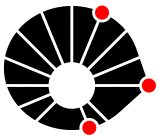
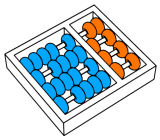




# Qualitative Data Analytics: Desenvolvimento de ferramenta para análise qualitativa de dados

*F. C. Marques* *B. B. N. França*

Relatório Técnico - IC-PFG-20-32
Projeto Final de Graduação
2020 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

# Qualitative Data Analytics: Desenvolvimento de ferramenta para análise qualitativa de dados

Filipe Cavalheiro Marques, Breno Bernard Nicolau de França

Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Caixa Postal 6176
13083-970 Campinas-SP, Brasil

f148524@dac.unicamp.br, breno@ic.unicamp.br

**Resumo.** A análise de dados qualitativos é uma etapa essencial para pesquisas que envolvem dados majoritariamente descritivos, e pode ser apoiada por uma gama de soluções tecnológicas disponíveis no mercado. No entanto, produtos estáveis e eficientes são escassos e, quando existem, têm custos impeditivos para pesquisadores e estudantes. Neste projeto, foram desenvolvidas funcionalidades para uma nova aplicação *web open-source* de análise de dados qualitativos e que contemplam os procedimentos realizados durante a fase de *open coding* de uma pesquisa.

# 1. Introdução

Pesquisas qualitativas são essenciais para a construção de conhecimento e produção científica em áreas que tratam de dados descritivos, a exemplo de textos e dados de mídia. As conclusões que se pode tirar desse tipo de informação geralmente são realizadas através de abstrações e interpretações destes dados, emergindo comportamentos e fenômenos da situação em que foram obtidos, como o uso de termos específicos durante uma entrevista identificados durante sua transcrição. Dá-se à investigação desse tipo de dado o nome de análise de dados qualitativos.

A análise qualitativa pode ser apoiada por soluções de software disponíveis no mercado e que baseiam suas features nos procedimentos sugeridos pelos métodos de análise existentes na literatura. No entanto, pode-se perceber uma escassez de produtos gratuitos e confiáveis, já que os software amplamente utilizados e que se sobressaem têm custos tão elevados que sua aquisição se torna inviável para muitos estudantes, pesquisadores e empresas. Por isso, foi visto como oportunidade construir um produto *open-source* que atenda às demandas de pesquisa e análise destes usuários.

Este projeto tem o objetivo de construir uma ferramenta com funcionalidades voltadas a uma das primeiras etapas desta análise de dados, a codificação e categorização de conceitos durante a codificação aberta (do inglês, *open coding*) [1].

Este relatório está dividido da seguinte forma: a Seção 2 apresenta os principais conceitos da análise de dados qualitativos que procurei contemplar na construção do projeto, e mostra um panorama geral dos principais softwares atualmente disponíveis e seus custos de aquisição. A Seção 3 apresenta os métodos utilizados na descoberta do problema e na construção de sua solução. A Seção 4 discute em profundidade as funcionalidades desenvolvidas, como elas se relacionam aos conceitos teóricos de análise, e as tecnologias e arquitetura utilizadas. A Seção 5 traz as considerações finais e possíveis trabalhos a serem realizados no futuro com base no projeto apresentado.

# 2. Fundamentação teórica

## 2.1. Análise de dados qualitativos

A análise de dados qualitativos é baseada primariamente na conceitualização e rotulação de fontes de dados diversas, com o intuito de detectar conceitos recorrentes e outras abstrações, a fim de relacioná-los, permitindo inferir hipóteses e, possivelmente, obter uma explicação para as mesmas. Este tipo de análise é especialmente importante para ramos como ciências sociais, educação, interação humano-computador, que têm suas bases pautadas primariamente por artefatos qualitativos como entrevistas e análises comportamentais.

O foco deste projeto é a etapa de *open coding* [1], apresentada na literatura como um processo analítico de identificação e categorização de conceitos baseado em suas propriedades e dimensões. É importante ressaltar que, por mais que estejamos seguindo um procedimento específico de pesquisa qualitativa, o *open coding* é bastante similar em diferentes métodos de análise qualitativa [1][2].

O foco desta etapa está na conceitualização de elementos, ou seja, a ação de identificar conteúdos diversos de fontes de pesquisa, como eventos, passagens e ações, associando-os aos códigos, conceitos que abrangem estes elementos com certa abstração. É imprescindível que o pesquisador seja capaz de identificar facilmente de que se trata um código criado para que possa relacioná-lo a categorias e outros códigos posteriormente, sem a necessidade de revisitar constantemente todas as fontes de pesquisa.

Visto que um pesquisador pode identificar dezenas de códigos relevantes ao longo de suas investigações, é importante que esses conceitos sejam agrupados sob categorias, abstrações advindas dos dados identificados durante a conceitualização. É importante que o nome de uma categoria seja claro e marcante, facilitando assim recordar de que esta se trata. Categorias podem ser divididas em subcategorias, trazendo níveis de especificidade das informações disponíveis em um código, como momento e circunstância que ele pode ocorrer em relação à categoria em que está presente.

Percebe-se a grande necessidade de relembrar constantemente de que se trata cada elemento da investigação, seja ele um código ou categoria. Para auxiliar neste processo é

comum que pesquisadores utilizem de *memos*, registros de seus pensamentos, ideias, questões e conclusões que podem auxiliá-los na análise posterior dos dados e próximos passos de sua pesquisa.

Em posse dos artefatos descritos, é possível dar continuidade à pesquisa qualitativa, através dos processos como *axial coding* e *selective coding*, não incluídos no escopo deste projeto.

## 2.2. Panorama de produtos disponíveis no mercado

Uma das principais motivações para a realização deste projeto está no panorama atual de produtos digitais com foco em apoiar pesquisas qualitativas, os chamados CAQDAS (*Computer-assisted qualitative data analysis software*). O mercado atual tem à disposição poucas ferramentas de qualidade e com custo bastante elevado. Como referências disso, temos o *ATLAS.ti* [3] (licenças educacionais a partir de R$ 117,66 ao mês por usuário em sua versão cloud), o *MAXQDA* [4] (licença educacional a partir de US$ 469,80 anuais para um mínimo de três estudantes) e o *NVivo* [5] (licença acadêmica vitalícia para um estudante a partir de US$ 849,00). Estes valores já altos podem ser impeditivos especialmente para institutos de pesquisa e profissionais independentes. Importante ressaltar que na pesquisa qualitativa é passível o viés de interpretação. Por isso, é sempre recomendado que essa pesquisa não seja realizada por uma única pessoa, aumentando assim a quantidade de licenças.

Outra ferramenta que deve-se comentar também é o *QDA Miner* [6]. Esta é a única a oferecer camada gratuita de seu serviço em sua versão *Lite*. Entretanto, sabe-se que ele é bastante instável, o que eventualmente causa perda de fontes de pesquisa e marcações realizadas. Em uma pesquisa baseada integralmente nesses dois elementos, isso pode gerar perdas consideráveis de esforço investido.

# 3. Métodos

## 3.1. Revisão bibliográfica e estudos dos requisitos do problema

A primeira etapa do projeto foi a compreensão de conceitos essenciais relacionados à análise qualitativa de dados, em especial quanto à etapa de *open coding*, segundo a metodologia de pesquisa apresentada na literatura. Em paralelo a esse estudo, analisei a documentação funcional do PipocaQDA [7], projeto de software livre com a intenção de construir uma ferramenta do tipo CAQDAS. Essa especificação foi fornecida no início do projeto com o intuito de apoiar a construção e definição da arquitetura da solução. Em posse dela, foi possível relacionar os conceitos oferecidos na bibliografia às funcionalidades esperadas ao fim do processo de desenvolvimento.

## 3.2. Pesquisa e desenvolvimento da solução

Com uma base sólida dos conceitos acima, foi necessário iniciar a definição das tecnologias a serem usadas. Após pesquisas que levaram em conta escalabilidade, custo e simplicidade de desenvolvimento, decidiu-se por desenvolver uma aplicação web criada sobre a plataforma Angular [8] e baseada em TypeScript [9]. Para armazenamento de dados, foi utilizado  o banco de dados Cloud Firestore [10], um banco NoSQL em nuvem disponível na plataforma Firebase [11], oferecida pela Google. O resultado deste projeto pode ser encontrado no repositório *Qualidata*[1], no Github.

# 4. Solução Proposta

## 4.1. Interface com o usuário e fluxo de navegação

Durante a etapa de *open coding* em uma pesquisa qualitativa, o usuário deve ser capaz de realizar as seguintes operações principais:

- Adicionar uma fonte de pesquisa textual;
- Codificar trechos de uma fonte;
- Visualizar os trechos codificados de uma fonte de pesquisa;

[1] Disponível em https://github.com/filipemarques33/qualidata

- Visualizar todos os trechos relacionados a um código;
- Categorizar códigos criados;

Para garantir que todas essas funcionalidades sejam contempladas e que um usuário possa realizá-las com foco inequívoco a cada operação, a navegação de um usuário foi projetada como visto no fluxo de navegação apresentado na Figura 1.

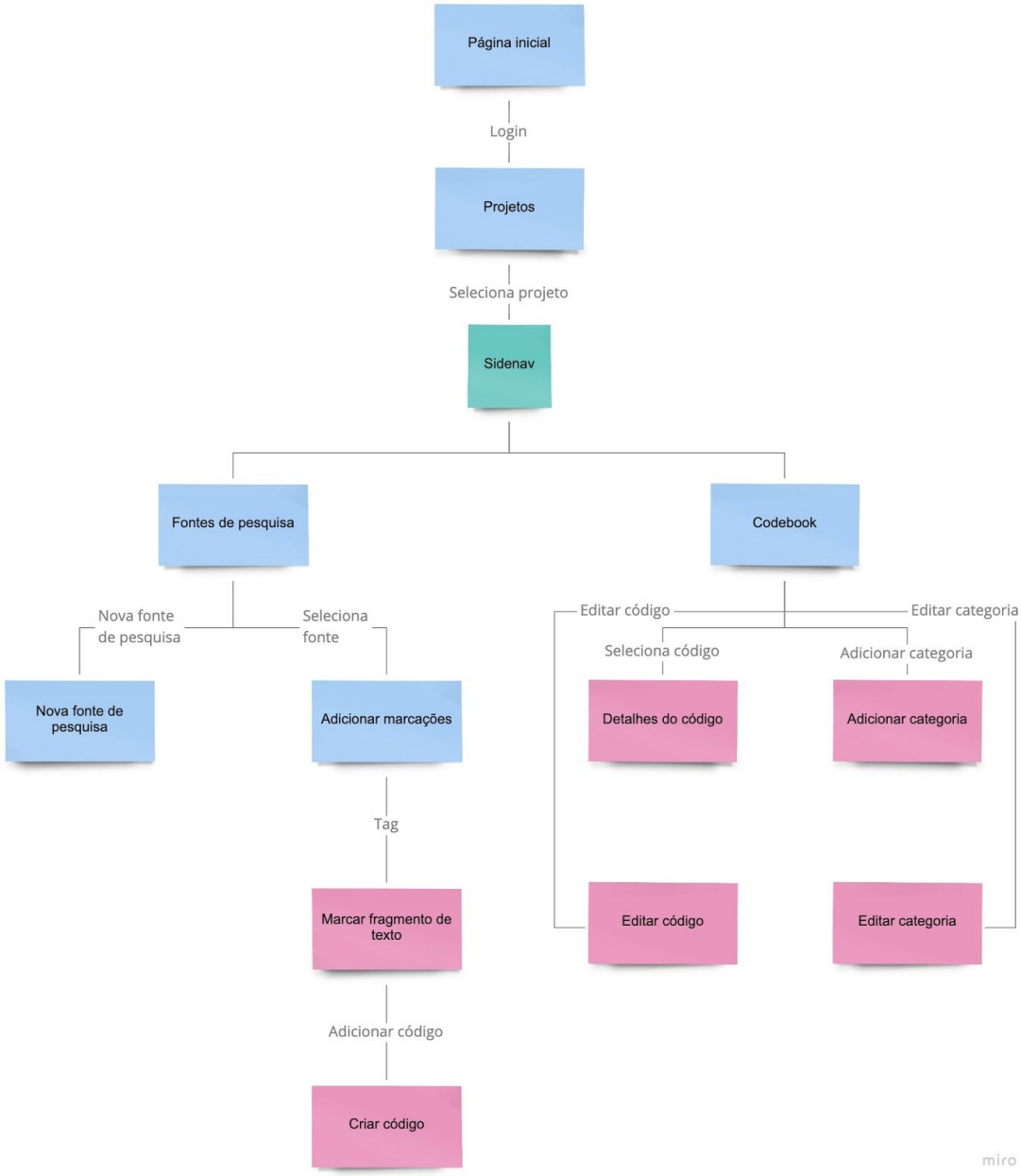



Figura 1: Diagrama de fluxo de navegação na aplicação.

Na Figura 1, os nós azuis representam interfaces inteiras Os nós rosas representam caixas de diálogo, e o nó verde refere-se  a um elemento de navegação “barra lateral” para alternar entre as seções primárias. Os rótulos das arestas indicam o nome do botão ou ação necessária para direcionar o usuário de uma interface para outra.

As especificações das interfaces que realizam essas operações e a forma como elas se relacionam ao processo de Open Coding serão listadas na seção a seguir.

## 4.2. Interfaces Gráficas com o Usuário

### 4.2.1. Fontes de pesquisa

A seção de fontes de pesquisa atua como o repositório de todos os documentos textuais referentes a uma mesma pesquisa, conforme apresentado na Figura 2.

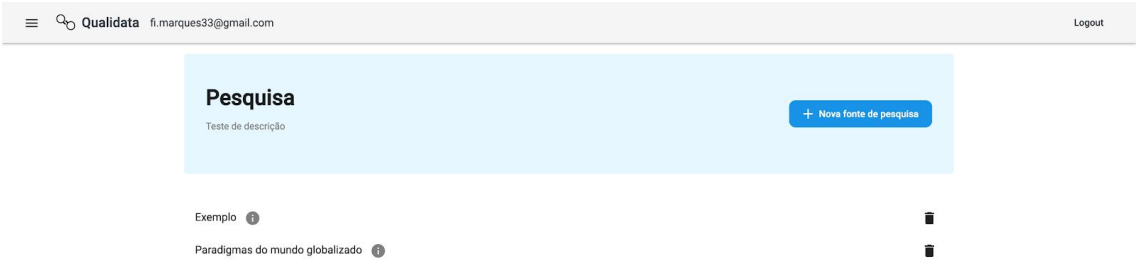



Figura 2: Página de fontes de pesquisa

A listagem dessas informações é simples e as interações possíveis na interface são:

- Nova fonte de pesquisa: direciona para a página de criação de uma nova fonte.
- Selecionar uma das fontes: direciona para a página de marcação de códigos em uma página.

### 4.2.2. Nova fonte de pesquisa

Na página de adição de uma nova fonte de pesquisa (Figura 3), o usuário pode adicionar sua fonte textual à pesquisa através do editor de texto do TinyMCE [12]. É nesse momento que pode-se identificar as vantagens de utilização de um editor WYSIWYG. Não só o usuário pode formatar um texto redigido por ele, sem a necessidade de compreender a fundo o funcionamento de uma página HTML, como poderá copiar e colar textos de fontes

externas diretamente para a página com pouca ou nenhuma perda de formatação em relação ao documento original.

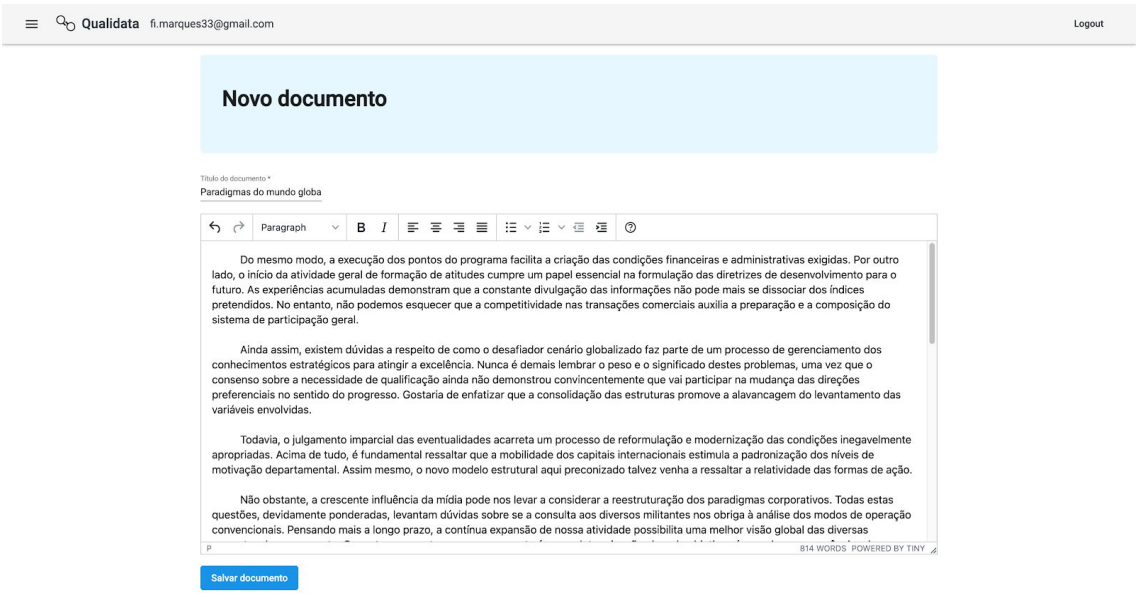



Figura 3: Página de nova fonte de pesquisa

Como decisão de projeto, nesta interface ainda não é possível marcar o documento com seus códigos. Essa decisão foi tomada para garantir que este documento já esteja criado no banco antes do início das codificações e, assim, tenha um identificador para ser referenciado nos fragmentos de texto criados.

- Nova fonte de pesquisa: direciona para a página de criação de uma nova fonte.
- Selecionar uma das fontes: direciona para a página de marcação de códigos em uma página.

### 4.2.3. Ver e adicionar marcações a uma fonte

Esta página talvez possa ser considerada a mais complexa e a mais importante do projeto (Figura 4). É aqui que ocorrem as principais interações necessárias para se salvar um fragmento de texto a uma fonte de pesquisa e a um código relacionado, o que é equivalente à etapa de conceitualização do *open coding*.


Figura 4: Página de visualização e adição de marcações

Nessa página, o pesquisador deve fazer a seleção de um trecho de texto e iniciar o processo de marcação ao clicar no botão "Tag" no canto direito do banner, acima do texto. Em seguida, é realizado o tratamento do fragmento selecionado utilizando a API de manipulação de um editor do TinyMCE. Ao utilizar a função

```
tinymce.activeEditor.selection.getSel().getRangeAt(0)
```

é retornado um elemento Range [13], um elemento DOM que referencia especificamente os nós de início e fim da seleção atual e seus respectivos deslocamentos dentro de cada nó. Um dos grandes desafios deste projeto foi identificar uma maneira de restaurar este elemento Range, uma vez que um nó não pode ser facilmente salvo num banco com formato chave-valor. Para isso, ambos os nós de início e fim foram convertidos em expressões *XPath* [14], strings que utilizam uma estrutura de caminho para percorrer um documento de marcação XML. Com isso, posteriormente, é possível restaurar a seleção do fragmento em questão através da função *evaluate* disponível para componentes Document, e reaver o nó salvo. Mas, no momento da marcação de um trecho, é enviado para o modal que vai realizar a marcação em códigos específicos um objeto Fragment, que contém as informações de Range e conteúdo textual da seleção.

A página de codificação também é a responsável por exibir os trechos de texto deste documento que foram marcados e relacionados a códigos da pesquisa. Para isso, ao lado do editor de texto, foi adicionado um painel que renderiza dinamicamente, para cada fragmento deste documento, um componente de marcação que indica a posição e altura aproximadas do fragmento de texto em questão, bem como uma lista com todos os códigos ligados a este fragmento. Ao passar o cursor por cima desse componente, ele também é capaz de selecionar o fragmento diretamente no editor.

Por uma complexidade técnica elevada, não foi possível contemplar a edição de documento nesta tela, pois isso interferiria em todos os fragmentos de um texto, potencialmente alterando suas propriedades de Range e conteúdo, simultaneamente.

### 4.2.4. Marcar fragmento de texto

Com o fragmento obtido da etapa anterior, nesta interface (Figuras 5 e 6) o pesquisador pode selecionar da lista de todos os códigos disponíveis, aqueles que considera ter maior relação com este conteúdo. Para isso, é exibido um menu de seleção múltipla, que utiliza um componente externo chamado NgxMatSelectSearch [15] para realizar a busca nessa lista.

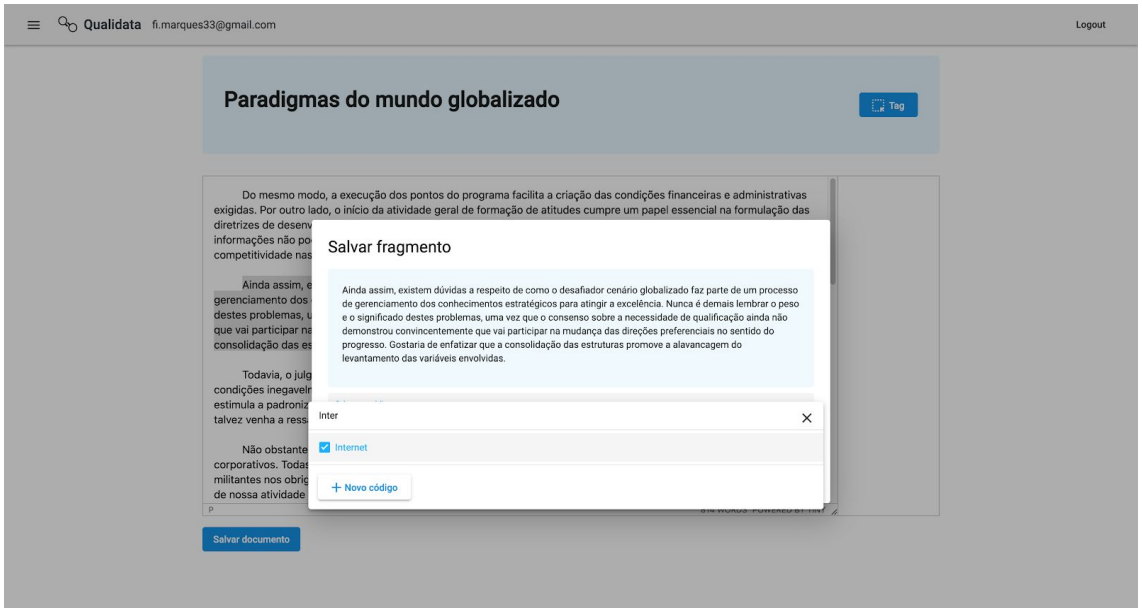
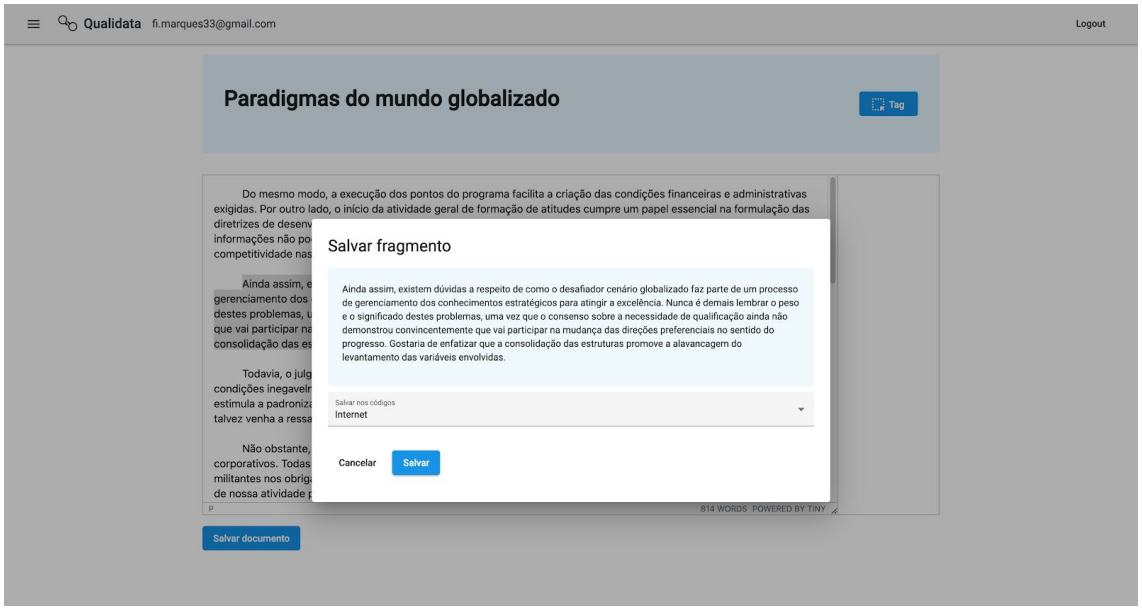



Figura 5: Modal de marcação de fragmento, demonstrando onde é possível adicionar novos códigos

O usuário também é capaz de solicitar a adição de novos códigos através do botão "Novo código" no rodapé da lista (Figura 5).




Figuras 6: Modal de marcação de fragmento preenchido

### 4.2.5. Criar novo código

Durante o processo de marcação, caso o pesquisador precise definir um novo código, ele o pode fazer diretamente durante a etapa de marcação, ao ativar este modal (Figura 7). Aqui, o usuário poderá definir o nome, cor, descrição e a categoria à qual este código pertence.

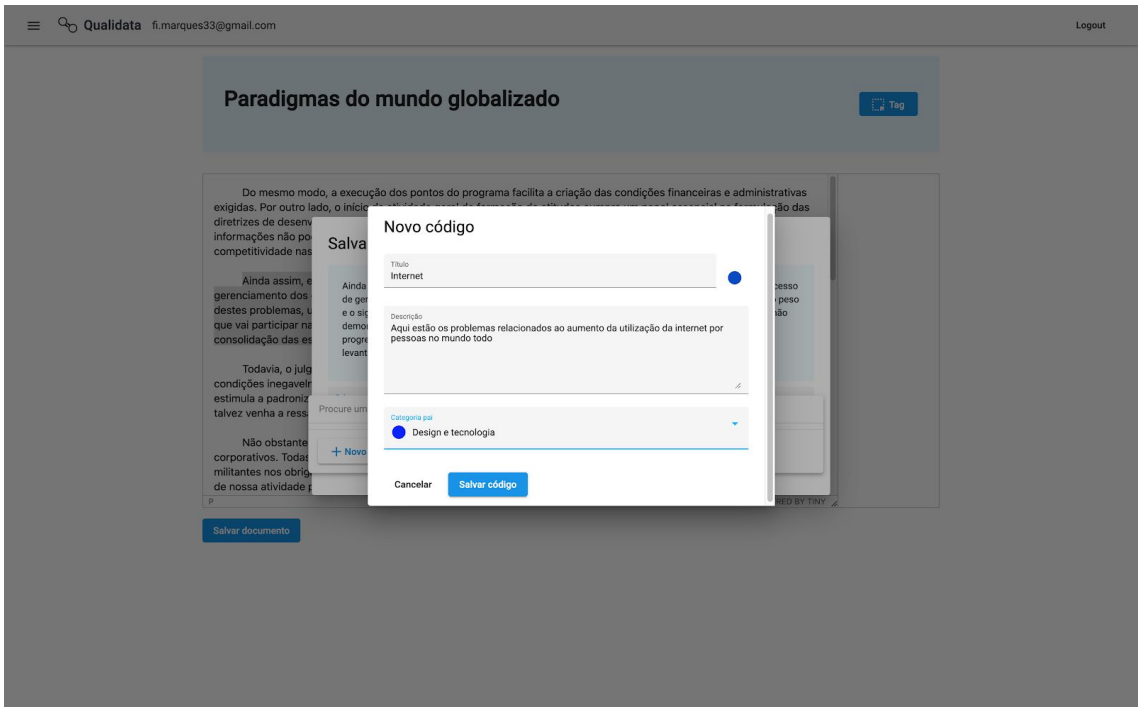



Figura 7: Modal de criação de código

Conceitualmente, a descrição de um código funciona de maneira bastante similar a um memo do processo de análise de dados. Ambos buscam ser descrições textuais e trazer pontos de atenção, conclusões e questionamentos acerca do código descrito.

Além disso, a definição da cor não é meramente um valor estético ou de identificação em meio à lista. Durante a etapa de *axial coding*, posterior ao *open coding*, esta cor é a que definirá a exibição deste código na rede de relacionamentos entre códigos e categorias.

Por fim, este novo código só é salvo em banco de dados se ele for selecionado como um dos códigos do fragmento do modal anterior. Este comportamento é esperado porque, para a análise de dados qualitativos, a criação de códigos sem um conteúdo que os justifique pode enviesar a pesquisa.

### 4.2.6. Codebook

O codebook é a seção onde pode-se visualizar todos os códigos e categorias criados durante uma pesquisa. A visualização desta lista é uma árvore destes componentes de maneira a refletir a ordenação em categorias, subcategorias e seus códigos (Figura 8).

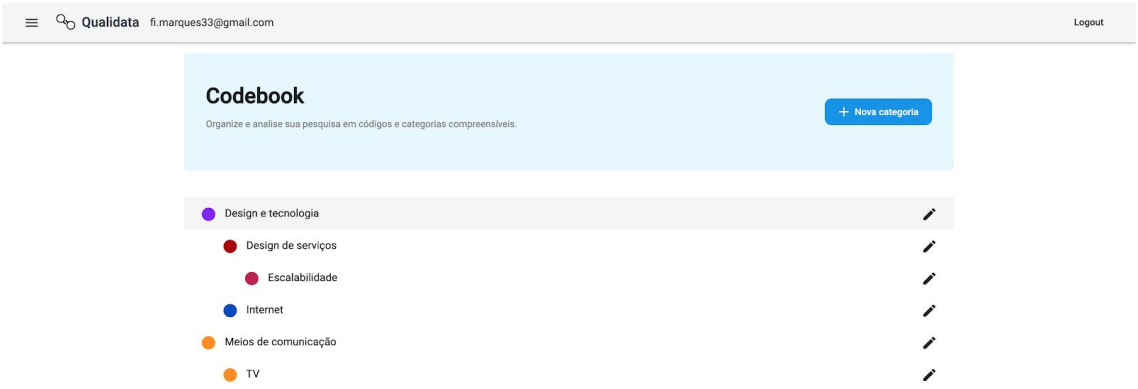



Figura 8: Página de Codebook

Aqui, o usuário tem quatro possibilidades de interação:

- Ativar o modal de adicionar categoria ao clicar no botão "Nova categoria";
- Editar uma categoria ao clicar no botão de edição ao lado de um categoria;
- Ver todos os códigos relacionados a um código ao interagir com sua célula;
- Editar um código ao clicar no botão de edição ao lado de um código.

### 4.2.7. Nova categoria

Pensando na fase de categorização do *open coding*, é oferecida ao usuário a possibilidade de adicionar categorias à pesquisa (Figura 9). Uma categoria pode ter um nome, descrição, cor e a categoria à qual pertence. Este último campo garante a possibilidade de criar uma estrutura com categorias e subcategorias.

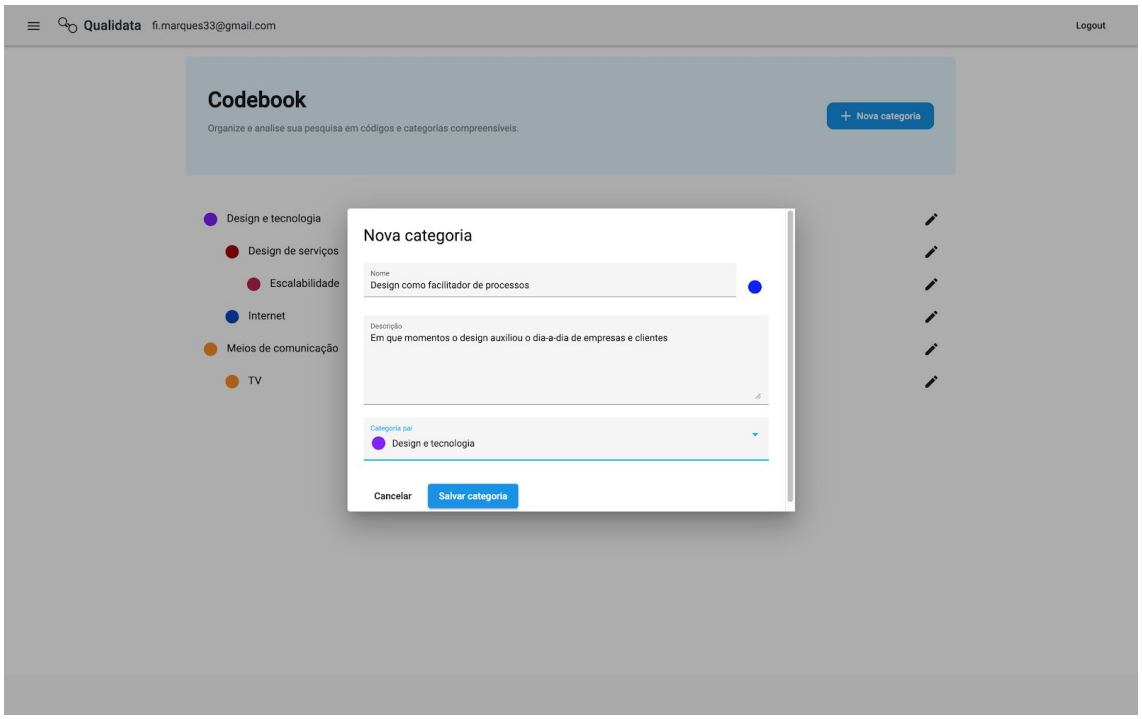



Figura 9: Página de nova fonte de pesquisa

Da mesma forma que no código, a descrição aqui serve a um comportamento similar ao de *memos*, e a cor desta categoria será importantíssima para a etapa de *axial coding*.

É importante ressaltar também que, da mesma maneira que um código, uma categoria sem conteúdos adicionados pode ser prejudicial à pesquisa. Contudo, a categoria tem uma relação muito mais fraca em relação a outras categorias e códigos do que uma relação entre códigos e fragmentos. Pensando nisso, e para simplificar algumas etapas do fluxo de adição de um código, foi decidido seguir com esse comportamento na versão atual.

### 4.2.8. Detalhes do código

Ao interagir com um código, é exibida uma nova visualização na tela do codebook (Figura 10). Esta tem como objetivo consolidar todos fragmentos de texto que foram conectados a esse código. Isso permite um panorama geral de todo o conteúdo que levou à criação e análise desse dado como relevante.

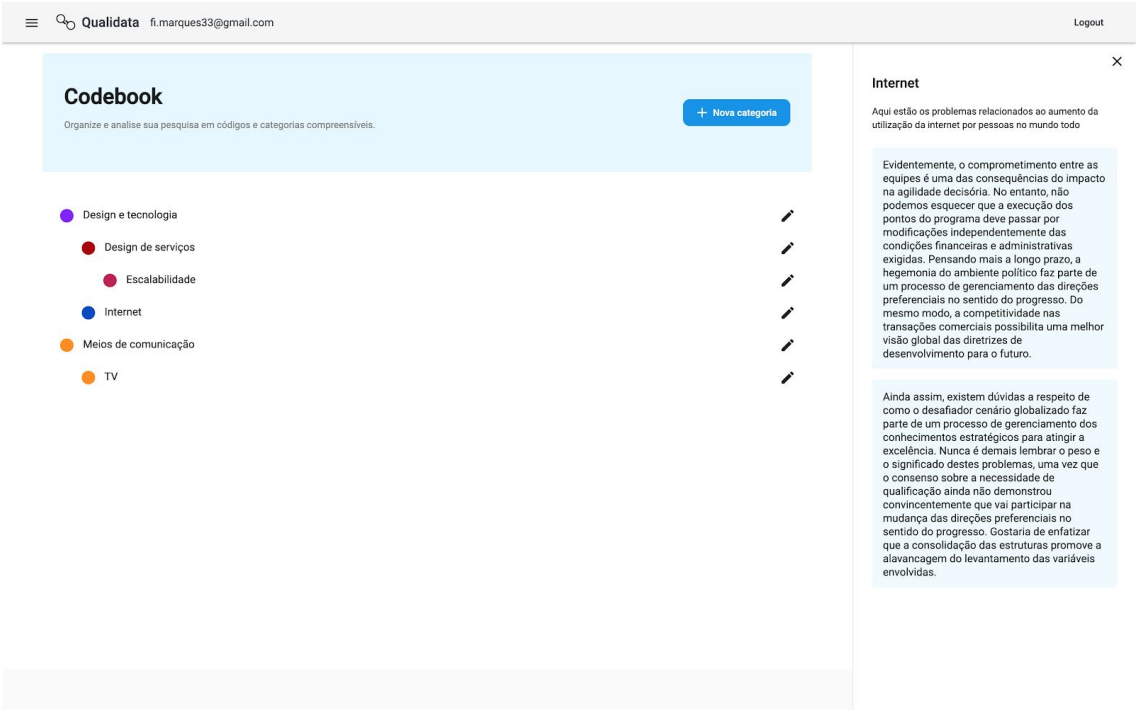



Figura 10: Página de nova fonte de pesquisa

Se tiver alguma dúvida específica referente ao contexto de algum dos códigos ele pode clicar em algum dos trechos da lista para ser redirecionado para o documento onde este fragmento pode ser encontrado.

### 4.2.9. Editar código / Editar categoria

Ao clicar nos botões de edição de cada uma das linhas dos códigos ou categorias, é disparado o modal referente à edição de cada um deles. Seus comportamentos e campos são os mesmos encontrados nos respectivos modais de criação, mas as edições têm características importantes para o projeto em sua versão atual, pois em permitem que sejam adicionados adendos à descrição conforme novos fragmentos são codificados, e garantem a possibilidade de rearranjar as categorias e códigos, mesmo sem a funcionalidade de se fazer isso via *drag and drop*.

## 4.3. Tecnologias Utilizadas

Como descrito acima, a solução proposta para a criação de um sistema de análise de dados qualitativos open-source foi o desenvolvimento de uma aplicação *web*. Porém, a decisão de qual biblioteca ou plataforma utilizar deu-se pela análise das ferramentas disponíveis e que tivessem melhores integrações entre si.

Como esta análise é dada através de fontes de pesquisa diversas, como textos, áudios, vídeos, fotografias, entre outros, foi necessário priorizar a construção de um produto mínimo que pudesse realizar as etapas do *Open Coding*. Portanto, para este primeiro momento, a decisão foi por construir um sistema que pudesse analisar documentos de texto e analisar seus fragmentos em conceitos de pesquisa. Ficou claro, assim, que seria essencial um editor de *rich text* gratuito e de integração simples à plataforma. Entre as soluções disponíveis, as que mais se destacaram foram o TinyMCE [12], com versão disponível primariamente para Angular, e o Draft.js [16], disponível em React. Ambos são editores de texto do tipo "*What You See Is What You Get*" (WYSIWYG), ou seja, que permitem a visualização do conteúdo de texto formatado em tempo real e sem a necessidade do usuário final ter conhecimento prévio de HTML.

Também era sabido que, por simplicidade e agilidade, era ideal utilizar uma solução de banco de dados que não precisasse do desenvolvimento de um back-end, mas que ao mesmo tempo fosse viável para a filosofia open-source proposta. Eram candidatos o SQLite [17], MongoDB [18] e Cloud Firestore [10]. Após algumas pesquisas preliminares, confirmei que era necessário a implementação de um back-end para as duas primeiras, e com isso escolhi a Cloud Firestore como banco de dados para o projeto. Este é um banco de dados NoSQL presente na plataforma Firebase, gerida e desenvolvida pela Google. Apesar de ser uma ferramenta paga, ela ainda possui um plano gratuito razoável que deve permitir as interações dos primeiros usuários com a plataforma.

Nessa fase, também foi essencial identificar a biblioteca de interface a ser utilizada, e uma das que mais se destacou foi a Angular Material [19], uma biblioteca de componentes desenvolvidos nativamente para Angular e em acordo com as guidelines do Material Design [20]. Entre suas vantagens, estavam a facilidade de instalação com um

aplicativo desenvolvido sobre Angular, aplicação rápida de seus componentes às interfaces da plataforma e potencial familiaridade para usuários, principalmente aqueles que já estão habituados com produtos da Google, como Gmail e Google Drive.

Com isso, tornou-se bastante claro que o melhor *framework* para iniciar o desenvolvimento do produto foi o Angular [8]. Para trazer algum contexto, Angular é uma plataforma *open-source* também fornecida e mantida pela Google e com colaboração de diversos desenvolvedores independentes e focada no desenvolvimento web. Seu desenvolvimento é baseado em TypeScript [9], extensão tipada do JavaScript [21].

A colaboração de desenvolvedores e empresas diversas na comunidade também é uma grande vantagem na utilização do Angular, pois são inúmeras as possibilidades de desenvolvimento utilizando as várias bibliotecas disponibilizadas por eles. Algumas delas foram essenciais para o desenvolvimento em tempo hábil deste projeto.

## 4.4. Arquitetura

Como não se fez necessária a implementação de um *back-end* e todas as chamadas ao banco de dados foram resolvidas diretamente com o serviço *Cloud Firestore*, a arquitetura do sistema (Figura 11) tornou-se bastante simples, demandando apenas dos componentes para renderização e controle da interface com o usuário; uma camada de serviços para operações diversas referente às classes criadas, como obtenção e tratamento das mesmas antes de seu envio para os componentes de interface; e serviços de *storage*, responsáveis pela realização das chamadas ao banco de dados.

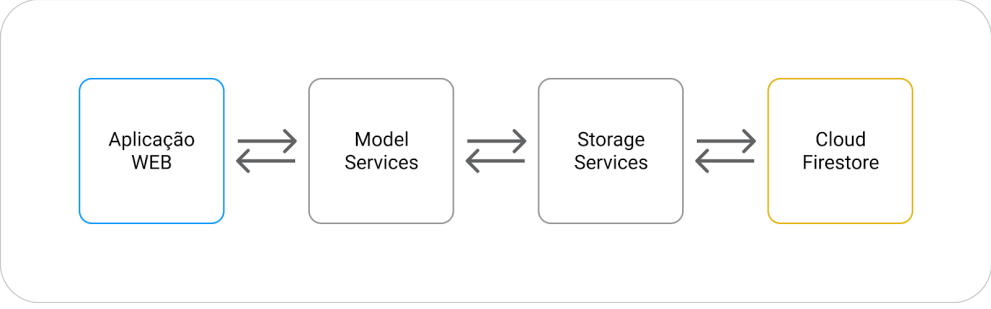



Figura 11: Diagrama simplificado da arquitetura do projeto

A decisão de organizar a arquitetura do projeto dessa forma foi tomada pensando que, no futuro, quando ele estiver disponível em produção, pode ser necessário alterar o banco de dados escolhido por uma solução financeiramente mais viável, e ao modularizar especificamente os serviços de armazenamento, o esforço para refatorar as demais camadas será reduzido.

# 5. Considerações Finais e Trabalhos Futuros

Ao fim do projeto, é possível realizar as operações básicas de marcação básica de documentos para fins de análise qualitativas. O usuário é capaz de identificar, codificar e se relembrar de seus códigos tanto através de sua visualização no painel lateral a uma fonte de pesquisa textual quanto em uma lista ordenada de códigos.

Como explicado ao longo do projeto, diversas decisões foram tomadas em prol da velocidade de lançamento deste projeto. Elas serviram à versão atual permitindo a criação de um produto mínimo testável e é importante reconhecer as limitações causadas por essa abordagem. Ainda é possível encontrar alguns bugs de responsividade ao ampliar e reduzir a janela em algumas das interfaces gráficas, e visões que não atualizam em tempo real no momento da adição de mais um elemento em banco de dados, como no caso da listagem do codebook. Essas alterações, para futuras versões, podem garantir uma experiência de uso muito mais fluida e ágil para o pesquisador

Um grande passo para a continuidade do projeto é receber e tratar fontes de pesquisa com outros formatos, como imagens, vídeos e áudios, pois são dados de suma importância para investigações de várias áreas do conhecimento.

Por fim, gostaria de deixar a reflexão de que fazer parte dos passos iniciais de uma plataforma *open-source* é algo recompensador, por ser uma oportunidade de agradecer à comunidade por todo o apoio oferecido ao longo destes anos de graduação e poder também retornar um pouco a ela.

# 6. Referências bibliográficas

[1] Strauss, A. and Corbin, J., 1998. *Basics of qualitative research techniques*. Thousand Oaks, CA: Sage publications

[2] Cruzes, D.S. and Dyba, T., 2011, September. Recommended steps for thematic synthesis in software engineering. In 2011 international symposium on empirical software engineering and measurement (pp. 275-284). IEEE.

[3] ATLAS.ti. Disponível em <https://atlasti.com/>. Acesso em 19 de Janeiro de 2021.

[4] MAXQDA. Disponível em <https://www.maxqda.com/>. Acesso em 19 de Janeiro de 2021.

[5] NVivo. Disponível em <https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home>. Acesso em 19 de Janeiro de 2021.

[6] QDA Miner Lite. Disponível em <https://provalisresearch.com/products/qualitative-data-analysis-software/freeware/>. Acesso em 19 de Janeiro de 2021.

[7] PipocaQDA. Disponível em <https://github.com/PipocaSw/PipocaQDA>. Acesso em: 19 de Janeiro de 2021.

[8] Google Angular. Disponível em: <https://angular.io/docs>. Acesso em: 19 de Janeiro de 2021.

[9] Microsoft TypeScript. Disponível em: <https://www.typescriptlang.org/>. Acesso em 19 de Janeiro de 2021.

[10] Google Cloud Firestore. Disponível em <https://firebase.google.com/docs/firestore>. Acesso em 19 de Janeiro de 2021.

[11] Google Firebase. Disponível em <https://firebase.google.com/>. Acesso em 19 de Janeiro de 2021.

[12] TinyMCE Documentation. Disponível em: <https://www.tiny.cloud/docs/>. Acesso em: 19 de Janeiro de 2021.

[13] Range - Web APIs MDN. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/Range>. Acesso em 19 de Janeiro de 2021.

[14] XPath Tutorial. Disponível em <https://www.w3schools.com/xml/xpath_intro.asp>. Acesso em 19 de Janeiro de 2021.

[15] NgxMatSelectSearch. Disponível em <https://www.npmjs.com/package/ngx-mat-select-search>. Acesso em 19 de Janeiro de 2021.

[16] Draft.js. Disponível em <https://draftjs.org/>. Acesso em 19 de Janeiro de 2021.

[17] SQLite. Disponível em <https://www.sqlite.org/index.html>. Acesso em 19 de Janeiro de 2021.

[18] MongoDB. Disponível em <https://www.mongodb.com/>. Acesso em 19 de Janeiro de 2021.

[19] Angular Material. Disponível em <https://material.angular.io/>. Acesso em 19 de Janeiro de 2021.

[20] Material Design. Disponível em <https://material.io/>. Acesso em 19 de Janeiro de 2021.

[21] JavaScript | MDN. Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em 19 de Janeiro de 2021.