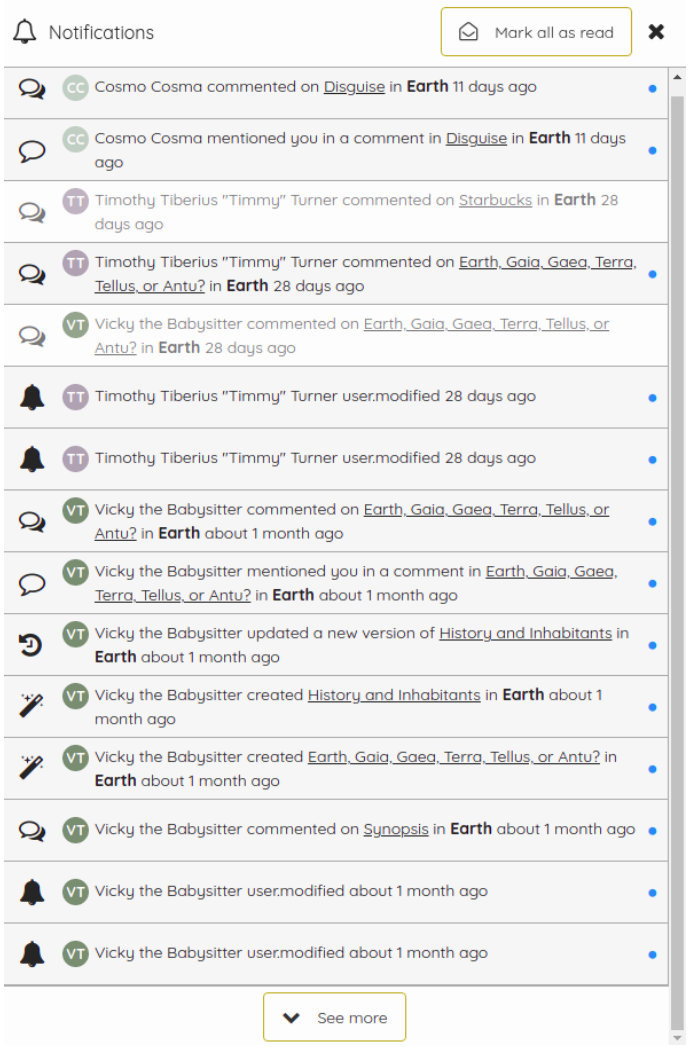
Foi constatado também que Tracim estava enviando muitos *e-mail*s e que muitas vezes algumas informações eram perdidas, especialmente para os usuários que têm vários espaços. Além disso, devido a grande quantidade, os *e-mail*s poderiam ser considerados como *spam*. Até agora, as notificações da plataforma de todos os tipos eram enviadas por *e-mail*. Isto é, um *e-mail* era recebido a cada conteúdo modificado, outro para cada membro adicionado ao espaço, etc.

O recurso de notificação destina-se, portanto, a permitir que apenas certas informações sejam enviadas por *e-mail* e o restante seja feito dentro da plataforma através de um muro de notificações. Isto facilitaria também mudanças na configuração no futuro, tornando as notificações mais inteligentes e mais convenientes para o usuário.

Tendo em conta os TLMs que haviam sido implementados, o usuário ainda obteria *feedback* em tempo real sobre o que está acontecendo em sua instância Tracim durante

o uso e sem precisar de múltiplos *softwares*. A informação é centralizada e acessível de forma fácil, conforme vemos na Figura 4.13.

Figura 4.13: Muro de notificações



Fonte: o autor

Como explicado na seção 4.1, o desenvolvimento aqui foi dividido entre *issue* de base e melhorias. No entanto, o conjunto desses desenvolvimentos nunca foi implementado, pois para a empresa era mais vantajoso passar para a próxima funcionalidade do que fazer todas as melhorias dessa. Felizmente, algumas melhorias foram retomadas um tempo depois. Para ser mais precisa, esses desenvolvimentos de base e primeiras melhorias foram feitos em setembro de 2020 e em outubro de 2021 outras melhorias foram aplicadas.

Para que o muro de notificações fosse sempre acessível e que fosse atualizado assim que um TLM chegasse, todos os dados relativos a ele foram salvos no Redux em um objeto contendo:

- *list*: uma lista de notificações;
- *hasNextPage*: se todas as notificações já foram carregados ou ainda tem outras páginas;
- *nextPageToken*: *token* utilizado no backend para obter a próxima página de notificações.

E as seguintes ações foram adicionadas:

- *setNotificationList*: define uma nova lista de notificações;
- *appendNotificationList*: adiciona um conjunto de novas notificações ao final da lista já existente, usado quando o usuário clica para ver mais;
- *addNotification*: adiciona uma nova notificação no começo da lista já existente, usado quando uma nova notificação chega enquanto o usuário está conectado;
- *updateNotification*: atualiza uma notificação já existente, usado quando o usuário lê uma notificação;
- *setNextPage*: define os campos *hasNextPage* e *nextPageToken*.

4.2.7 Menções

Com essa mesma ideia de centralizar a informação em um mesmo *software*, nós observamos que para informar uma pessoa específica sobre uma mudança ou mensagem, os usuários mandavam um *e-mail* ou usavam algum outro *software*.

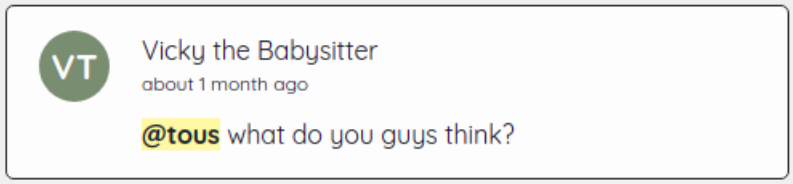
A funcionalidade das menções, criada para resolver o problema, consiste em enviar uma notificação ao usuário assim que alguém envie um comentário ou escreva em um documento @ seguido pelo seu identificador. Para a parte prática, nós teríamos duas etapas de base: detecção da menção e envio da notificação.

A análise sintática, também conhecido por *parser*, da mensagem é feita inicialmente no *frontend*, identificando possíveis menções e as envolvendo em elementos HTML do tipo `span`, para que, ao enviar o texto para o *backend*, ele pudesse facilmente encontrar as menções e analisá-las.

Para fazer isso, foi usado o DOMParser (MOZILLA, 2022), que é uma *interface* que permite navegar no código fonte HTML de uma sequência de caracteres em um documento DOM, e um algoritmo que ao achar um @ precedido de um espaço ou começo de frase e seguido por um conjunto de caracteres autorizados, adicionava um `span` com

um *id* único e uma classe "*mention*". A classe serve para ter um auxílio visual, todas as menções são em negrito como na Figura 4.14.

Figura 4.14: Comentário com menção



Fonte: o autor

O *backend* por sua vez procura por esses elementos `span` e confere se o texto corresponde a um identificador existente ou não. Se sim, ele envia uma notificação ao usuário, se não, ele envia uma mensagem de erro que sera tratada pelo *frontend*.

Neste desenvolvimento, foi preciso lidar com o Tinymce (TINY, 2022), um *software* usado no Tracim para permitir a edição avançada de conteúdo. Foi muito complicado fazer mudanças nele porque como é algo externo ao Tracim foi necessário fazer pedaços de código específico para alterar seus parâmetros de configuração ao invés de diretamente seu código.

Assim como a funcionalidade anterior, no momento em que as bases foram terminadas, começamos a implementar melhorias, tais como: as menções para o próprio usuário possuem uma classe própria para adicionar um fundo amarelado de destaque, detecção do @ enquanto o usuário escreve para sugerir formas de completar automaticamente a menção, etc.

5 TESTES COM USUÁRIOS

Nas próximas seções falaremos sobre os testes realizados com diferentes usuários para analisar se a experiência realmente melhorou após os desenvolvimentos.

Na seção 5.1, é explicada toda a metodologia usada, explicando desde os fatores de ambiente de trabalho considerados antes do teste, o teste em si e até as possíveis formas de análise pós teste. Na seção 5.2, os resultados obtidos são apresentados e analisados.

5.1 Metodologia dos testes

Para realizar os testes com usuários foram preparadas duas instâncias que rodavam localmente em um computador pessoal com versões diferentes de Tracim. O objetivo era testar a versão antiga (sem as modificações) e a versão nova (modificada). A primeira versão é a 2.2_rc1 que data de 9 de maio de 2019 e a segunda versão é 3.1.1 que data de 21 de setembro de 2020.

Para facilitar a compreensão dos participantes e evitar tendências para a versão nova, as versões foram denominadas rosa (2.2_rc1) e amarela (3.1.1), segundo os parâmetros escolhidos para a sua cor principal.

Ambas instâncias possuíam um banco de dados quase idêntico, baseado no desenho animado Os Padrinhos Mágicos (FANDOM, Community, 2012?). A única diferença era relacionada as novas funcionalidades implementadas (por exemplo o identificador de usuário que foi explicado na seção 4.2.4). Com isso, a experiência dos testes das duas versões diferia somente nas funcionalidades.

Junto com as duas instâncias, foi preparado também um formulário utilizando Framaforms (YAKFORMS, 2022), que foi inspirado em alguns modelos e artigos online (SOLLOWAY, Ashley; NICHOLS, Timothy; COOLRIDGE, Mariah; SKILLMAN, Ellis, 2022)(CHOPRA, Paras, 2010), para verificar se a pessoa já utilizou Tracim, com que frequência e quando. Avaliando assim o quão confortável a pessoa estava com a plataforma e se ela já conhecia as diferenças entre versões.

Nesse mesmo formulário, após as perguntas, uma lista de tarefas era dada e o tempo levado para realizar cada tarefa cronometrado. Inicialmente, para pessoas confortáveis com o *software*, o tempo máximo antes de considerar como a tarefa como "não conseguiu realizar" foi de cinco minutos e para pessoas não ou pouco confortáveis o tempo foi de dez minutos.

Logo nos primeiros testes, foi possível detectar algumas falhas na metodologia dos testes, ajusta-las e começar novamente os testes. Uma delas foi o tempo máximo de cada tarefa que estava muito longo e algumas pessoas desistiam antes de chegar no tempo máximo. Diminuímos, então, para três minutos para uma pessoa experiente, ajudando se possível com alguma dica quando a pessoa chegava em torno de um minuto e meio, e para as pessoas menos experientes cinco minutos, ajudando ao chegar em dois.

Cada tarefa era feita em uma versão e depois na outra. Para diminuir um possível viés causado pela ordem em que as versões eram testadas, metade dos usuários foi testado primeiramente com a versão rosa e a outra metade primeiro com a versão amarela.

Ao final de cada tarefa, o usuário deveria dizer em qual versão ele teve uma melhor experiência realizando-a ou se não houve uma diferença significativa. No fim de todas as tarefas, o participante escolhia a versão que ele prefere de maneira geral, qual foi a parte/tarefa mais confusa do teste e qual foi a tarefa em que ele sentiu uma diferença mais chocante entre versões.

Todos os testes foram feitos de forma presencial e na mesma máquina.

Retomando os três fatores principais para os testes de *UX* explicados na seção 2, podemos facilmente fazer um paralelo com os nossos testes:

- análise da eficácia: determinar um tempo máximo segundo a capacidade e conhecimento do participante determinada pelo formulário e dar a possibilidade de abandonar a tarefa;
- análise de eficiência: cronometragem do tempo;
- análise da satisfação: questões do formulário e análise holística.

Para os três fatores que influenciam na experiência, um cuidado extra foi tomado para que todos os testes fossem feitos na mesma máquina, com os mesmos periféricos e de forma presencial (mesmo sistema para todos) e que parte do contexto de uso fosse levado em conta através do formulário.

O único fator que não foi nem um pouco controlado foi o usuário, que acabou tendo um perfil similar por serem todos de círculos sociais parecidos, mas nenhum esforço além foi feito. Inclusive, se fosse possível ter um alcance maior, certamente seriam privilegiados participantes similares ao nosso público alvo e que nunca usaram a plataforma.

A amostra de usuários também foi um elemento importante a ser estudado antes do começo dos testes. Em 2000, Jakob Nielsen publicou um artigo sobre o número ideal

de usuários a serem testados para a usabilidade (NIELSEN, Jakob, 2000) e seu estudo concluiu que apenas cinco usuários eram necessários. Alguns anos mais tarde ele publicou outros artigos de continuação ao primeiro que citavam casos específicos onde testar mais usuários traria um resultado melhor.

Um desses casos é o estudo quantitativo que, segundo ele, consiste em coletar métricas de qualidade, tais como tempo de aprendizagem, eficiência de uso, memorabilidade, erros do usuário e satisfação subjetiva. Então, em 2006 (NIELSEN, Jakob, 2006), ele definiu vinte usuários como uma boa amostragem para esse tipo de teste.

Para esse trabalho 19 usuários participaram dos testes.

5.2 Resultados e análises

Nessa seção, analisamos os resultados dos testes feitos. Cada subseção aborda uma tarefa específica do formulário presente no apêndice A e na subseção 5.2.7 temos algumas análises globais ao teste.

5.2.1 Tarefa 1

A primeira tarefa dada ao participante foi "**Abra o conteúdo Juandissimo / Gnomes**", onde o objetivo seria encontrar um arquivo.

Como o banco de dados era o mesmo, assim que o participante achasse o arquivo na primeira versão, ele saberia onde estava na outra. Para resolver esse problema, a instrução dada era procurar por *Juandissimo* na primeira versão testada e *Gnomes* na segunda. Ambos arquivos possuíam a mesma arborescência em espaços diferentes. A seguir, dispomos do caminho de cada um, começando pelo espaço e após todas as pastas até o arquivo.

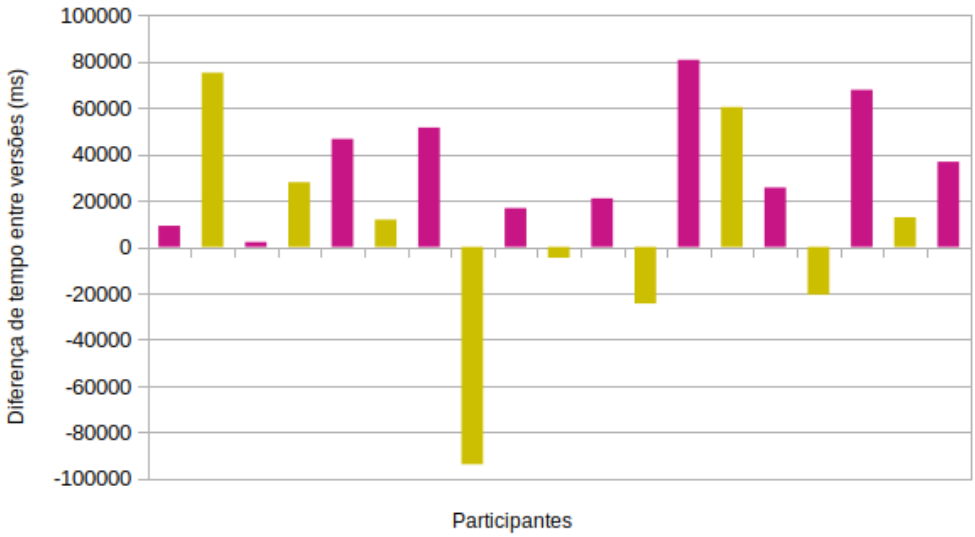
- *Fairy World* $\rightarrow$ *Fairy Godparents* $\rightarrow$ *Other fairies* $\rightarrow$ *Juandissimo*
- *Earth* $\rightarrow$ *Not for human beings* $\rightarrow$ *Already done* $\rightarrow$ *Gnomes*

Era esperado que na versão rosa os participantes procurassem em cada espaço e cada pasta até achar o arquivo e que na versão amarela eles usassem o mecanismo de busca disponível no cabeçalho.

No gráfico da Figura 5.1 temos a diferença de tempo em milissegundos da versão

rosa para a versão amarela. Podemos ver que aproximadamente 79% (15 participantes dentre os 19) realizou a tarefa mais rapidamente na versão amarela, independente da versão de início (indicada pela cor da barra). O desvio padrão das diferenças para essa tarefa foi de aproximadamente 41129,2 ms, sendo o valor médio 21190 ms.

Figura 5.1: Diferença entre versões na tarefa 1



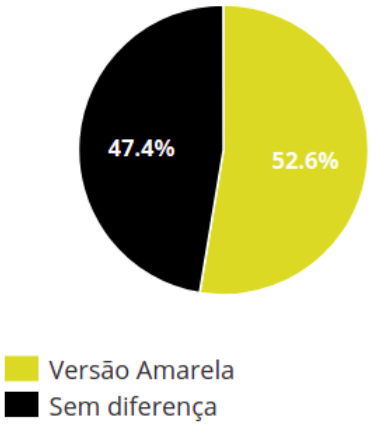
Fonte: o autor

Apesar desse resultado ter sido positivo, a diferença é menor do que esperada. Uma boa parte das pessoas usou o mesmo mecanismo de busca por exaustão nas duas instâncias. Os participantes ficavam tão focados em entender o *software* e no conteúdo em si que não olhavam para o cabeçalho da página, onde a entrada de busca se encontrava.

Os pontos se intercalam entre usuários que começaram pela versão rosa e participantes que começaram pela amarela. Mesmo não aparentando mudanças significativas no tempo, podemos constatar que todos os valores negativos, ou seja, pessoas que demoraram mais tempo na versão amarela do que na rosa, são de participantes que iniciaram na versão amarela. Supõe-se que esse efeito acontece pois o participante reservou um tempo para conhecer Tracim antes de fazer a busca de fato, visto que ela estava mais visível, enquanto na outra versão ele se foca no desafio.

Notavelmente, a preferência pela versão amarela é relativamente fraca, como vemos na Figura 5.2. Entretanto, dentre os 10 participantes que a preferiram, 9 utilizaram o mecanismo de busca como esperado e 7 incluíram a tarefa 1 na sua resposta para a questão “Em que tarefa foi a diferença entre as versões é mais marcante?” como apresentado na subseção 5.2.7.

Figura 5.2: Resultados da tarefa 1



Fonte: o autor

5.2.2 Tarefa 2

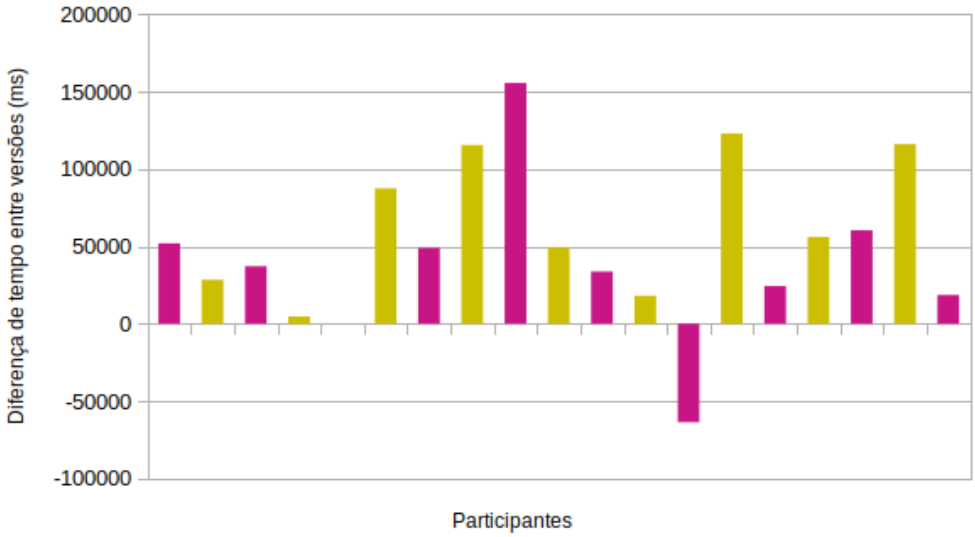
A segunda tarefa era "**Compartilhar o arquivo "Anti Cosmo" com o e-mail cosmo@cosmo.cosmo**". Esperava-se que na versão rosa os usuários enviassem o arquivo por *e-mail* e na versão amarela que eles usassem o mecanismo de compartilhamento implementado.

Essa tarefa foi uma surpresa positiva no geral, como podemos ver no gráfico da Figura 5.3, somente uma pessoa foi mais performante na versão rosa do que na amarela. Curiosamente, não foi a mesma pessoa que preferiu a versão rosa na Figura 5.4. Os participantes comentaram que o ícone usado no menu era muito conhecido e usado em todas redes sociais, então eles não tiveram que pensar duas vezes antes de clicar (Figura 4.9 na subseção 4.2.3). O desvio padrão das diferenças para a tarefa 2 foi de aproximadamente 51466,02 ms, sendo o valor médio 50817,9 ms.

A preferência pela versão rosa veio do fato da instância estar rodando no navegador Firefox e ao clicar com o botão direito do *mouse* na imagem, o participante havia uma opção "*Email image...*", enquanto a amarela rodava no Google Chrome que não dispõe dessa opção.

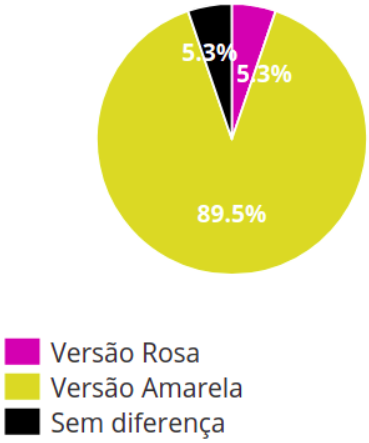
Infelizmente, essa tarefa foi a segunda com maior taxa de desistência. Cinco participantes abandonaram a tarefa na versão rosa. Provavelmente, esse número é ligado ao fato de não ser possível fazer o que foi pedido sem utilizar um *software* externo a Tracim (como a ferramenta de *e-mail*).

Figura 5.3: Diferença entre versões na tarefa 2



Fonte: o autor

Figura 5.4: Resultados da tarefa 2



Fonte: o autor

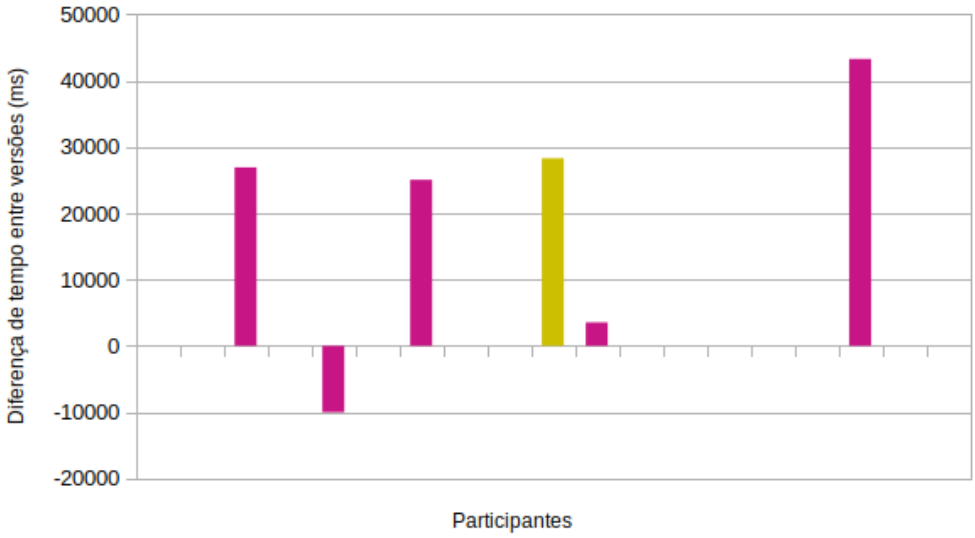
5.2.3 Tarefa 3

A terceira tarefa foi "**Uma pessoa que não tem uma conta no Tracim quer compartilhar um arquivo com todos os membros do espaço Cosma-Fairywinkle family, mas não sabe seus detalhes de contato. Como você ajudaria essa pessoa?**". Assim como a tarefa anterior, era esperado que na versão rosa fosse usado uma ferramenta externa e na versão amarela o mecanismo implementado.

Conhecendo o *software*, já era esperado que essa tarefa tivesse menos sucesso, pois ela foi feita com urgência, não teve muito tempo de concepção e não é muito utilizada nem mesmo pela empresa no dia-a-dia.

Dentre os 19 participantes, 13 disseram que fariam exatamente a mesma coisa nas duas versões, como podemos ver pela quantidade de zeros na Figura 5.5. O desvio padrão das diferenças para a tarefa 3 foi de aproximadamente 13407,11 ms, sendo o valor médio 6160 ms.

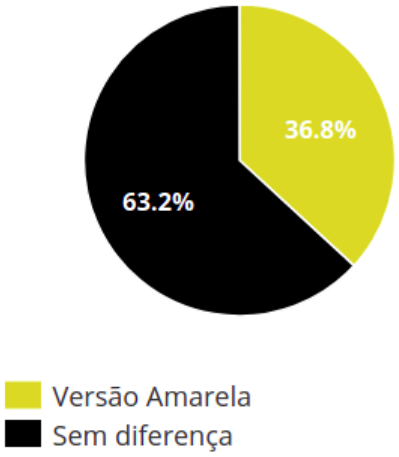
Figura 5.5: Diferença entre versões na tarefa 3



Fonte: o autor

Devido a esse índice de descoberta baixo, também temos um impacto direto na preferência pela versão, o que fica claro na Figura 5.6. Nós tivemos 12 das 19 pessoas que não tiveram uma preferência clara, sendo uma dessas pessoas o participante que levou mais tempo na versão rosa.

Figura 5.6: Resultados da tarefa 3



Fonte: o autor

Curiosamente, duas pessoas preferiram a versão amarela mesmo fazendo a mesma coisa nas duas versões. Ambas tinham como solução criar uma conta Tracim para a

pessoa externa e adicioná-la no espaço correspondente. A preferência ocorreu, pois na versão amarela eles tinham a explicação de cada função de um membro no espaço, como podemos ver na Figura 5.7, o que fez com que eles se sentissem mais seguros nessa versão.

Figura 5.7: Explicação da função de cada membro do espaço

Fonte: o autor

Boa parte das pessoas procurou pela funcionalidade exclusivamente na página inicial do espaço, com isso em mente, uma forma de melhorar o índice de descoberta da funcionalidade seria adicionar na página inicial um botão de ação, por exemplo "Criar *link* de *upload*".

5.2.4 Tarefa 4

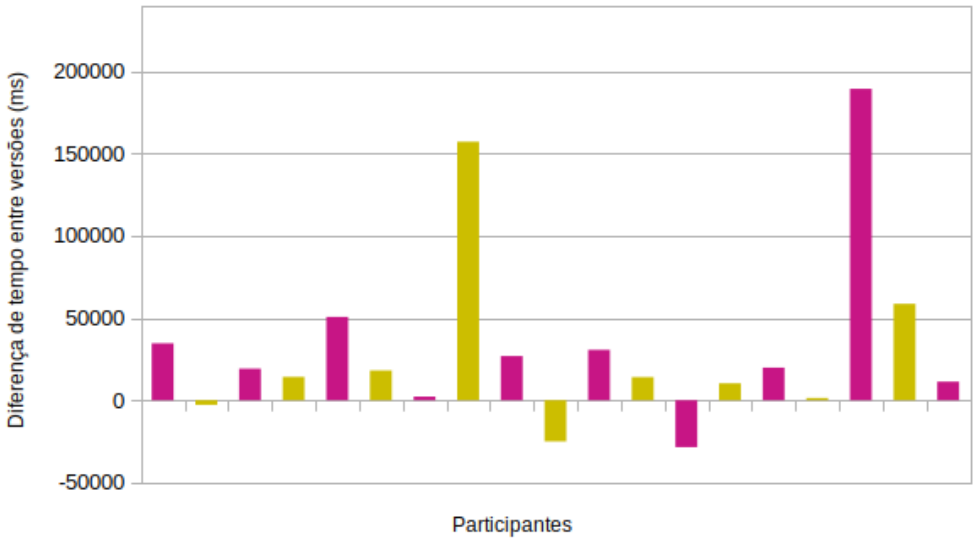
A quarta tarefa foi "**No arquivo Disguise, converse entre Cosmo e Wanda. Exemplo: "O que você acha desse disfarce?""Acho ótimo!""Ok, vamos fazê-lo essa noite."**", onde o participante controlava os dois usuários (Cosmo e Wanda).

Nesse caso eles deveriam fazer o mesmo procedimento nas duas versões, enviar comentários no histórico do arquivo, e o objetivo era medir o quanto a falta da atualização em tempo real perturbaria os usuários e faria com que eles levassem mais tempo tentando entender o que estava acontecendo.

Para as pessoas que já conheciam o software, a diferença de tempo foi bem menor que a esperada pois eles já sabiam que a mensagem tinha sido bem enviada e só não aparecia na tela. Considerando todos valores apresentados na Figura 5.8, a média de diferença de tempo entre versões foi aproximadamente de 31685 ms, porém se consideramos so-

mente os participantes que não conheciam Tracim, a média sobe para 45911 ms. O desvio padrão na tarefa 4 foi de aproximadamente 53054,25 ms, sendo o valor médio 31685,53 ms.

Figura 5.8: Diferença entre versões na tarefa 4

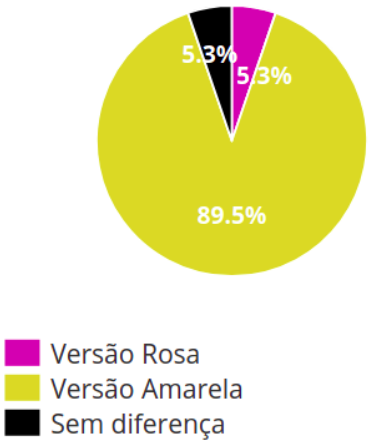


Fonte: o autor

Apesar de esperar uma média de diferenças maior, os *Tracim Live Messages* foram sem dúvida um grande avanço no *software* e permitiram que outras funcionalidades, como as notificações, existissem.

Na Figura 5.9, notamos que a preferência ainda é clara pela versão amarela e somente 5,3% (que corresponde a uma pessoa) não viram diferença. A única pessoa que votou pela versão rosa justificou que foi principalmente pelo *design* dos comentários que era diferente entre versões e que ela preferia muito mais o antigo.

Figura 5.9: Resultados da tarefa 4



Fonte: o autor

Uma pessoa não terminou a tarefa, pois após enviar a primeira mensagem e mudar de usuário, ela disse que nunca teria como saber que alguém enviou uma mensagem, se não fosse ela mesmo que tivesse feito, e que, no papel de segundo usuário, ela não via sentido em procurar uma mensagem ou querer responder algo que ela nem sabia que existia.

Essa resposta provocou uma reflexão sobre como chocaria bem mais os usuários se a pessoa que aplica o teste fosse um dos usuários, mas para isso seria preciso dois computadores, o que não era possível para esse momento.

5.2.5 Tarefa 5

A tarefa 5 era "**Peça para Timmy Turner ler o documento Synopsis**" e esperava-se que o participante mencionasse o Timmy na versão amarela, usando o carácter @, e que usasse o *e-mail* para a versão rosa.

Os participantes que adicionavam somente um comentário sem menção eram interrogados sobre como eles saberiam que o Timmy ia de fato ler e que seria o Timmy certo que receberia a mensagem, visto que o espaço possuía mais de um membro chamado Timmy. A maioria tentava buscar outras soluções depois das perguntas.

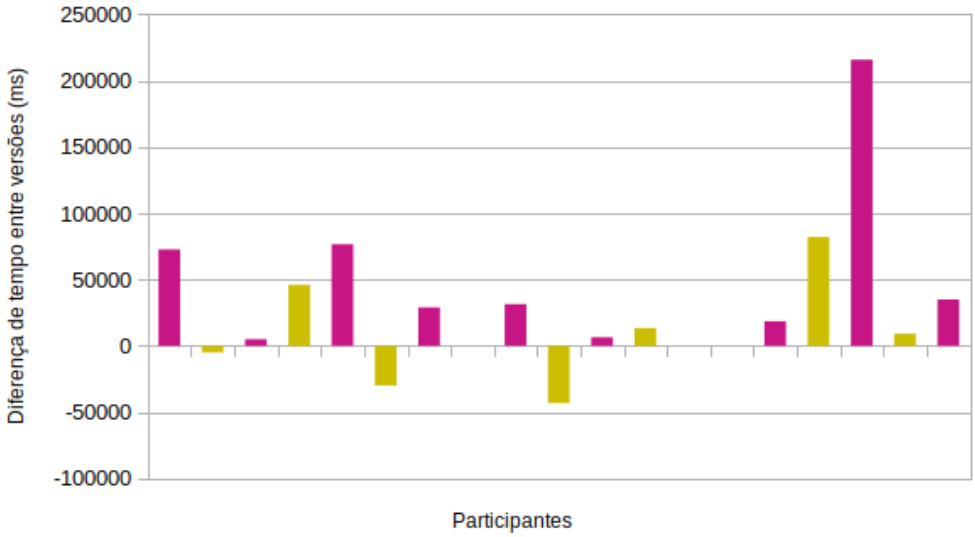
Na Figura 5.10, temos três participantes que fizeram exatamente a mesma coisa nas duas versões, ou enviavam um *e-mail* ou só comentavam sem mencionar e torciam para que ele fosse ver. Outras três pessoas foram mais performantes na versão rosa, todas haviam começado a tarefa pela versão amarela, então testavam direto a menção e visto o histórico das outras tarefas falavam rapidamente em usar uma ferramenta externa. O desvio padrão das diferenças para essa tarefa foi de aproximadamente 54450,04 ms, sendo o valor médio 29645,26 ms.

Independente das pessoas terem demorado um certo tempo para fazer a menção, a Figura 5.11 mostra que a satisfação geral ainda é positiva em relação a versão amarela.

O ponto que fez as pessoas demorarem mais foi elas tentarem criar uma discussão privada com o Timmy ao invés de usar o documento que era público. Na frustração de não conseguirem, três participantes abandonaram a tarefa, dois na versão rosa e o outro em ambas versões.

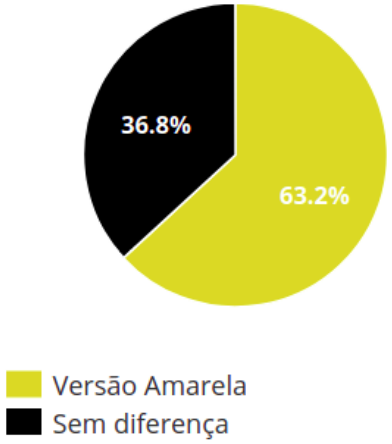
A tarefa 5 foi a segunda tarefa que mais recebeu votos para "Em que tarefa foi a diferença entre as versões é mais marcante?" (subseção 5.2.7). Seis pessoas consideraram a menção uma grande melhoria de uma versão para a outra.

Figura 5.10: Diferença entre versões na tarefa 5



Fonte: o autor

Figura 5.11: Resultados da tarefa 5



Fonte: o autor

5.2.6 Tarefa 6

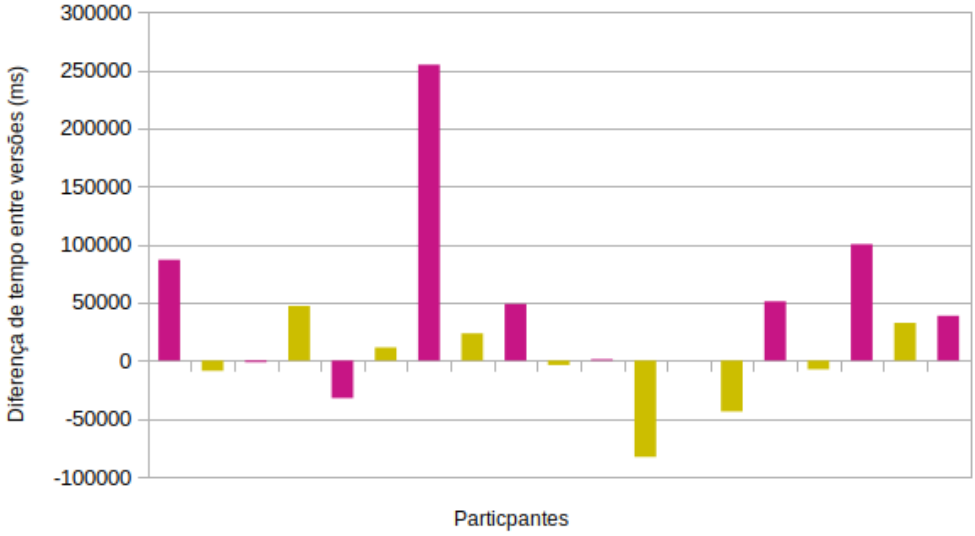
A última tarefa dada era "**Qual foi o último conteúdo comentado por Vicky / Timmy?**". Para essa tarefa havia um muro de notificações na versão amarela e era necessário abrir cada arquivo na versão rosa. Supunha-se que nesse momento do teste os participantes já estavam acostumados com o conceito dos espaços e seus membros e que só procurariam no único espaço em que Vicky e Timmy eram membros.

No final, poucas pessoas de fato usaram o muro de notificações e uma boa parte procurou em todos os espaços. Como era algo que eles já haviam feito na primeira tarefa e dessa vez um pouco mais complicado, a tarefa ficou cansativa e as pessoas abandonavam.

Com a maior taxa de desistência, seis pessoas ao total, nós tivemos quatro participantes que desistiram em ambas versões e dois que abandonaram somente na rosa.

Ainda que o *zigzag* do gráfico da Figura 5.12 faça pensar que a versão que o usuário começava influenciava no seu tempo, não foi possível achar uma relação forte. Somente uma das diferenças negativas e o participante que teve a maior diferença positiva começaram pela versão rosa, o que fez com que a média de diferença de tempo das pessoas que começaram na versão rosa foi de 54727 ms e na versão amarela de -3684 ms. O desvio padrão das diferenças para a tarefa 3 foi de aproximadamente 68441,1 ms, sendo o valor médio 27058,42 ms.

Figura 5.12: Diferença entre versões na tarefa 6



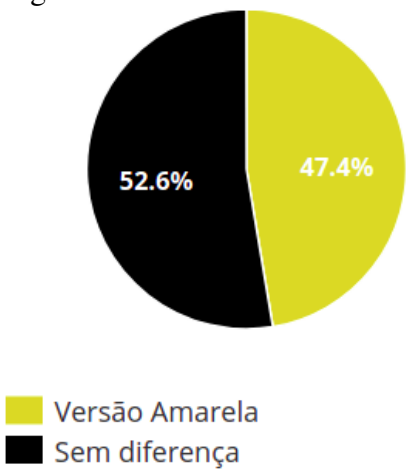
Fonte: o autor

A tarefa 6 era a única tarefa que havia de fato uma resposta correta. Das 19 pessoas, quatro responderam o conteúdo incorreto, sendo que uma dessas pessoas errou em ambas versões. Algumas pessoas buscaram apenas nos conteúdo do tipo discussão, mesmo já tendo presenciado comentários em outros tipos.

Ao total, onze pessoas incluíram a tarefa 6 na sua resposta à “Qual foi a tarefa mais confusa do teste?” como apresentado na subseção 5.2.7.

Não obstante todas as desistências e os erros, a versão amarela ainda obteve a preferência de nove pessoas, o que totaliza aproximadamente 47% dos participantes, como pode ser visto na Figura 5.13

Figura 5.13: Resultados da tarefa 6

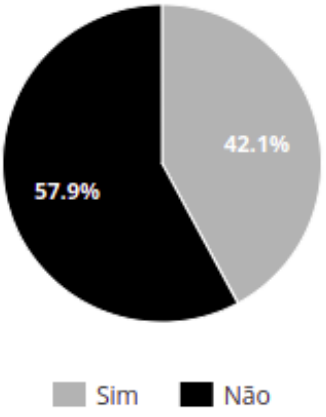


Fonte: o autor

5.2.7 Análises gerais

Antes de começar as tarefas, foi analisado nível de conhecimento que os participantes tinham do Tracim. A primeira pergunta era "**Você já usou Tracim?**" e para as onze pessoas que responderam não, 57,9% dos participantes como vemos na Figura 5.14, foi dada uma breve introdução sobre o *software* para que elas não ficassem perdidas.

Figura 5.14: Porcentagem de participante que conheciam Tracim

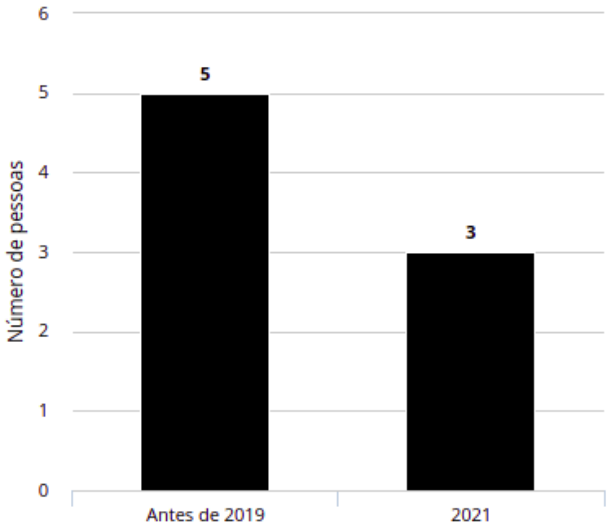


Fonte: o autor

Para os demais participantes era perguntado "**Quando começou a usar o software?**" referindo-se ao ano (Figura 5.15) e caso a resposta fosse diferente de "Antes de 2019" era perguntado também o mês (Figura 5.16). O objetivo era saber se eles haviam conhecido a versão de Tracim anterior às modificações aplicadas a partir de maio de 2019.

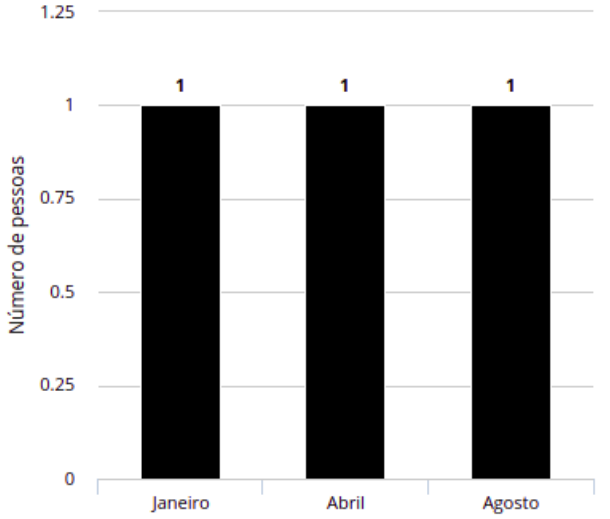
Por último nessa etapa, para os 8 participantes que responderam sim à primeira questão, perguntava-se "**Com que frequência você utiliza Tracim atualmente?**". O

Figura 5.15: Ano de início de uso do *software*



Fonte: o autor

Figura 5.16: Mês de início de uso do *software*



Fonte: o autor

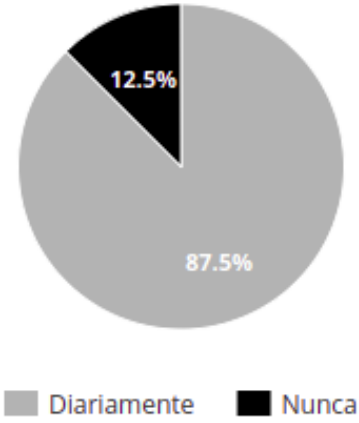
objetivo era medir o quão acostumados eles estavam com a versão atual, que inclui todas as mudanças. Como podemos ver na Figura 5.17, somente uma pessoa (12,5%) não usa o *software* diariamente.

Após realizar todas as tarefas, as pessoas respondiam três perguntas em relação ao teste. O objetivo dessas perguntas era confirmar, com o *feedback* do participantes, coisas observadas pela pessoa que aplicava o teste.

A primeira pergunta era "**Qual foi a tarefa mais confusa do teste?**", com a possibilidade de escolher uma ou mais tarefas.

Pelos resultados anteriores era esperado que a tarefa 3 fosse a mais confusa, pois

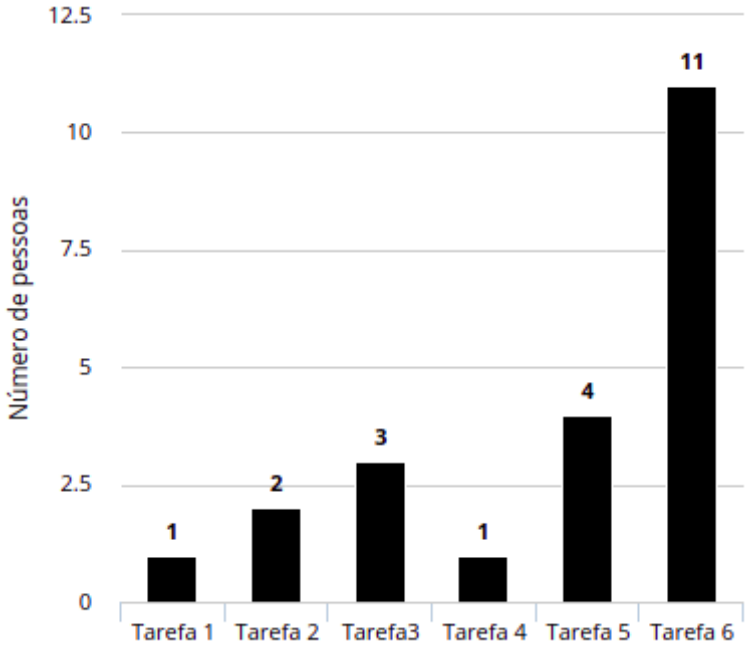
Figura 5.17: Usagem atual do Tracim



Fonte: o autor

a maioria das pessoas não conseguiu achar uma diferença entre versões. Contudo, por ser uma pergunta com resposta esperada e sendo a última do teste, as pessoas se concentraram na tarefa 6. A Figura 5.18 mostra que 11 das 19 pessoas escolheram a última tarefa.

Figura 5.18: Tarefa mais confusa



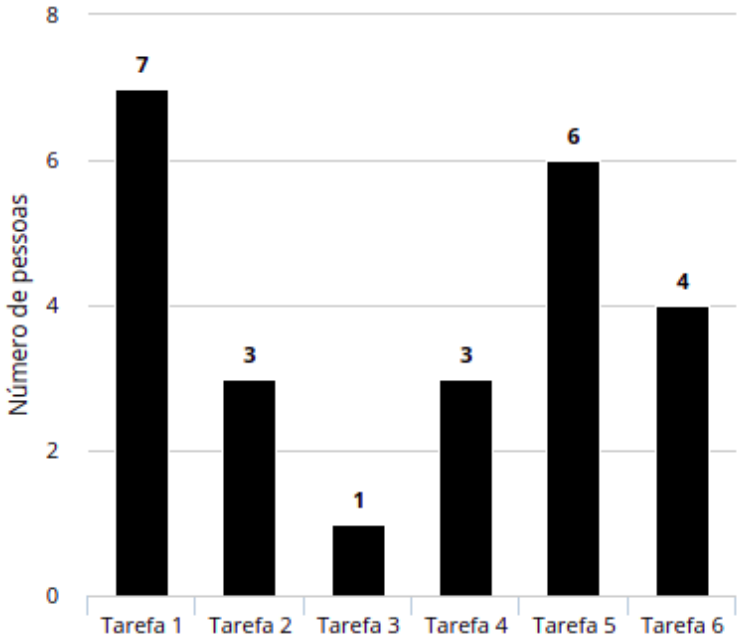
Fonte: o autor

A segunda questão era "**Em que tarefa foi a diferença entre as versões é mais marcante?**", também sendo múltipla escolha entre as tarefas. O maior objetivo era saber quais funcionalidades fazem diferença e/ou estavam suficientemente bem desenvolvidas e quais ainda precisam ser melhor pensadas.

Na Figura 5.19, vemos que claramente a possibilidade de fazer uma busca e as

menções se destacam. Ambas funcionalidades são amplamente utilizadas na *internet*, o que faz com que seja uma funcionalidade básica para os usuários e eles sempre esperam que exista.

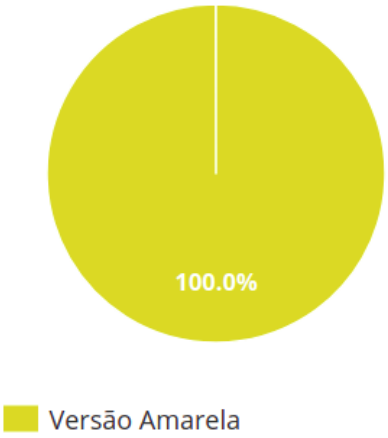
Figura 5.19: Tarefa com maior diferença entre versões



Fonte: o autor

A última pergunta, "**De uma forma geral, em que versão você teve uma experiência melhor na realização das tarefas?**", servia de confirmação de que a experiência de usuário melhorou. Segundo a Figura 5.20, todos os participantes confirmaram essa expectativa.

Figura 5.20: Resultado da melhor experiência de uma forma geral



Fonte: o autor

Mesmo sendo o resultado esperado, é interessante ressaltar que a resposta não foi

automática para todos participantes. Alguns voltaram na lista de tarefas e contaram quantas vezes eles haviam preferido a versão amarela, quantas a rosa e quantas sem diferença.

Se somarmos os votos do participante em “Sem diferença” e “Versão Rosa”, três pessoas tiveram mais votos nessa soma do que na versão amarela e seis tiveram a mesma quantidade. Acredita-se que isso não influenciou na preferência geral pois, nas questões onde houve preferência pela amarela, a diferença era muito mais marcante do que nas que não houve.

6 CONCLUSÃO

Nesse trabalho, foram apresentados os desenvolvimentos e testes com o objetivo de melhorar a experiência de usuário no Tracim. Tracim, de *TRACability IMproved*, é um *software* sob licença livre que visa facilitar a gestão e o trabalho em equipe.

No total foram realizadas correções de *bugs* e seis novas funcionalidades: busca, compartilhamento de arquivos, identificador, *Tracim Live Messages*, notificações e menções. Após o desenvolvimento, foram feitos testes de usuário com uma versão anterior as modificações e uma versão posterior para medir a melhoria.

Foram identificadas algumas evoluções a serem feitas tanto nas funcionalidades em si quanto na metodologia de testes. A seguir, a lista dessas evoluções em relação aos desenvolvimentos:

- Na funcionalidade de busca, seria interessante poder buscar por um usuário da plataforma.
- Muitos participantes gostariam de ter tido acesso à uma página de atividades recentes de um único usuário.
- A funcionalidade de compartilhamento precisa de uma melhor taxa de descoberta. Um botão na página inicial do espaço ajudaria com isso.
- Poder criar um *link* de compartilhamento sem precisar associar um *e-mail*, para não precisar recriar o *link* a cada novo compartilhamento.
- Quando o usuário edita o texto em modo de edição avançada, adicionar um botão que permita adicionar menções no texto. Isso ajudaria a descobrir a funcionalidade mais facilmente.
- Conversar com verdadeiros usuários dos três perfis para analisar suas verdadeiras necessidades.
- Realizar uma análise heurística antes dos próximos desenvolvimentos.

Em seguida, evoluções relativas à metodologia de testes:

- Seria interessante fazer mais perguntas demográficas sobre os participantes, principalmente sobre o quão confortável eles estão com a informática.
- Ter um grupo mais amplo de participantes, tanto em quantidade quanto em diversidade.
- Durante os testes cada tarefa era feita em uma versão e depois na outra, intercalando entre participantes qual era a versão de início. Outra forma interessante seria fazer

as 6 tarefas em uma versão inteira e após na outra, ainda intercalando a versão de início. Isso influenciaria menos a realização da tarefa na segunda versão, porém a preferência das primeiras tarefas seria menos fiável pois seriam influenciado pelas demais.

- Reservar um tempo no início do teste para que o participante utilize o *software* e possa fazer perguntas.
- Realizar as tarefas em ordens diferentes para cada participante, diminuindo um possível viés de ordem.
- Realizar uma análise estatística mais detalhada da diferença de tempo entre versões.

O teste possuía 6 tarefas, das quais 4 tiveram uma preferência pela versão posterior aos desenvolvimentos e 2 não tiveram uma preferência clara. Ao perguntar aos participantes se havia uma preferência geral entre as duas versões, 100% dos participantes preferiram a versão após melhorias. Podemos concluir que tudo o que foi projetado e implementado durante o trabalho teve um impacto positivo no software.

REFERÊNCIAS

ABRAMOV, Dan. **Redux: A Predictable State Container for JS Apps**. 2022. Available from Internet: <https://redux.js.org/>.

ACCORSI, Philippe. **Some buttons is now blue, ex in app file (regression)**. 2019. Available from Internet: <https://github.com/tracim/tracim/issues/1712>.

ALGOO. **Algoo: Éditeur de logiciels collaboratifs innovants**. Moirans: [s.n.], 2021. Available from Internet: <https://www.algoo.fr/>.

ALGOO. **Tracim: Collaboration simple et performante, pour tous**. Moirans: [s.n.], 2021. Available from Internet: <https://www.algoo.fr/fr/tracim/>.

ALGOO. **VisioCall: La visioconférence facile pour tous**. Moirans: [s.n.], 2021. Available from Internet: <https://www.algoo.fr/fr/visioconference-suricate-tv/>.

BOOTSTRAP. **Bootstrap: Build fast, responsive sites with Bootstrap**. 2022. Available from Internet: <https://getbootstrap.com/>.

CALCONNECT. **CalDAV**. 2022. Available from Internet: <https://devguide.calconnect.org/CalDAV>.

CHOPRA, Paras. The Ultimate Guide To A/B Testing. June 2010. Available from Internet: <https://www.smashingmagazine.com/2010/06/the-ultimate-guide-to-a-b-testing/>.

ELASTICSEARCH. **Elasticsearch: O coração do Elastic Stack**. 2022. Available from Internet: <https://www.elastic.co/fr/elasticsearch/>.

FANDOM, Community. **Wiki Os Padrinhos Mágicos**. 2012? Available from Internet: <https://ospadrinhosdetimmy.fandom.com/pt-br/>.

FANOUT. **pushpin: Add push to your API**. 2022. Available from Internet: <https://pushpin.org/>.

FREE SOFTWARE FOUNDATION. **GNU GENERAL PUBLIC LICENSE: Version 3**. 2007. Available from Internet: <https://www.gnu.org/licenses/gpl-3.0.html>.

FREE SOFTWARE FOUNDATION. **GNU LESSER GENERAL PUBLIC LICENSE: Version 3**. 2007. Available from Internet: <https://www.gnu.org/licenses/lgpl-3.0.html>.

GITHUB. **gbonaspetti**. 2022. Available from Internet: <https://github.com/gbonaspetti/>.

GITHUB. **tracim: Threads, files and pages with status and full history. All in the same place.** 2022. Available from Internet: <https://github.com/tracim/tracim>.

IO, Cypress. **Cypress: The web has evolved. Finally, testing has too**. 2022. Available from Internet: <https://www.cypress.io/>.

JITSI. **Jitsi Meet: Free Video Conferencing Solutions**. 2022. Available from Internet: <https://jitsi.org/jitsi-meet/>.

KRUG, S. **Não me faça pensar: Uma abordagem de bom senso à usabilidade na web e mobile**. 2. ed. Rio de Janeiro: Alta Books, 2014. ISBN 9788576088509.

MARTINS, Giulia B. **Hide the archiving function**. 2019. Available from Internet: <https://github.com/tracim/tracim/issues/2347>.

MARTINS, Giulia B. **Make the menu inside the apps fully clickable and add a tooltip**. 2019. Available from Internet: <https://github.com/tracim/tracim/issues/2275>.

MARTINS, Giulia B. **Sidebar breaks in Administration when we have a big description for a Shared space**. 2019. Available from Internet: <https://github.com/tracim/tracim/issues/1781>.

META, Plataforms. **React: A JavaScript library for building user interfaces**. 2022. Available from Internet: <https://reactjs.org/>.

MOZILLA. **DOMParser: Web APIs**. 2022. Available from Internet: <https://developer.mozilla.org/en-US/docs/Web/API/DOMParser>.

NIELSEN, Jakob. Why You Only Need to Test with 5 Users. Mars 2000. Available from Internet: <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>.

NIELSEN, Jakob. Quantitative Studies: How Many Users to Test? June 2006. Available from Internet: <https://www.nngroup.com/articles/quantitative-studies-how-many-users/>.

NORMAN, D.; MILLER, J.; HENDERSON, A. What you see, some of what's in the future, and how we go about doing it: Hi at apple computer. In: . CHI '95: Conference Companion on Human Factors in Computing Systems, 1995. Available from Internet: <https://www.researchgate.net/publication/202165701_What_You_See_Some_of_What's_in_the_Future_And_How_We_Go_About_Doing_It_HI_at_Apple_Computer>.

OPEN SOURCE INITIATIVE. **The MIT License**. 2022. Available from Internet: <https://opensource.org/licenses/MIT>.

OPENJS. **Mocha: simple, flexible, fun**. 2022. Available from Internet: <https://mochajs.org/>.

REMIX. **React Router**. 2022. Available from Internet: <https://reactrouter.com/>.

SEVAJOL, Bastien; HUGUIÈS, Côme. **When edit html document TAB is dangerous**. 2019. Available from Internet: <https://github.com/tracim/tracim/issues/1409>.

SMARTBEAR. **Swagger UI: REST API documentation tool**. 2021. Available from Internet: <https://swagger.io/tools/swagger-ui/>.

SOLLOWAY, Ashley; NICHOLS, Timothy; COOLRIDGE, Mariah; SKILLMAN, Ellis. Usability testing report. 2022. Available from Internet: <https://library.xtensio.com/usability-testing-report-template-and-examples>.

STANDARDIZATION, I. O. for. **ISO 13407:1999: Human-centred design processes for interactive systems**. 1. ed. [S.l.], 1999. Available from Internet: <https://www.iso.org/standard/21197.html>.

STANDARDIZATION, I. O. for. **ISO 9241-210:2010: Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems**. 1. ed. [S.l.], 2010. Available from Internet: <https://www.iso.org/standard/52075.html>.

STANDARDIZATION, I. O. for. **ISO 9241-210:2019: Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems**. 2. ed. [S.l.], 2019. Available from Internet: <https://www.iso.org/standard/77520.html>.

STYLUS. **Stylus: Foresight has never been so crucial**. 2022. Available from Internet: <https://www.stylus.com/>.

TINY. **TinyMCE: The most advanced WYSIWYG HTML editor**. 2022. Available from Internet: <https://www.tiny.cloud/>.

WHITEHEAD. **WebDAV Resources**. 2010. Available from Internet: <http://webdav.org/>.

YAKFORMS. **Framaforms**. 2022. Available from Internet: <https://framaforms.org/>.

ČIHAŘ, Michal. **Weblate: Localização contínua baseada na web**. 2012–2022. Available from Internet: <https://weblate.org/>.

APÊNDICE A — FORMULÁRIO DE TESTE

Questão 1: Versão de início

- Tipo: Escolha única, obrigatória e única questão preenchida pela pessoa que aplica o teste.
- Opções:
  - Versão Rosa
  - Versão Amarela

Questão 2: Você já usou Tracim?

- Tipo: Escolha única, obrigatória.
- Opções:
  - Sim
  - Não

Questão 3: Quando começou a usar o software? Ano

- Tipo: Escolha única em seletor, somente visível se a resposta da questão 2 é Sim.
- Opções:
  - Antes de 2019
  - 2019
  - 2020
  - 2021
  - 2022

Questão 4: Quando começou a usar o software? Mês

- Tipo: Escolha única em seletor, somente visível se a resposta da questão 3 é diferente de Antes de 2019.
- Opções:
  - Janeiro
  - Fevereiro
  - Março
  - Abril
  - Maio

- Junho
- Julho
- Agosto
- Setembro
- Outubro
- Novembro
- Dezembro

Questão 5: Com que frequência você utiliza Tracim atualmente?

- Tipo: Escolha única, somente visível se a resposta da questão 2 é Sim.
- Opções:
  - Diariamente
  - Semanalmente
  - Mensualmente
  - Nunca

Tarefa 1: 1. Abra o conteúdo Juandissimo / Gnomes.

Questão 5: 1. Em qual versão essa tarefa foi mais fácil de fazer?

- Tipo: Escolha única.
- Opções:
  - Versão Rosa
  - Versão Amarela
  - Sem diferença

Tarefa 2: 2. Compartilhar o arquivo "Anti Cosmo" com o e-mail cosmo@cosmo.cosmo.

Questão 6: 2. Em qual versão essa tarefa foi mais fácil de fazer?

- Tipo: Escolha única.
- Opções:
  - Versão Rosa
  - Versão Amarela
  - Sem diferença

Tarefa 3: 3. Uma pessoa que não tem uma conta no Tracim quer compartilhar um arquivo com todos os membros do espaço Cosma-Fairywinkle family, mas não sabe seus

detalhes de contato. Como você ajudaria essa pessoa?

Questão 7: 3. Em qual versão essa tarefa foi mais fácil de fazer?

- Tipo: Escolha única.
- Opções:
  - Versão Rosa
  - Versão Amarela
  - Sem diferença

Tarefa 4: 4. No arquivo Disguise, converse entre Cosmo e Wanda. Exemplo: “O que você acha desse disfarce?” “Acho ótimo!” “Ok, vamos fazê-lo essa noite.”

Questão 8: 4. Em qual versão essa tarefa foi mais fácil de fazer?

- Tipo: Escolha única.
- Opções:
  - Versão Rosa
  - Versão Amarela
  - Sem diferença

Tarefa 5: 5. Peça para Timmy Turner ler o documento Synopsis.

Questão 9: 5. Em qual versão essa tarefa foi mais fácil de fazer?

- Tipo: Escolha única.
- Opções:
  - Versão Rosa
  - Versão Amarela
  - Sem diferença

Tarefa 6: 6. Qual foi o último conteúdo comentado por Vicky/Timmy?

Questão 10: 6. Em qual versão essa tarefa foi mais fácil de fazer?

- Tipo: Escolha única.
- Opções:
  - Versão Rosa
  - Versão Amarela
  - Sem diferença

Questão 11: Qual foi a tarefa mais confusa do teste?

- Tipo: Múltipla escolha.
- Opções:
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6

Questão 12: Em que tarefa foi a diferença entre as versões é mais marcante?

- Tipo: Múltipla escolha.
- Opções:
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6

Questão 13: De uma forma geral, em que versão você teve uma experiência melhor na realização das tarefas?

- Tipo: Escolha única.
- Opções:
  - Versão Rosa
  - Versão Amarela
  - Sem diferença