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

PEDRO FREDERICO FALLAVENA KAMPMANN

Uma solução para a extração e consulta de dados gerenciais do projeto SOS PME

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

Orientador: Prof[a]. Dr[a]. Karin Becker

Porto Alegre
2021

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer à minha família. A meu pai, Frederico, e à minha mãe, Maria Inez, agradeço pelos bons conselhos, pelas oportunidades e pelas boas condições de estudo que puderam me propiciar.

Agradeço ao meu parceiro, Leonardo, por compartilhar comigo tanto os altos quanto os baixos dos últimos meses. Agradeço também aos meus amigos que, mesmo sem nos termos visto pessoalmente, me apoiaram bastante durante este período.

Agradeço à Daniela Brauner, coordenadora do projeto SOS PME, pela atenção e tempo dedicados a explicar a estrutura interna e as necessidades enfrentadas pelo projeto. E, por fim, agradeço à Prof[a]. Karin Becker pela oportunidade de participar deste projeto, pela motivação e paciência e pela ótima orientação.

RESUMO

O projeto SOS PME da Escola de Administração UFRGS tem como objetivo auxiliar pequenas e médias empresas com problemas relacionados à crise econômica gerada pela pandemia da COVID-19. O SOS PME utiliza um quadro Trello, uma solução Kanban, para organizar os colaboradores no atendimento às empresas. Este quadro concentra, assim, todas informações destes atendimentos, tais como os problemas enfrentados, soluções propostas, e lições aprendidas. Contudo, a plataforma Trello não permite que toda esta informação seja consultada e explorada de forma adequada. Isto acaba impedindo a gestão do projeto de se beneficiar desta rica experiência, visto que a extração de informações relativas aos atendimentos oferecidos também é um de seus objetivos. Pensando neste problema, este trabalho visa oferecer uma solução para usuários não-especialistas em Bancos de Dados que permita consultar, extrair e atualizar informações necessárias para atividades gerenciais e analíticas do SOS PME. A solução permite importar para um Sistema de Banco de Dados os dados provenientes do Trello, definir consultas de forma facilitada, e exportar os dados para serem analisados por outras ferramentas. Ao concluir este trabalho, desenvolveu-se uma aplicação organizada em uma arquitetura cliente-servidor, que permite que as funcionalidades sejam acessíveis a partir da web.

Palavras-chave: Sistema de Gerência de Banco de Dados. SOS PME. Trello. MongoDB.

ABSTRACT

The SOS PME project, developed by the Escola de Administração UFRGS (Business Faculty of UFRGS), aims to help small and medium-sized businesses facing dificulties due to the economic crisis caused by the COVID-19 pandemic. The project uses a Trello board, which itself is a Kanban board, as a way to organize information regarding the assistance provided to the businesses, such as descriptions of the problems they faced, the proposed solutions, and the lessons learned from each assistance process. However, the Trello platform does not allow all this information to be properly queried nor explored. In the end, this prevents the project's management team from fully benefiting from the experience, since extracting information regarding the assistance provided to the businesses is also one of its goals. This Bachelor Thesis focuses its efforts in providing a solution to non-specialist users that allows them to query, extract and update information necessary for SOS PME's management and analytical activities. The solution allows its users to import data retrieved from a Trello board into a Databank, specify queries in a simplified way, and to export data for analysis with other tools. Thus, we provide an application that uses the client-server model in order to allow its functionalities to be accessible from the web.

Keywords: Database Management System, SOS PME, Trello, MongoDB.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

UFRGS	Universidade Federal do Rio Grande do Sul
EA	Escola de Administração
SGBD	Sistema de Gerência de Banco de Dados
JS	Javascript
JSON	Javascript Object Notation
BSON	Binary JSON
CSV	Comma-separated values
SQL	Structured Query Language
NoSQL	Not Only SQL
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
URL	Uniform Resource Locator
API	Application Programming Interface
MVC	Model-View-Controller

SUMÁRIO

1 INTRODUÇÃO

No ano de 2020, em função da pandemia do novo coronavírus, empresas de diversos segmentos precisaram adaptar sua atuação a fim de manter seus negócios e ao mesmo tempo garantir a segurança de seus funcionários e clientes. Em muitas situações, o trabalho presencial foi completamente comprometido, tornando-se inviável por conta da alta transmissibilidade da doença. Dentro de um novo cenário econômico fortemente impactado pela COVID-19, fez-se necessária uma readaptação do modelo usual de trabalho. Empresas precisaram enfrentar novos desafios não planejados, como por exemplo o impedimento da abertura de lojas físicas, gastos adicionais com treinamentos sanitários, adaptação para o trabalho remoto, ou a implementação de medidas de segurança a fim de proporcionar um ambiente de trabalho com grau mínimo de risco à saúde. Além disso, juntamente à economia em desaceleração e aos novos gastos, muitas empresas encararam dificuldades em se manterem operantes com todos seus funcionários, ou até mesmo continuarem abertas.

Observando o contexto previamente descrito, professores e alunos da Escola de Administração da UFRGS (*EA-UFRGS*) idealizaram em Março de 2020 o SOS PME - Rede de Assessoria Empresarial[1], um projeto de extensão que tem por objetivo assistir e orientar pequenas e médias empresas. O SOS PME oferece suporte às empresas solicitantes principalmente nas áreas administrativa, contábil e de marketing. As consultorias são realizadas por equipes especializadas, que trabalham diretamente em conjunto com a empresa, de maneira a identificar e entender os problemas enfrentados pelo negócio. Então, alternativas são pensadas a fim de melhor orientar a empresa a superar as dificuldades constatadas.

Atualmente, a gestão interna dos atendimentos é feita à mão, através de ferramentas *ad-hoc*. Por exemplo, são utilizados formulários *Google Forms* para a inscrição das empresas no programa SOS PME, planilhas para o controle de participação dos voluntários, e a plataforma de gerenciamento de projetos Trello[2] para acompanhamento dos atendimentos. O Trello é uma aplicação *web* do tipo *Kanban* (ESPINHA, 2019), e no SOS PME cada atendimento a uma empresa é representado por um cartão. Assim, todo o conhecimento e experiência sobre a empresa, o atendimento e os resultados são informações internas ao respectivo cartão. Na prática, o Trello tornou-se a ferramenta mais usada pelos voluntários nos atendimentos do SOS PME.

[1] <https://www.ufrgs.br/escoladeadministracao/sospme/>
[2] <https://trello.com/>

Como o Trello é voltado originalmente à gestão de projetos, o seu paradigma dificulta a extração de informações sobre os atendimentos em geral, pois implica um laborioso trabalho de verificação individual de cada cartão. Para o SOS PME, isto dificulta a troca de experiências e lições aprendidas nos atendimentos realizados. Essa dificuldade se manifesta principalmente na exportação de dados provindos de atendimentos anteriores, por exemplo, sobre as equipes de voluntários que atuaram neles. Além disso, torna-se mais difícil a obtenção de informação sobre os resultados de cada atendimento, impedindo que as lições aprendidas de atendimentos encerrados possam ser reaproveitadas por outras equipes no futuro.

Também é de extrema importância considerar o crescimento do SOS PME, seja no número de casos atendidos ou no número de voluntários. Tal aumento dificulta a exportação de relatórios, que precisa ocorrer manualmente, e a geração de certificados de participação, visto que tais dados encontram-se muitas vezes espalhados em diferentes listas e cartões do Trello.

Uma alternativa seria exportar estes dados para um Sistema de Gerenciamento de Banco de Dados (SGBD) (ELMASRI; NAVATHE, 2005), o que facilitaria a formulação de consultas usando os recursos deste tipo de plataforma. Contudo, isto não permitiria a execução de tarefas de extração de dados por não especialistas. Sendo assim, faz-se então necessária a criação de uma plataforma que possibilite a extração de informações gerenciais e analíticas de forma rápida e eficiente relativas ao projeto SOS PME. Na medida que o projeto SOS PME vem se consolidando, e formando parcerias para atendimentos mesmo por outras Universidades, a possibilidade de extrair e compartilhar todas as informações faz-se ainda mais necessária.

Este trabalho tem como objetivo desenvolver uma infraestrutura geral que colete dados a partir das ferramentas utilizadas para a gestão do projeto SOS PME, e que permita que eles sejam mais facilmente consultados por pessoas ligadas ao projeto, através de uma interface de consulta voltada a não especialistas em Bancos de Dados. Além disso, visa também oferecer mecanismos para viabilizar sua exploração com o uso de ferramentas *third-party* (e.g. Google Data Studio[3]).

O restante do trabalho é organizado como segue: O Capítulo 2 apresenta a fundamentação teórica, detalhando os tipos de SGBDs considerados para a implementação do trabalho proposto e suas vantagens e desvantagens, a plataforma Trello e as ferramentas analíticas utilizadas para a gestão do projeto SOS PME. O Capítulo 3 discute em maior

[3]<https://datastudio.google.com>

detalhe o SOS PME, como ele é organizado internamente e suas demandas em termos de gestão de dados. Os Capítulos 4 e 5 apresentam, respectivamente, a proposta de solução e o detalhamento de sua concepção e implementação. O Capítulo 6 apresenta conclusões e as direções futuras para o trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo procura apresentar uma visão geral das principais tecnologias e conceitos relacionados ao presente trabalho.

2.1 Kanban

Baseado no modelo de produção *lean* Toyota, o *Kanban* é um sistema visual de organização de produção e gestão de tarefas (ESPINHA, 2019). É utilizado para a modelagem de projetos, normalmente para a coordenação interna de equipes, embora também possa ser usado individualmente. O sistema é composto de um quadro, subdividido em colunas, e cartões, como demonstra a Figura 2.1.

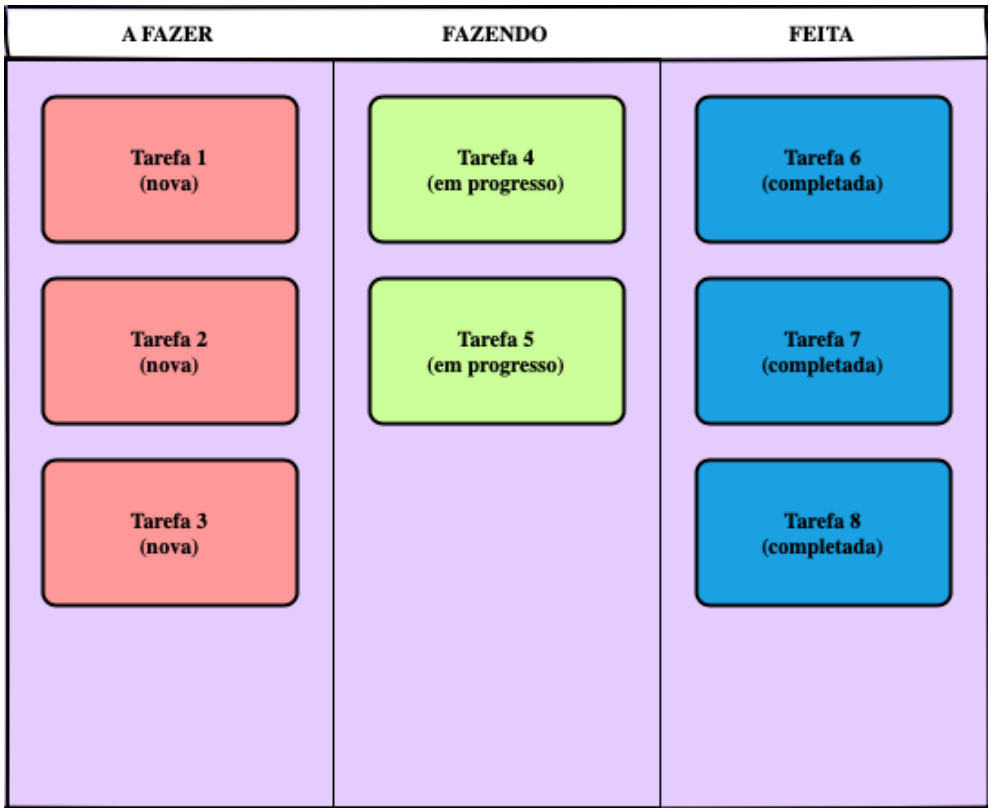


Figura 2.1 – Exemplo de quadro Kanban

O quadro como um todo é o *Kanban*, e possibilita uma visão geral do fluxo de trabalho modelado. Um quadro geralmente contém pelo menos as colunas “A fazer”, “Fazendo” (ou “Em execução”) e “Feito”.

Cada coluna modela um possível estágio ou estado de processamento de um item no projeto, variando conforme as necessidades da equipe. Já cada item de trabalho, seja

uma tarefa, ação a ser tomada ou outro item definido pela equipe, é representado por um cartão. Cada item é atribuído a uma coluna, conforme o seu estado. Quando este estado muda, o cartão é movido para a respectiva coluna. No contexto representado na Figura 2.1, se espera que os cartões da coluna “Fazendo” sejam movidos para a coluna “Feita” se ou quando as tarefas forem completadas. É comum também a utilização de um sistema de cores para os cartões, a fim de indicar diferentes aspectos do item, como por exemplo níveis de prioridade ou tipos de atividade.

A utilização do sistema *Kanban* propicia à gestão de um projeto um panorama do seu estado atual e do ritmo do fluxo de trabalho. Além disso, o poder de modelagem do sistema também permite controlar a interdependência de processos, ajudando a garantir que nenhuma atividade tenha seu progresso bloqueado pela falta de uma anterior.

2.2 Trello

A ferramenta Trello é uma aplicação *on-line* colaborativa para organização de projetos. Através de uma interface visual, ela oferece uma série de ferramentas para o gerenciamento de projetos.

Figura 2.2 – Exemplo de um quadro Trello

A influência *Kanban* é bastante visível na interface e ferramentas disponíveis no Trello. Como ilustrado pela Figura 2.2, ele conta com as mesmas entidades centrais de quadro, colunas (aqui chamadas listas) e cartões. Porém, por se tratar de uma aplicação digital, o Trello possui funcionalidades que não fazem parte de um quadro *Kanban* clássico. Ele permite associar uma equipe de usuários a cada cartão, anexar documentos

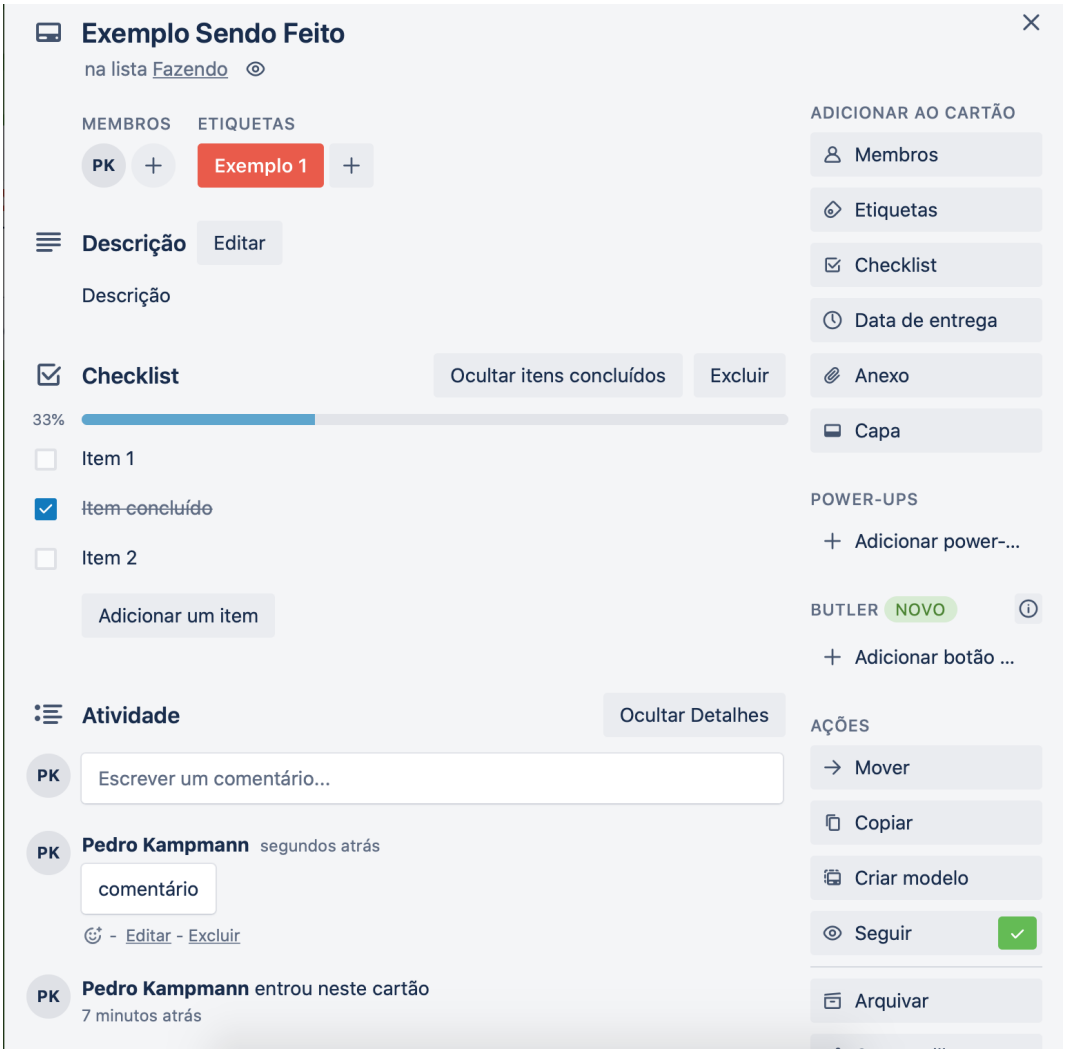


Figura 2.3 – Exemplo de um cartão Trello

e comentários a ele, definir uma *checklist* (lista de afazeres controlada) com objetivos e atribuir uma cor ou etiqueta ao cartão. A Figura 2.3 traz um exemplo de um cartão que utiliza estas funcionalidades. Assim como para um *Kanban* físico, cores e etiquetas permitem distinguir grupos de tarefas que possuem alguma relação, por exemplo por serem do mesmo lote de tarefas ou por pertencerem a uma mesma categoria. No entanto, o Trello permite alterar estas marcações a qualquer momento ao longo do projeto.

Dada a sua simplicidade, a plataforma acaba incentivando a criatividade e liberdade de modelagem de processos por parte dos usuários. O Trello também possibilita a integração de diversas ferramentas populares com a sua plataforma, como *Slack*[1] e *Google Drive*[2], além de disponibilizar uma ferramenta própria de automatização de ações chamada *Butler*[3]. Esta ferramente permite com que longas sequências de tarefas repeti-

[1] <https://slack.com/intl/pt-br/>
[2] <https://www.google.com/drive/>
[3] Tutorial do Butler do Trello: <https://blog.trello.com/br/trello-butler-tutorial>

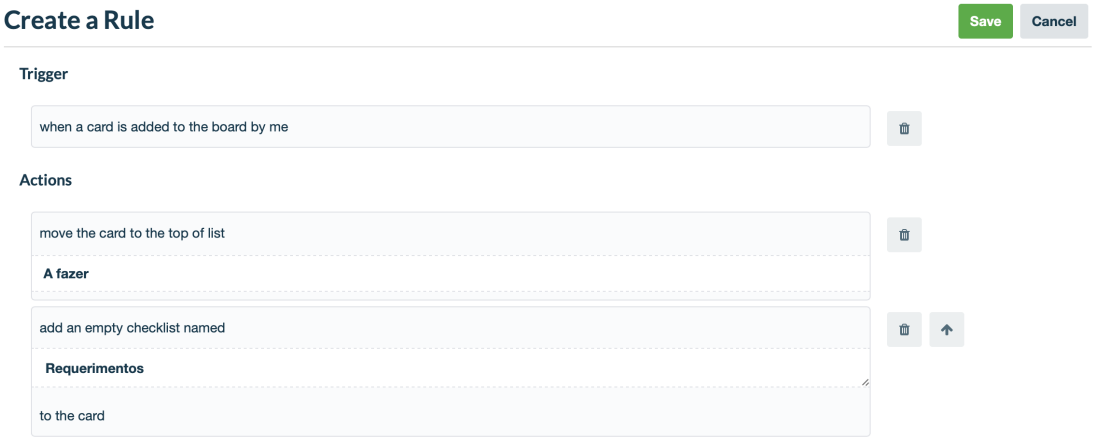


Figura 2.4 – Exemplo de uma regra de automatição da plataforma Trello

tivas sejam feitas através de um botão ou sempre que um conjunto de regras é ativado. A Figura 2.4 exemplifica uma regra que define que todo novo cartão criado no quadro em questão deve ser movido para o topo da lista "A fazer" e ter uma *checklist* chamada "Requerimentos" anexada a ele.

Por conta da sua flexibilidade, o Trello é utilizado por grupos de usuários com propósitos muito diversos. Sua popularidade continua crescendo principalmente entre equipes de desenvolvimento ágil, apesar da plataforma não suportar nativamente todas as funcionalidades necessárias para, por exemplo, se implementar *Scrum*. Ainda assim, existem diversas ferramentas disponíveis no Trello que replicam estas funcionalidades, e um dos mais usados, *Scrum by Vince*[4], é utilizado atualmente em mais de dez mil quadros Trello. Ele permite atribuir pontos a cada tarefa, definir uma data para o final de cada *sprint* e até gerar artefatos complementares, como o gráfico de *Burndown*.

Porém, o Trello também é muito utilizado por outros tipos de equipes, como é o caso do veículo de notícias *on-line* britânico *Wired*[5], que utiliza o Trello para modelar as etapas de publicação de um artigo, desde a ideia inicial até o seu lançamento, passando pelas etapas de pesquisa e edição. Etiquetas em cada cartão indicam a seção de notícias a que o texto pertence, e um power-up de calendário permite à equipe visualizar o cronograma de publicações do mês inteiro (TRELLOBLOG, 2020).

A plataforma permite exportar manualmente um quadro completo a partir da interface do *site*, em formato *JSON*[6]. Um exemplo de um quadro *Trello* exportado neste formato pode ser visto na Figura 2.5. A plataforma também possibilita a extração de in-

[4]Scrum by Vince: <https://trello.com/power-ups/5965595bd3e1d4824063fe3a/scrum-by-vince>
[5]Wired: <https://www.wired.co.uk>
[6]JSON.org: <https://www.json.org/json-pt.html>

formações para ambientes externos tanto através de uma *API*, com a qual se pode acessar os diferentes elementos do quadro individualmente, quanto através dos chamados *Webhooks*, que são pontos de acesso à *API* que notificam as aplicações de mudanças no quadro.

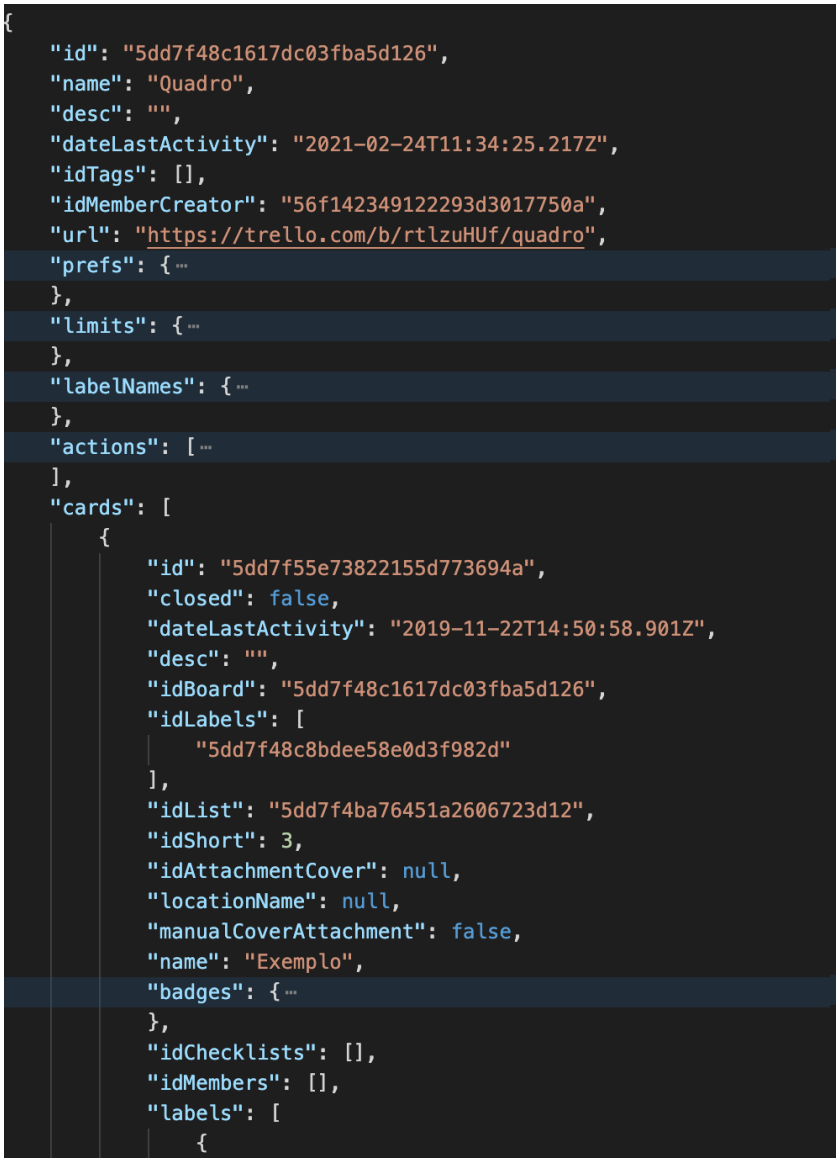


Figura 2.5 – Excerto de uma representação de um quadro Trello em formato JSON

No entanto, o Trello acaba tendo certas limitações como consequência de ser um sistema de modelagem de projetos em estágios. Apesar de possuir recursos de busca integrados à plataforma, estes não são muito avançados. Uma busca por todos os cartões de uma certa etiqueta, como por exemplo ilustrada na Figura 2.6, retorna na interface somente os nomes e etiquetas de cada cartão, ainda divididos em listas. Isto acaba fazendo com que muitas vezes seja necessária uma verificação manual de cada um deles, caso as informações de interesse não estejam visíveis neste primeiro momento.

Neste trabalho, o quadro da plataforma Trello utilizado pelo projeto SOS PME

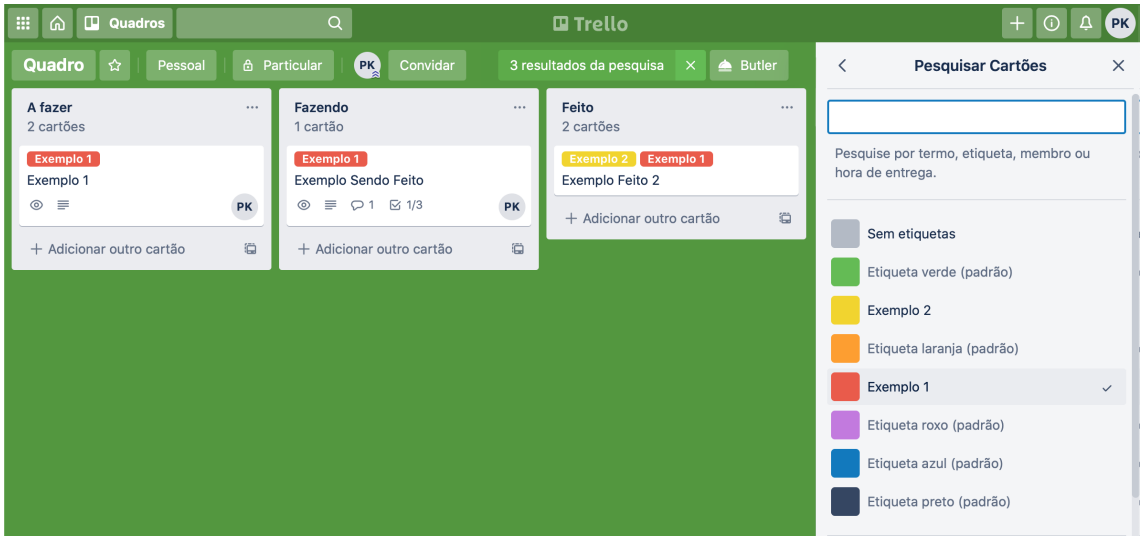


Figura 2.6 – Exemplo do resultado de uma pesquisa por etiqueta na plataforma Trello

para sua gestão interna é a principal fonte de dados da aplicação aqui proposta. O projeto utiliza seu quadro como ponto centralizador de dados sobre os atendimentos, tornando-o o maior foco de extração de informações.

2.3 Sistemas de Gerência de Banco de Dados

Um Sistema de Gerência de Banco de Dados (SGBD), é uma coleção de programas que permite aos usuários e outras aplicações manter um Banco de Dados, facilitando sua definição, construção e manipulação (ELMASRI; NAVATHE, 2005). Os SGDBs podem ser classificados de acordo com o modelo de dados no qual eles se baseiam, ou seja, de qual forma eles armazenam dados. Destes, dois modelos se destacam pela popularidade: o Relacional e o Orientado a Documentos.

2.3.1 O Modelo Relacional

O modelo de dados relacional se baseia na Álgebra Relacional, provinda da teoria de conjuntos e da lógica de predicados de primeira ordem. Ela define uma Relação como um conjunto de n-tuplas, onde cada uma possui n valores de dados relacionados, chamados atributos (ELMASRI; NAVATHE, 2005). Os SGBDs relacionais representam essas relações em forma de tabelas, onde cada coluna representa um atributo e cada linha, uma tupla. A fim de evitar ambiguidade na identificação de tuplas, é determinado que pelo menos um atributo deva possuir somente valores únicos e não-nulos para cada uma delas.

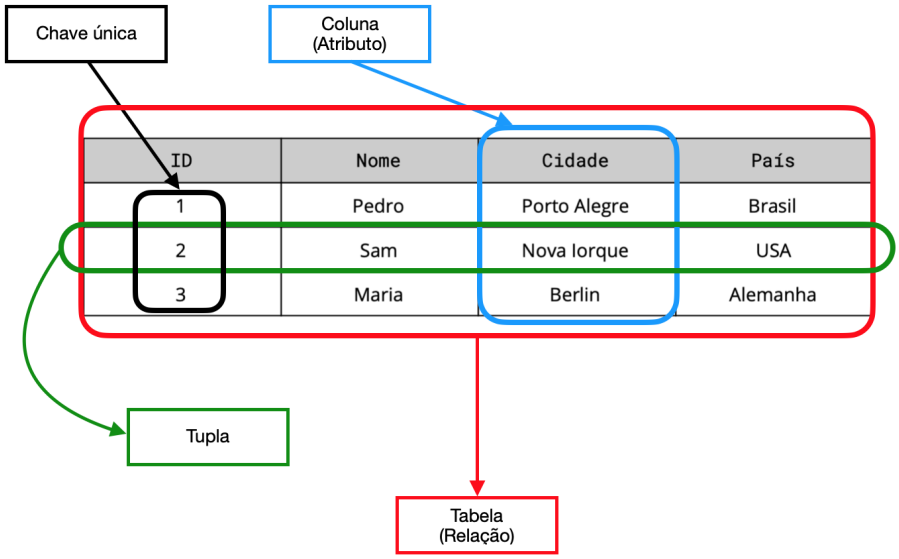


Figura 2.7 – Exemplo de uma tabela relacional

Este atributo é chamado de chave primária da relação. Um exemplo de tabela relacional pode ser visto na Figura 2.7.

As tabelas podem ser interligadas através de chaves estrangeiras, que são atributos especiais que referenciam valores de atributos de outras tabelas, tipicamente suas chaves primárias.

A Álgebra Relacional, com suas operações baseadas em conjuntos, serviu como base teórica para a definição da linguagem de consulta *Structured Query Language* (SQL)[7]. A SQL rapidamente se tornou a forma canônica de interação com Bancos de Dados, sendo adotada por soluções comerciais desde a década de 1980, e contribuindo para o domínio do modelo Relacional como padrão de indústria desde então.

2.3.2 NoSQL

Os SGBDs relacionais oferecem, através da SQL, muita flexibilidade na definição e construção de consultas. Principalmente na época em que surgiram, esta flexibilidade era mais importante do que considerar a existência de esquemas flexíveis (REDMOND; WILSON, 2012). Porém, hoje em dia existe uma quantidade muito maior de dados semi-estruturados para análise, cujo crescimento se deu principalmente por causa da *internet*.

Este fato, junto a outras demandas observadas em casos de uso do mundo real, motivou a criação das alternativas atualmente conhecidas como *NoSQL* (REDMOND;

[7]What is SQL? The lingua franca of data analysis: <https://www.infoworld.com/article/3219795/what-is-sql-the-lingua-franca-of-data-analysis.html>

WILSON, 2012) (*Not only SQL*), do inglês, “Não Somente SQL”. O termo se refere ao conjunto de modelos de dados que não utilizam tabelas para sua representação, como por exemplo os modelos Orientado a Colunas, de Grafos e Orientado a Documentos. De forma resumida, Bancos de Dados NoSQL armazenam dados sem impor que eles possuam uma estruturação interna prévia, nem que os mesmos atributos existam para todas as tuplas de uma coleção.

A criação dos SGBDs NoSQL foi motivada, como mencionado anteriormente, pelo aumento dos casos de uso focados em pontos fracos dos SGBDs relacionais, como por exemplo a demanda por escalonamento horizontal, alta disponibilidade e adaptação rápida a mudanças no esquema dos dados. Ainda assim, a grande maioria destes sistemas implementa algum tipo de linguagem de consulta baseada em SQL.

2.3.3 O Modelo Orientado a Documentos

Os SGBDs NoSQL orientados a documentos armazenam dados como documentos, que são conjuntos de pares chave-valor. Tanto as chaves (campos únicos de identificação) quanto os valores podem ser estruturas de quaisquer tipos, inclusive outros documentos. Esta flexibilidade na representação de dados e a possibilidade de aninhamento dos dados contribuiu para o crescimento da popularidade deste modelo nos últimos anos (REDMOND; WILSON, 2012). Um exemplo deste tipo de documento pode ser visto na Figura 2.5, que mostra dados organizados como chaves e valores, inclusive contendo aninhamento.

Também conhecidos como SGBDs *schemaless* (sem esquema), estes sistemas são mais comuns em aplicações *on-line* ou baseadas em dados semi-estruturados, já que não é imposta a existência de uma estruturação prévia dos dados.

Neste trabalho visamos dar apoio à extração e gestão de informações do projeto SOS PME. Essas informações estão contidas principalmente em cartões da plataforma Trello, como dados semi-estruturados. Assim, o modelo Orientado a Documentos, por conta de sua flexibilidade, foi o selecionado. Foram analisados dois SGBDs Orientados a Documentos para se identificar a opção mais adequada.

O primeiro, chamado CouchDB[8], é um sistema criado para dar apoio a aplicações *on-line*. Projetado para lidar com a baixa confiabilidade da rede, tem como foco principal fornecer alta disponibilidade. É muito escalável, podendo ser implementado tanto em

[8]<https://couchdb.apache.org>

```
$ curl http://localhost:5984/music/74c7a8d2a8548c8b97da748f43000ac4
{
  "_id":"74c7a8d2a8548c8b97da748f43000ac4",
  "_rev":"4-93a101178ba65f61ed39e60d70c9fd97",
  "name":"The Beatles",
  "albums": [
    {
      "title":"Help!",
      "year":1965
    },{
      "title":"Sgt. Pepper's Lonely Hearts Club Band",
      "year":1967
    },{
      "title":"Abbey Road",
      "year":1969
    }
  ]
}
```

Figura 2.8 – Exemplo de consulta CouchDB utilizando o método HTTP GET
Fonte: Adaptado de (REDMOND; WILSON, 2012)

aplicações móveis quanto em grandes *Datacenters*. O CouchDB se baseia em REST[9], uma arquitetura de representação de dados para a *web*, o que significa que toda comunicação entre a aplicação e o SGBD se dá através de requisições HTTP. Internamente armazenados no formato semi-estruturado *JSON*, os dados podem ser consultados através do método HTTP GET. A Figura 2.8 apresenta uma consulta a um Banco de Dados CouchDB, chamado "*music*", que utiliza este método para recuperar um documento a partir de seu valor "_id".

No entanto, consultas mais complexas, que envolvam a criação de uma Visão[10], só podem ser executadas através de tarefas *MapReduce* escritas na linguagem de programação Javascript, e comunicadas ao SGBD com o método HTTP POST. O modelo de programação *MapReduce* (em inglês, a junção das palavras "Mapear" e "Reduzir") foi criado para facilitar o processamento de um grande volume de dados de forma paralela e distribuída (PAIVA, 2011). Ele consiste dessas duas operações, *map* e *reduce*. A operação *map* é geralmente associada a funções simples executadas em paralelo sobre cada elemento de uma sequência, neste caso, sobre cada chave da entrada. O resultado emitido pela função *map* é um conjunto intermediário de dados, com a operação *reduce* responsável pela combinação ou sumarização destes.

A Figura 2.9 exemplifica uma solução Map/Reduce para a contagem de palavras. Neste caso, a função *map* emite um par chave-valor com a palavra em questão e o valor 1, e a função *reduce* soma todos os valores emitidos para cada chave, o que resulta no número de ocorrências daquela palavra.

[9] Arquitetura REST: Saiba o que é e seus diferenciais <https://www.totvs.com/blog/developers/rest/>
[10] CouchDB: Introduction to Views <https://docs.couchdb.org/en/stable/ddocs/views/intro.html>

```
function map(palavra) { //mapeia um item
    emite(palavra, 1); //emite um par chave-valor

}a

function reduce(chave, valores) { //combina todos os valores que
                                  //possuem a mesma chave
    retorna soma(valores);

}
```

Figura 2.9 – Pseudo-código de funções Map/Reduce

O segundo SGBD orientado a documentos analisado, o MongoDB[11], é o SGBD NoSQL mais popular atualmente no *ranking* DB-Engines[12]. Foi construído com uma linguagem de consulta baseada em SQL, para incentivar a sua facilidade de uso. Os documentos são internamente armazenados em formato *BSON*, que é uma forma binária do formato *JSON*, e podem ser consultados utilizando-se a sua própria linguagem de consulta, ou através de um dos *drivers* disponíveis para várias linguagens de programação.

Foi projetado com performance e facilidade de acesso aos dados como objetivos principais, sendo muito adotado para aplicações de *Big Data* por conta de sua capacidade de lidar com dados em grande escala.

O SGBD MongoDB foi o escolhido para a aplicação proposta neste trabalho, pois os dados presentes no quadro Trello utilizado pelo projeto SOS PME são do tipo semi-estruturado. Além disso, dentre os SGBDs orientados a documentos analisados, o MongoDB oferece um nível mais baixo de complexidade inicial para um projeto em contraste com o CouchDB. Apesar de focado na utilização com grandes volumes de dados, ele conta com uma documentação muito completa e interfaces simplificadas, o que também promove seu uso em projetos de menor tamanho.

2.4 MongoDB

Esta seção visa apresentar o SGBD escolhido para este trabalho em maior detalhe. Como anteriormente mencionado, o MongoDB é um SGBD *NoSQL* orientado a documentos. Foi criado com o objetivo de oferecer versatilidade de uso, escalabilidade e boa performance, aliando os benefícios de um modelo NoSQL com os benefícios e funciona-

[11] <https://www.mongodb.com>
[12] DB-Engines Ranking <https://db-engines.com/en/ranking>

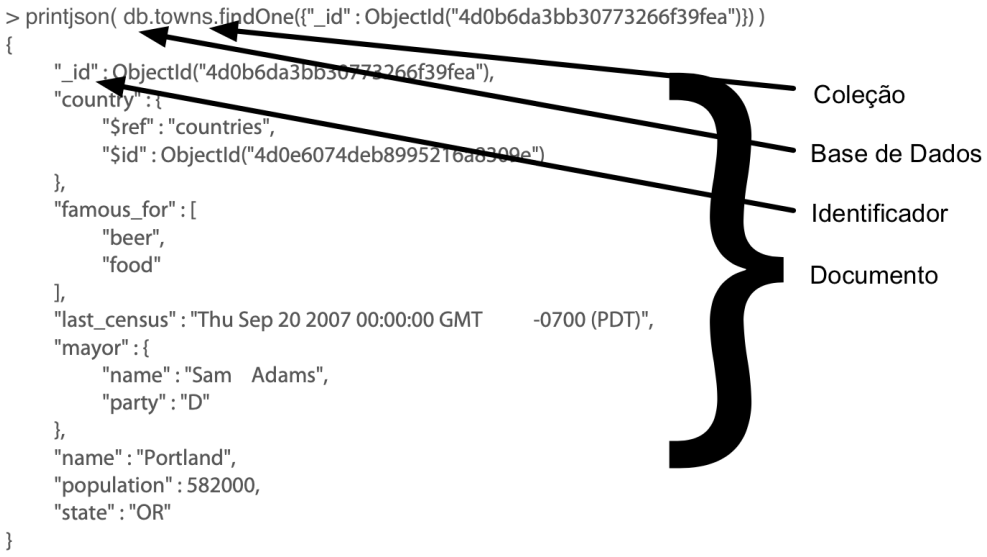


Figura 2.10 – Exemplo de um documento MongoDB
Fonte: Adaptado de (REDMOND; WILSON, 2012)

lidades do modelo relacional, que é o padrão de indústria.

Os documentos armazenados pelo MongoDB são agrupados em coleções, que são análogas às tabelas de um SGBD relacional. Cada documento, por sua vez, é como uma tupla de uma tabela relacional, mas que pode conter campos de valores que nenhum outro documento da coleção contém, ou até mesmo objetos aninhados cuja profundidade e estrutura interna não são conhecidas. Como também acontece para tuplas em tabelas relacionais, documentos do MongoDB recebem uma chave identificadora única.

Estes aspectos são ilustrados pela Figura 2.10, que mostra um documento MongoDB típico. Este exemplo é de um documento pertencente a uma coleção de cidades, com atributos de diferentes tipos. O atributo "country" (país) contém um objeto que funciona como uma chave estrangeira, referenciando um documento de outra coleção. O atributo "famous_for" (famoso por) contém uma lista de valores, e "mayor" (prefeito) contém um objeto aninhado que, por sua vez, contém outros atributos. O atributo "name" (nome), por sua vez, possui um valor simples, "Portland".

Estes documentos são armazenados internamente em formato *BSON*, que é uma forma binária do formato *JSON*. Como o formato é independente de plataforma, sua estrutura interna de pares chave-valor possui equivalentes em muitas linguagens de programação. Isto acaba facilitando a utilização de SGBDs que utilizem o formato *JSON*.

A Figura 2.11 traz um exemplo de coleção MongoDB. Para fins de comparação, a estrutura interna dos documentos é equivalente à apresentada na Figura 2.7. Aqui, no entanto, o atributo "endereço" foi modificado para melhor ilustrar a capacidade do SGBD

Figura 2.11 – Exemplo da estrutura interna de uma coleção MongoDB

de armazenar objetos aninhados, e o atributo “telefones” foi adicionado para demonstrar a possível variação no número de valores para o mesmo campo dos documentos.

Através de uma linguagem de consulta própria baseada em SQL, o MongoDB permite consultar a Base de Dados de maneira *ad hoc*, o que oferece ao MongoDB um dos considerados pontos positivos dos SGBDs relacionais. Além disso, também é possível definir critérios de busca baseados em valores aninhados dos documentos. Isto é mostrado na Figura 2.12, onde a notação “endereco.pais” denota a chave “país” pertencente ao objeto aninhado que, por sua vez, é o valor sob a chave “endereço”. O valor “Brasil” é o predicado da consulta.

Diferentemente de SGBDs relacionais, o MongoDB não oferece operações de junção *server-side*[13]. Porém, como ele possui o recurso do aninhamento de objetos, o MongoDB permite que dados provindos de certos documentos possam ser facilmente anexados a outros. Assim, a necessidade de realizar operações computacionalmente caras de junção é bastante reduzida, já que uma única operação de busca no Banco de Dados retorna o documento inteiro, inclusive dados aninhados.

Além de disponibilizar sua linguagem própria para consulta e *scripting*, também

[13]literalmente “lado do servidor” em inglês, expressão utilizada para código que roda no servidor ao invés de na aplicação cliente

```
>_MongoSH Beta
> db.collection.find({ "endereco.pais" : "Brasil" })
< { _id: ObjectId("602fd68b352c0e9a045effd2"),
    nome: 'Pedro',
    endereco: { cidade: 'Porto Alegre', pais: 'Brasil' },
    telefones: [ '(51)999999999', '+555132165487' ] }
>
```

Figura 2.12 – Exemplo de consulta a um documento aninhado em MongoDB

existem diversos *drivers* disponíveis para linguagens de programação populares, como Java, Python, PHP e Javascript. Assim, é possível se conectar e acessar as funcionalidades do banco de dados diretamente de uma aplicação final. Isto, em conjunto com a simplicidade da representação de dados nativa, permite uma alta integração do MongoDB.

Neste trabalho, utilizamos o MongoDB para manter os dados provindos do quadro Trello utilizado pelo projeto SOS PME. A escolha se baseou primariamente na sua afinidade com o formato *JSON*, também utilizado pela plataforma. Embora o modelo de dados utilizado pelo Trello seja bem definido, o que poderia contradizer a escolha por um SGBD orientado a documentos, o MongoDB apresentou-se como o melhor para a aplicação em questão, já que facilitou a importação dos dados e sua combinação com dados de outras fontes. Assim, as tarefas de validação e normalização dos dados passaram a ser responsabilidades da aplicação, e não um pré-requisito para armazená-los no SGBD. Desta forma, os dados só precisam ser normalizados quando exportados pelo usuário final do sistema.

2.5 Ferramentas Analíticas

Uma ferramenta analítica é uma aplicação que oferece um conjunto de funcionalidades voltadas à análise de dados, muito útil para a tomada de decisões. Com o crescimento em popularidade dos sistemas ditos *Big Data*, cresceu também a necessidade de se utilizar técnicas voltadas para a análise e extração de informações destes dados, já que o seu enorme volume impossibilita análises feitas por especialistas. Existem muitas ferramentas analíticas disponíveis atualmente, variando muito em termos de complexidade de uso e das funcionalidades implementadas.

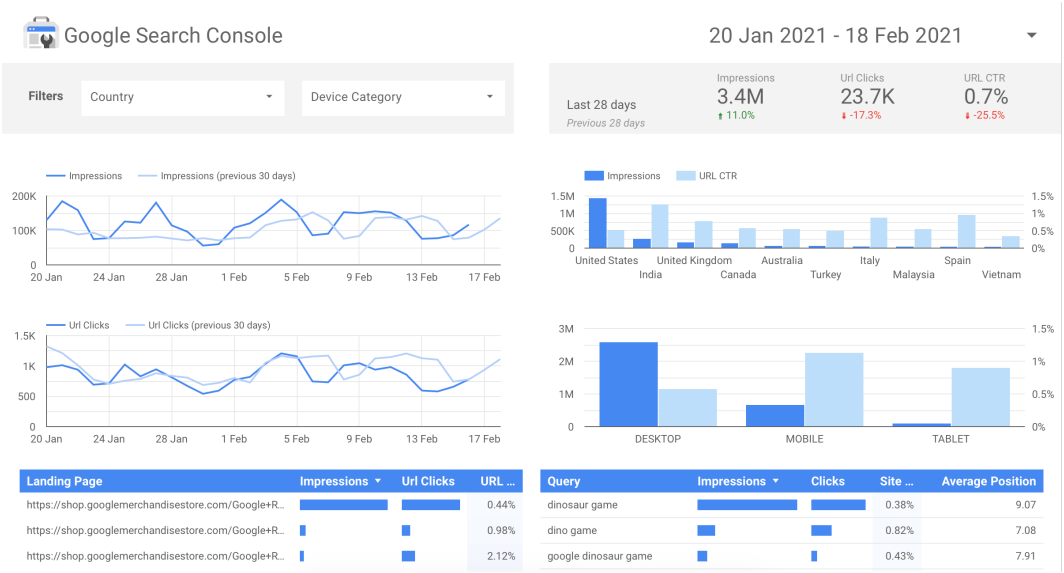


Figura 2.13 – Exemplo de Dashboard do Google Data Studio

Embora os serviços mais populares sejam aqueles que oferecem soluções para empresas e grandes projetos, existe também uma classe de ferramentas analíticas mais simples, voltadas à análise *web*. Em geral, costumam ser gratuitas e focar em aspectos como a contagem de visitas ao *website* e/ou interações com *links*, entre outros.

Bastante comum nestas ferramentas é o conceito de um *dashboard*, como ilustrado pela Figura 2.13. Um *dashboard* é um painel virtual composto de diferentes gráficos, métricas analíticas e outros artefatos visuais, cujo objetivo é centralizar toda a informação obtida dos dados.

O Google Data Studio[14] é uma plataforma *on-line* gratuita, com foco na visualização de dados provenientes de diferentes fontes de dados. Ele oferece uma interface simplificada, que não exige conhecimentos de programação por parte de seus usuários. Como é parte da família de serviços Google, a plataforma oferece facilidades na integração de fontes de dados de seus outros serviços, inclusive tabelas e arquivos *CSV* armazenados no *Google Drive* ou *Google Sheets*. Também permite conexões com certos Bancos de Dados relacionais que utilizam SQL. As opções de importação de dados disponíveis para um *dashboard* do Google Data Studio consideram somente dados tabulares, podendo ser vistas na figura 2.14.

A criação de um *dashboard* na aplicação acontece de forma gráfica, com a conexão a uma fonte de dados e a configuração de cada elemento que o compõe. Um elemento permite filtrar campos, *drill-down* em dimensões e aplicar certas funções sobre os dados nele contidos. Elementos podem ser tabelas, gráficos de barra, de linhas, de setores, entre

[14] <https://datastudio.google.com>

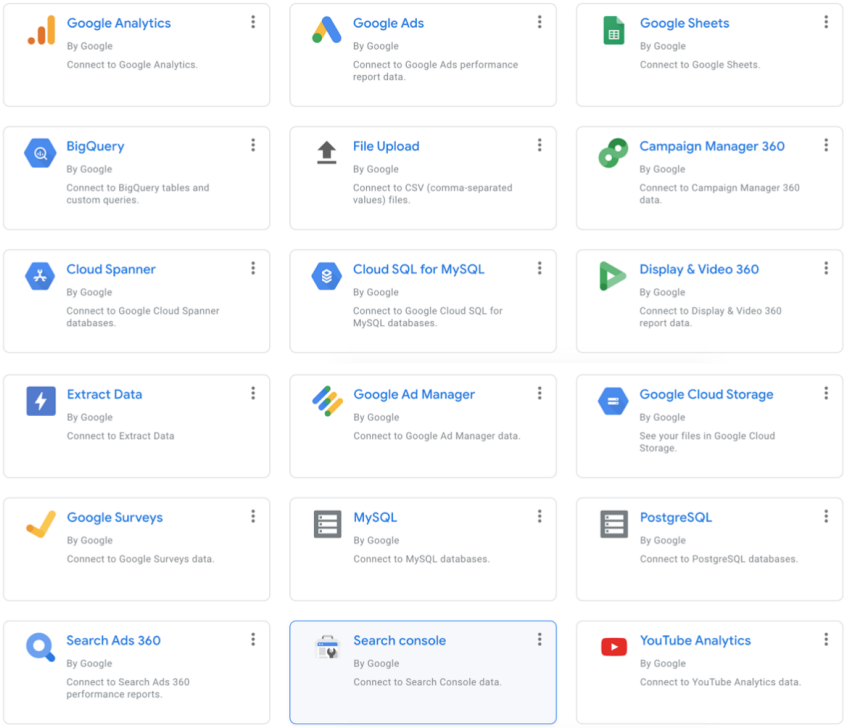


Figura 2.14 – Opções de importação de dados oferecidas pelo Google Data Studio

outros.

O projeto SOS PME utiliza-se de um *dashboard* do serviço Google Data Studio para análise de dados relativos à sua gestão, ainda que de forma restrita. Por este motivo, este trabalho procura viabilizar um melhor preparo destes dados a fim de possibilitar um maior aproveitamento do valor que pode ser extraído dos dados relativos aos atendimentos do projeto SOS PME, por exemplo utilizando-se as funcionalidades desta ferramenta analítica.

2.6 Modelo Cliente-Servidor

O modelo Cliente-Servidor (BUSCHMANN et al., 1996) é uma arquitetura de sistema distribuído. Ele consiste de uma ou mais aplicações requerentes, os clientes, e um servidor centralizado, que responde a essas requisições. Embora o cliente e o servidor possam ser executados no mesmo sistema, é comum que eles façam partes de sistemas separados, que se comunicam por uma rede. Uma aplicação deste tipo oferece normalmente múltiplas instâncias da camada de apresentação, que se comunicam com uma única instância da camada de armazenamento de dados.

Um cliente é um módulo responsável pela interação com o usuário. Também

chamado de *Front-End*, ele permite aos usuários requisitar dados ao servidor, através da interface. O servidor, ou *Back-End*, é responsável por realizar as tarefas requisitadas e comunicar seus resultados ao cliente. A comunicação entre módulos se dá com a troca de mensagens, geralmente através de uma *API* (interface de programação de aplicações, sigla do inglês *Application Programming Interface*), que é uma camada de abstração para acessar serviços.

Este modelo foi utilizado na implementação do *software* proposto neste trabalho de forma a favorecer a Separação de Conceitos da aplicação (BUSCHMANN et al., 1996), já que um módulo não tem conhecimento da implementação interna do outro. Desta maneira, o cliente é uma aplicação *web*, a partir da qual consultas podem ser enviadas ao servidor. O servidor, por sua vez, realiza as tarefas relativas aos pedidos que recebe, se comunica com sua instância de Banco de Dados MongoDB e responde ao cliente com os resultados. O cliente, então, permite que esses resultados sejam visualizados através da interface e extraídos.

3 DESCRIÇÃO DO PROBLEMA

Este capítulo se dedica a apresentar o projeto SOS-PME e discutir a necessidade da obtenção de informações para a sua gestão.

3.1 O Projeto SOS PME

O projeto *SOS PME - Rede de Assessoria Empresarial* é uma iniciativa de professores da UFRGS, em sua grande maioria pertencentes à Escola de Administração, que tem como objetivo auxiliar pequenas e médias empresas. Concebido em Março de 2020, o projeto se propõe a assessorar os negócios com problemas relacionados à crise gerada pela pandemia da COVID-19. Com o seu sucesso, a iniciativa cresceu, se tornando um projeto de extensão e ganhando parceiros, como a Escola de Administração PUCRS. Ele conta exclusivamente com a participação de voluntários, em grande parte professores e alunos de diferentes áreas.

As atividades de assessoria ocorrem de maneira colaborativa entre as empresas parceiras e os programas de ensino participantes. Ao se inscrever no projeto, a empresa pode descrever sua situação e suas demandas de apoio em gestão. Assim, utilizando-se de uma metologia de ensino à distância, chamada *Service Learning*, a coordenação do projeto pode atribuir a cada caso um professor e, portanto, uma disciplina.

Desta maneira, os alunos e os mentores voluntários estão diretamente em contato com os empresários ao longo da duração das disciplinas, tendo a oportunidade de aplicarem os seus conhecimentos adquiridos e de desenvolverem juntos um plano de solução às questões levantadas.

Durante o ano de 2020, o SOS PME atendeu mais de 120 empresas, e contou com a participação de aproximadamente 300 voluntários.

3.1.1 Atendimentos

Ao se registrar para participar do projeto, o empresário responde a um questionário, onde a situação do seu negócio deve ser brevemente explicada. Nesta etapa, também são levantados dados como a idade da empresa, seu ramo de atuação e o número de funcionários, entre outros. Então, de acordo com as suas demandas, a empresa pode vir a

receber apoio nas áreas administrativa, contábil, jurídica e/ou tributária.

Os atendimentos ocorrem em *Sprints*, ou ciclos. Para cada um destes ciclos, um certo número de empresas parceiras são selecionadas para atendimento, dependendo da disponibilidade de voluntários, suas áreas de atuação e das demandas apontadas pelas empresas.

Cada atendimento é conduzido por uma equipe destes voluntários, designadas pela coordenação do projeto. Os voluntários produzem então, durante o atendimento, diversos artefatos e documentos para a empresa relativos às suas demandas. Estas ferramentas de apoio são eventualmente apresentadas à empresa em reuniões.

3.1.2 Organização Interna

O projeto SOS PME conta com voluntários que exercem funções em três níveis diferentes: o de aluno, professor ou coordenador. Além de atribuir os casos às disciplinas com os conteúdos que melhor condizem com as suas demandas, os membros coordenadores do projeto também são responsáveis por controlar a participação dos outros membros e atualizar o andamento de cada caso. Esta gestão interna do projeto se dá majoritariamente através de planilhas e de um quadro na plataforma *Trello*.

Figura 3.1 – Adaptação do quadro Trello utilizado pelo projeto (Atendimentos)

Como mencionado previamente, um quadro da plataforma *Trello* é formado de listas, e estas, por sua vez, contêm cartões. Para a gestão interna do projeto SOS PME, foi

criado um quadro com várias listas que representam diferentes aspectos organizacionais. Mais importante para o andamento do projeto são as listas utilizadas para controlar as etapas de atendimento, como “Empresas para atender”, “Em atendimento” e “Atendimentos concluídos”. A Figura 3.1 ilustra como estas listas são utilizadas no projeto. Existem também outras listas focadas na gestão interna do projeto, para as quais um cartão não representa um atendimento.

Cada atendimento é representado por um cartão, e um membro de cada equipe fica responsável por interagir com a coordenação do projeto e atualizar o cartão relativo ao atendimento do qual participa. Ele tem como tarefa anexar quaisquer documentos gerados pela equipe ao cartão, bem como garantir que ele reflita as demandas e a etapa atual do atendimento.

Os cartões que modelam atendimentos, como o da Figura 3.2, têm como atributos o nome da empresa, um conjunto de usuários associados a ele, que podem ser membros da equipe ou coordenadores, e etiquetas, indicando as principais demandas daquela empresa. Uma descrição dos problemas enfrentados por ela, retirada do questionário de inscrição, assim como comentários descrevendo as atividades da equipe também fazem parte da maioria dos cartões. Quando o atendimento é dado por concluído, espera-se que também haja pelo menos um documento anexado ao cartão, referente ao relatório final.

(a) Parte superior do cartão (b) Parte inferior do cartão

Figura 3.2 – Exemplo de um cartão referente a um atendimento completo

Sendo assim, um cartão do quadro *Trello* do projeto SOS PME pode conter informações e representar coisas diferentes de acordo com a lista na qual ele se encontra, como é o caso para cartões que representam atendimentos, como na Figura 3.2, e cartões que contém dicas ou representam reuniões. Cartões deste segundo tipo não seguem nenhum padrão de estrutura interna.

3.2 Problemas Enfrentados

A gestão do projeto SOS PME também tem como objetivo coletar informações sobre os atendimentos para fins de análise e aprendizado. Considerando sua proximidade com as disciplinas ministradas por alguns de seus membros, o projeto busca tanto beneficiá-las, provendo estudos de casos e possibilitando uma troca de experiências e lições aprendidas, quanto também se beneficiar com o conhecimento agregado pelas turmas. No entanto, isto só é possível caso as informações extraídas de casos anteriores reflitam as suas similaridades e as decisões feitas pelas equipes de assessoria.

Embora o uso da plataforma *Trello* como ponto centralizador de dados sobre os atendimentos seja suficiente para o projeto em nível operacional, ele acaba por dificultar a obtenção de informações em nível de gestão. Como um dos objetivos do projeto é o de reunir informações e estabelecer lições aprendidas a partir dos atendimentos, a utilização de um quadro *Trello* implica muitas vezes na necessidade de se verificar cartões individualmente, por exemplo de uma certa categoria ou com uma certa característica. Isto acaba impedindo que sejam detectados padrões e tendências em como os casos são tratados.

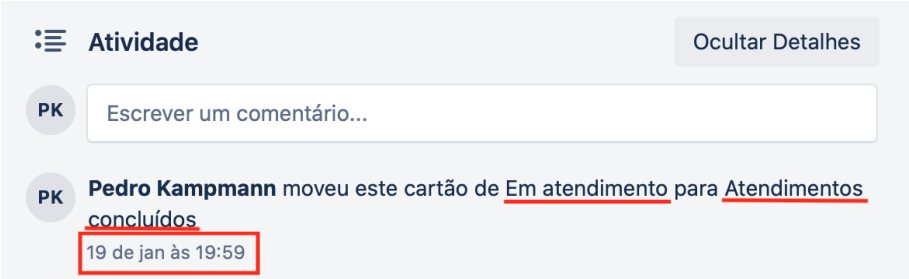


Figura 3.3 – Detalhe de um cartão demonstrando a data de conclusão de um Atendimento

Uma dessas dificuldades é ilustrada pela falta de informação sobre as datas de criação de um cartão e de finalização do atendimento por ele representado. Esta finalização ocorre no quadro *Trello* com a movimentação do cartão relativo ao atendimento para a lista de atendimentos concluídos e com a anexação de um documento, o relatório final. Embora cada cartão possua, no seu histórico de atividades, a sua data de criação e a data

em que foi movido de uma lista para a outra, seria necessário verificá-los um a um, já que esta informação não é imediatamente visível. A Figura 3.3 mostra em detalhe um exemplo desta seção de “Atividades” de um cartão que referencia um atendimento concluído.

Sendo assim, muito da extração de informações gerenciais a partir dos atendimentos acaba sendo feito à mão, como por exemplo a quantidade de casos com certas demandas e quais membros compunham as equipes. Porém, conforme o número de atendimentos completados cresce, também fica mais difícil conseguir essas informações. A fim de melhor compreender a situação enfrentada pela gestão do projeto SOS PME, foram levantadas situações, em conjunto com uma das coordenadoras do projeto, que representam essas dificuldades em se obter informação relevante.

São exemplos de atividades complexas de consulta sobre os dados do projeto:

- Gerar relatórios sobre a participação de voluntários
- Obter a relação de atendimentos encerrados
- Obter a relação entre cada voluntário e os atendimentos encerrados dos quais ele participou
- Gerar relatórios sobre atendimentos
- Obter as datas de início e fim de cada atendimento
- Obter a etapa atual dos atendimentos
- Obter a relação de demandas de cada atendimento
- Obter a relação dos participantes de cada atendimento

3.3 Considerações Finais

Neste capítulo, foi apresentada a organização interna do projeto SOS PME e os problemas enfrentados por sua gestão. Destes, buscou-se resolver aqueles relacionados à obtenção de informação presente no quadro da plataforma *Trello*, como descritos na seção anterior. Dado que diversas atividades gerenciais possuem um caráter periódico, como a geração de relatórios e certificados, identificou-se a necessidade de apoiar a criação e execução de consultas sobre os dados do projeto de maneira eficiente e prática.

4 PROPOSTA DE SOLUÇÃO

Este capítulo visa apresentar a solução proposta por este trabalho, bem como suas funcionalidades e outros conceitos envolvidos em sua concepção.

4.1 Visão Geral

Neste trabalho, propõe-se uma aplicação que oferece a seus usuários um conjunto de funcionalidades relacionadas aos dados do projeto SOS PME presentes no seu quadro *Trello*. Mais especificamente, se procura facilitar a criação e execução de buscas sobre estes dados, além de possibilitar a utilização dos resultados destas buscas por ferramentas analíticas.

A tarefa de reunir e extrair informações que sejam relevantes a nível de gestão do quadro da plataforma *Trello* é bastante exigente para os gestores do projeto, pois demanda tempo e, muitas vezes, a verificação manual de cartões. Sendo assim, propõe-se uma solução voltada às necessidades destes gestores.

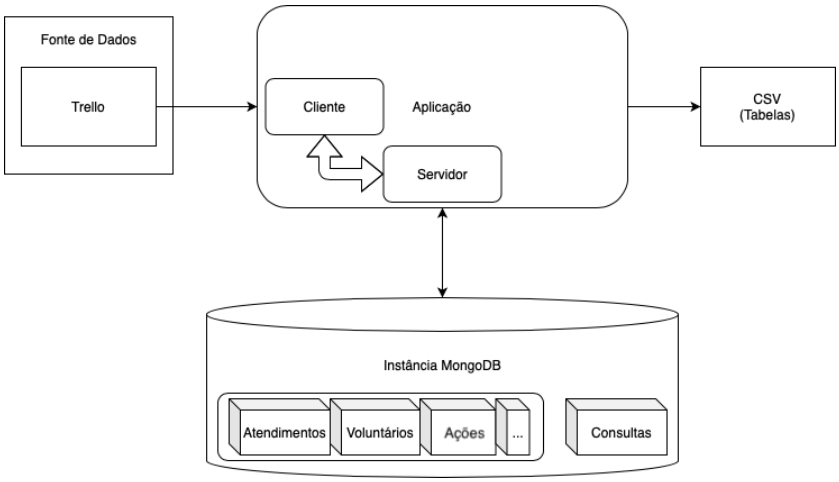


Figura 4.1 – Diagrama geral da aplicação

Através da aplicação, representada no seu contexto de uso na Figura 4.1, é possível importar dados referentes ao quadro *Trello* do projeto, consultá-los de maneira parametrizável e exportar os resultados destas consultas em um formato que viabiliza sua exploração posterior por Ferramentas Analíticas. Os dados extraídos do *Trello* são armazenados como documentos, em coleções pertencentes a uma instância de Banco de Dados MongoDB.

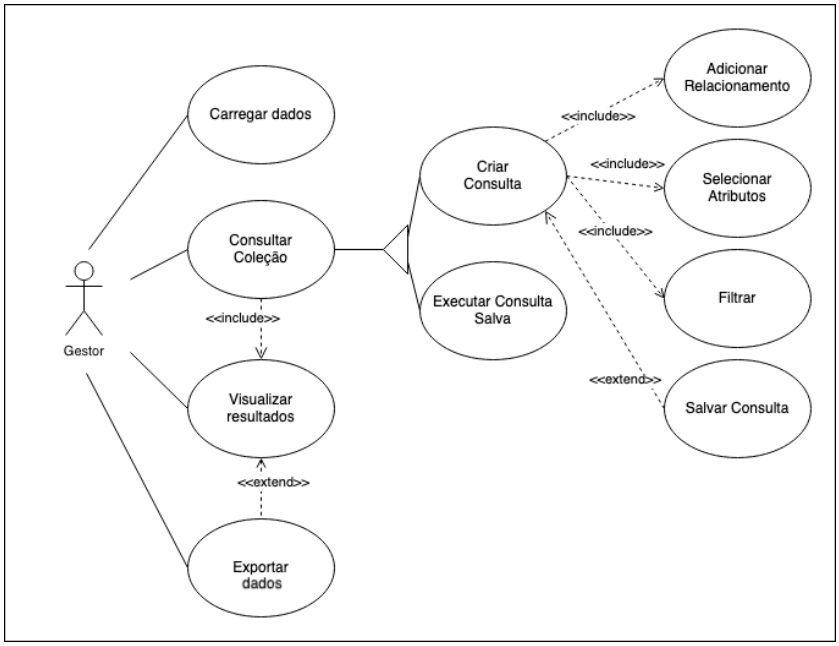


Figura 4.2 – Diagrama dos Casos de Uso da aplicação

A Figura 4.2 representa os casos de uso da aplicação, descrevendo as funcionalidades propostas neste trabalho para os usuários. Cada uma delas é explicada em maior detalhe na próxima seção.

4.2 Casos de Uso

A aplicação proposta neste trabalho visa oferecer aos gestores do projeto SOS PME funcionalidades relacionadas à obtenção de informação a partir dos dados extraídos do seu quadro da plataforma *Trello*.

4.2.1 Carregar Dados

A geração periódica de relatórios por parte da gestão do projeto torna indispensável que os dados disponíveis para consulta sejam os mais recentes. Assim, este caso de uso reflete a importação de dados novos pelo usuário na forma de um arquivo em formato *JSON*, que representa o quadro da plataforma *Trello*.

Após o *upload* deste arquivo, ele é formatado a fim de incluir certas informações que não fazem parte da estruturação original do arquivo, para que estas estejam disponíveis para consulta. Isto inclui atribuir a cada atendimento as suas datas de início e fim, o nome da lista em que o cartão a ele correspondente se encontra e as etiquetas que foram

associadas a ele.

4.2.2 Consultar Coleção

A principal necessidade da gestão do projeto SOS PME é obter informação pertinente ao seu funcionamento e aos atendimentos para fins de análise. Assim, a criação de consultas tem como objetivo oferecer a não-especialistas uma solução que permita selecionar e filtrar as informações relevantes para uma certa análise. Para possibilitar isso de maneira eficiente e prática, o usuário pode parametrizar uma consulta nova ou executar uma consulta pré-definida.

A fim de diminuir a complexidade envolvida na criação de uma consulta ao Banco de Dados, se propôs uma construção de consultas em quatro passos:

1. O usuário seleciona o artefato do quadro *Trello* a ser consultado (cartões ou membros).

2. Opcionalmente, pode-se selecionar um relacionamento, na forma de um artefato secundário (por exemplo, os membros de cada cartão). Assim, pode-se definir relacionamentos de Um para Muitos tanto dos cartões em relação a seus membros, quanto de cada membro para todos os cartões dos quais ele participou.

3. Escolhe-se os atributos dos artefatos que farão parte do resultado final (como o nome do cartão e o nome dos membros).

4. Pode-se filtar os resultados com os atributos a partir de valores de interesse específicos.

Por exemplo, pode-se consultar os cartões e relacionar cada um aos participantes atribuídos a ele. Assim, pode-se selecionar os atributos de nome do cartão, lista a qual ele pertence e nome dos usuários. Finalmente, os cartões consultados podem ser restringidos de maneira a só fazerem parte do resultado da consulta aqueles que fazem parte da lista de Atendimentos em Andamento.

Também existe a opção de se salvar uma consulta para ser executada em outro momento e atribuir um nome específico a ela. Com isto, procura-se facilitar a criação de consultas que são executadas múltiplas vezes. Quando o usuário deseja realizar uma destas consultas pré-definidas, ele pode escolhê-la a partir de um menu.

4.2.3 Exportar Dados

Ao executar uma consulta ao Banco de Dados, tenha ela sido definida naquele momento ou anteriormente, o usuário pode visualizar os seus resultados em uma tabela e exportá-los em formato *CSV*.

A exportação dos resultados de consultas realizadas na aplicação viabiliza a utilização destas informações pelos gestores do projeto. Este recurso possibilita a geração de documentos a partir desses resultados, por exemplo certificados de participação de voluntários. Além disso, a exportação desses resultados facilita também a geração de análises por ferramentas analíticas.

Como mencionado anteriormente, a ferramenta analítica *Google Data Studio*, que é utilizada pela gestão do projeto SOS PME, considera somente dados tabulares. Sendo assim, este caso de uso de exportação de dados tem como objetivo oferecer aos usuários uma maneira de extrair os dados em um formato compatível com a ferramenta. Para isto, aplica-se uma transformação para a Primeira Forma Normal[1] sobre os dados.

A Primeira Forma Normal define que os dados não podem possuir atributos multivalorados ou compostos. A fim de eliminar este tipo de atributo dos documentos armazenados pelo *MongoDB*, são utilizadas duas técnicas:

1) É desconstruído o relacionamento de Um para Muitos, caso ele exista. Considerando-se o exemplo do relacionamento de um cartão para todos os seus membros, isto significa que são criados vários documentos a partir do mesmo cartão, mas cada um contendo somente um dos membros.

2) Os documentos são achatados, o que significa fazer com que eles não possuam mais estruturas aninhadas, como listas e outros documentos e, ao invés disso, tenham todos os seus elementos como parte de um documento único.

O próximo Capítulo detalha a implementação desta solução.

[1] GeeksForGeeks - First Normal Form: <https://www.geeksforgeeks.org/first-normal-form-1nf/>

5 IMPLEMENTAÇÃO E TECNOLOGIAS UTILIZADAS

Este capítulo introduz detalhes sobre a implementação da aplicação proposta neste trabalho, e também apresenta as tecnologias nela utilizadas.

5.1 Tecnologias Utilizadas

a) Typescript: A linguagem de programação utilizada no desenvolvimento da aplicação se chama *Typescript*[1]. Ela é uma expansão da linguagem *Javascript*, criada a fim de oferecer uma alternativa à tipagem dinâmica e fraca da original. À exceção do seu sistema de tipos, código *Typescript* pode ser livremente convertido para código *Javascript*.

b) Javascript: Comumente abreviada como JS, *Javascript*[2] é uma linguagem de programação multi-plataforma e orientada a objetos, muito associada ao desenvolvimento *web*, já que hoje em dia todo navegador possui uma versão própria do seu ambiente de execução JS.

   Em anos recentes, o uso da linguagem Javascript tem aumentado em decorrência da popularização de *frameworks* voltados para o desenvolvimento *web*. A linguagem pode ser utilizada tanto para a implementação de servidores em módulos *back-end*, quanto de interfaces em módulos *front-end*. O desenvolvimento *back-end* é possível com o uso do ambiente de execução *Node.js*, que é independente de navegadores. Já o desenvolvimento *front-end* é auxiliado pelas bibliotecas voltadas à criação de interfaces, como é o caso de *React*, uma das mais populares atualmente.

c) React: A biblioteca *React*[3] é um *framework* voltado ao desenvolvimento de interfaces de usuário que introduz a noção de componentes. No desenvolvimento *React*, todo módulo da aplicação é um Componente, i.e. um pedaço isolado da aplicação, comumente implementado como uma classe da linguagem JS. Todo Componente possui um método de *render*, através do qual ele se torna visível na interface da aplicação. Em geral, uma aplicação é implementada como um Componente único que contém outros Componentes mais especializados.

[1] Typescript: <https://www.typescriptlang.org>
[2] Javascript: <https://www.javascript.com>
[3] React: <https://pt-br.reactjs.org>

A fim de promover o encapsulamento de cada Componente, utilizam-se em *React* os conceitos de um estado interno a ser mantido e de um ciclo de vida. O estado de um Componente reflete um conjunto de informações que podem mudar dinamicamente durante o seu ciclo de vida. Ao contrário das propriedades externas que ele pode receber de Componentes de nível mais alto, o seu estado interno é mutável e pode ser modificado por eventos da interface (como por exemplo o clique de um botão).

O ciclo de vida de um Componente se inicia com a sua montagem, através da chamada ao seu método de *render*, e se encerra com sua desmontagem. Geralmente, quando é necessário realizar chamadas externas a alguma API para obtenção de dados, estas ocorrem na etapa de inicialização do Componente, pois assim se garante que elas só serão realizadas uma única vez. Enquanto "vivo", o estado do Componente pode ser atualizado, causando novas chamadas de *render*.

A escolha pela biblioteca *React* para o desenvolvimento do módulo *front-end* da aplicação foi motivada pela sua popularidade, já que ela é considerada o padrão de indústria para aplicações *web*.

d) Node.js: *Node.js*[4] é um ambiente de execução assíncrono e orientado a eventos para *Javascript* e, por extensão, para *Typescript*. Ele permite a execução de código JS fora de um navegador, motivo pelo qual foi escolhido para a implementação do módulo *back-end*.

e) Express: O framework *Express*[5] é utilizado para implementar roteamento no módulo *back-end*. Assim, é possível definir tratamentos diferentes para pedidos HTTP recebidos em cada rota (ou, "ponto de acesso"). Quando um pedido HTTP é enviado a esta rota, ele é recebido, encaminhado ao componente correspondente e processado. Isto facilita também a implementação posterior de novas rotas.

5.2 Arquitetura da Aplicação

5.2.1 Visão Geral

Para a aplicação proposta neste trabalho, optou-se por uma arquitetura baseada no modelo cliente-servidor. O cliente (*Front-End*) é análogo à camada de apresentação,

[4]NodeJS: <https://nodejs.org>
[5]Express: <https://expressjs.com/>

enquanto o servidor (*Back-End*) serve como a camada de acesso aos dados.

Esta arquitetura garante uma separação entre estas responsabilidades. Dada a natureza *web* da aplicação, foi importante garantir que as funcionalidades de consulta e visualização de resultados fossem devidamente isoladas das tarefas de manipulação e acesso ao Banco de Dados. O restante desta seção discute a arquitetura e componentes do Front-end e do Back-end. A Seção 5.3 descreve com mais detalhes como cada funcionalidade da Figura 4.2 é implementada nos lados cliente e servidor.

5.2.2 Front-End

O *Front-End* é responsável pela interação dos gestores com o sistema, a fim de prover as funcionalidades resumidas na Figura 4.2. A interação do usuário com o sistema se dá através de uma interface gráfica, que permite definir os parâmetros das ações do usuário e visualizar resultados. Nesta aplicação, ele possui as seguintes responsabilidades:

- permitir a criação e parametrização de buscas sobre os dados;
- exibir os resultados destas buscas para os usuários;
- formatar estes dados corretamente;
- permitir a atualização do Banco de Dados.

Ele foi implementado com base no padrão de arquitetura MVC (ATWOOD, 2008) com a biblioteca *React*. Há 4 tipos de Componentes neste módulo da aplicação: *API*, *App*, *Página* e *Elemento*. A Figura 5.1 mostra como estes Componentes interagem no contexto MVC.

Eles são:

- *App*: Só há uma instância do Componente "*App*". Ele é responsável por mapear cada rota de navegação da aplicação ao Componente responsável, como pode ser visto na Figura 5.2. Estes componentes são as páginas, como por exemplo a da Figura 5.6a.
- *API*: O componente API também é uma instância única, e é responsável por se comunicar com o Back-End e recuperar dados. Como um componente, ela disponibiliza um conjunto de funções para o resto da aplicação que abstraem esta

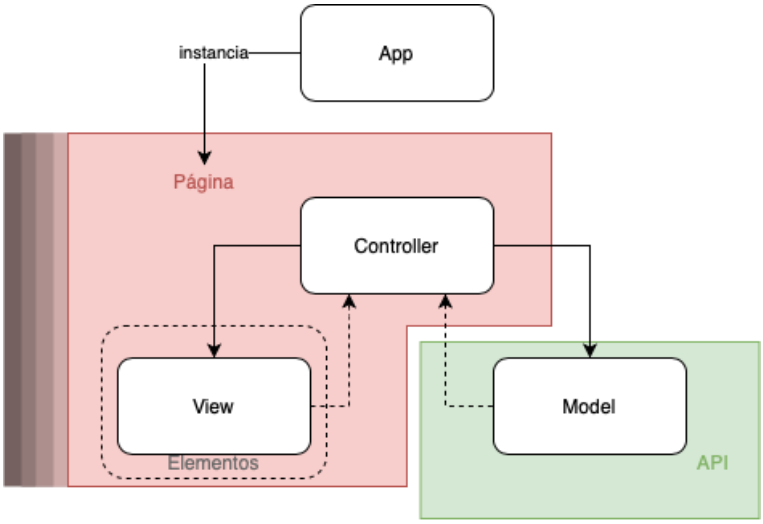


Figura 5.1 – Diagrama do Front-End como um MVC

comunicação, simplesmente retornando os dados requeridos. Internamente, no entanto, cada função envia uma requisição HTTP a uma rota (i.e. URL) específica do servidor.

- *Página*: Cada componente Página é responsável pelo controle de uma parte da aplicação. Cada Página mantém um estado interno e pode fazer chamadas à API. Toda a lógica que guia os componentes visuais (elementos) também é responsabilidade de uma Página, como por exemplo, a função a ser chamada quando determinado botão é clicado.

- *Elemento*: são componentes visuais interativos utilizados pelas páginas para a inserção ou apresentação de dados (e.g. caixas de texto, botões, listas de seleção, etc.).

```
function App(): JSX.Element {
  console.log('Rendering App');
  return (
    <Router>
      <Switch>
        <Route path="/" exact component={Home} />
        <Route path="/create" exact component={Create} />
        <Route path="/run" exact component={Run} />
        <Route path="/query" exact component={Query} />
      </Switch>
    </Router>
  );
}
```

Figura 5.2 – Trecho do Componente React “App”

5.2.3 Back-End

O *Back-End* não é diretamente visível ao usuário, e possui as seguintes responsabilidades:

- armazenar os dados em uma instância de Banco de Dados MongoDB;
- executar as consultas aos dados recebidas do *Front-End*;
- transferir os resultados destas consultas de volta ao *Front-End*, para serem exibidos;
- atualizar o Banco de Dados.

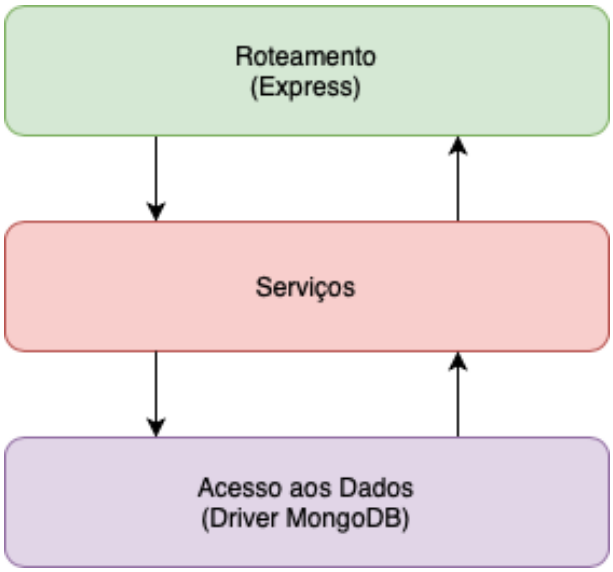


Figura 5.3 – Diagrama do Back-End em Camadas

O *Back-End* da aplicação foi implementado utilizando-se o padrão de arquitetura de camadas (SZALAI, 2019), seguindo o princípio da Separação de Conceitos. Estas camadas estão representadas na Figura 5.3. Esta arquitetura de 3 camada permite isolar a lógica de negócios das rotas de *API* da seguinte forma:

- Roteamento: A camada de Roteamento é responsável pela comunicação, ou seja, por receber pedidos e enviar respostas. As rotas utilizam serviços no processamento das respostas.
- Serviços: A lógica de negócios fica encapsulada na camada de Serviços. Na aplicação, existe um serviço principal, o do MongoDB, que expõe funcionalidades de alto nível para a camada de Roteamento.

- Acesso aos Dados: Ela é a responsável por estabelecer e disponibilizar a conexão à instância MongoDB, bem como implementar as funções oferecidas na camada de Serviços.

5.3 Implementação das Funcionalidades

5.3.1 Funcionalidades do *Back-End*

As principais funcionalidades do módulo servidor da aplicação são as de atualizar os dados do Banco de Dados, executar as consultas a ele, e armazenar consultas pré-definidas, as quais também podem ser executadas.

a) **Carregar dados:**

Ao receber um pedido referente à atualização dos dados, o servidor recebe também uma cópia do arquivo contendo os dados exportados da plataforma *Trello*. Como mencionado anteriormente e ilustrado pela Figura 2.5, estes dados existem em formato chave-valor. As chaves representam os diferentes tipos de artefatos da plataforma *Trello*, existindo uma coleção do Banco de Dados MongoDB para cada um. As coleções são chamadas *"cards"*, *"members"*, *"actions"* e *"checklists"*. Já os valores associados a cada chave e, portanto, a cada artefato e coleção, são formatados e utilizados para atualizar os documentos pertencentes à coleção correspondente.

Também são sintetizadas certas propriedades dos dados, tais como as datas de criação e finalização do atendimento dos cartões. A data de criação do cartão é derivada a partir do seu campo de identificação da plataforma *Trello*. Já a data de conclusão do atendimento é criada consultando-se os objetos de "ações"do quadro, a fim de achar a ação que descreve a movimentação do cartão em questão para a lista de "Atendimentos concluídos", como mostra a Figura 5.4.

Caso o nome da lista que contém o cartão seja o de "Atendimentos concluídos", procura-se a ação identificada por ter movimentado o cartão para aquela lista. Por fim, caso esta ação não exista, presume-se que a data da última atividade registrada para aquele cartão é uma boa aproximação da data de finalização de atendimento. Caso a lista que contém o cartão não seja a de atendimentos concluídos, se retorna um valor nulo.

```
function createDateFinished(card: any, lists: Array<any>, actions: Array<any>): string | null {
  const l = lists.find((l) => l != undefined && l.id == card.idList);

  if (l !== null && l.name == 'Atendimentos concluídos') {
    const a = actions.find(
      (a) => a.data.listAfter !== undefined && a.data.listAfter.name == l.name && a.data.card.id == card.id,
    );
    if (a != undefined) {
      return a.date;
    } else {
      return card.dateLastActivity;
    }
  }

  return null;
}
```

Figura 5.4 – Função de criação da data de Finalização do Atendimento

Alguns dados presentes no arquivo de origem podem ser inter-relacionados, como é o caso das listas que contém cartões. Originalmente, um cartão contém somente um campo relativo ao identificador único de uma lista, porém, durante a formatação dos dados, estes identificadores são substituídos pelo nome da lista que eles representam. Assim, não é necessário armazenar as listas no Banco de Dados nem realizar operações de junção entre coleções para recuperar esta informação, já que agora ela é parte dos cartões.

Por outro lado, a relação entre um cartão e seus membros é mantida como presente no quadro *Trello*, na forma de uma lista de identificadores únicos. Ao invés de anexados ao cartão, os membros são mantidos numa coleção própria, já que representam dados mais complexos. Quando se deseja consultar esta relação, é através destes identificadores que se realiza a operação de junção.

b) Executar consulta:

A funcionalidade de Executar Consultas aos dados é implementada de duas maneiras diferentes, dependendo da consulta estar sendo criada ou ser pré-definida.

Internamente, as consultas pré-definidas (como vistas na Figura 5.6b) são armazenadas em um Banco de Dados MongoDB próprio. Assim, quando é recebido um pedido para listar todas as consultas salvas ou para deletar uma delas, o *Back-End* pode simplesmente comunicar-se com o SGBD.

Uma consulta pode ser tanto recuparada do Banco de Dados, como é o caso para consultas salvas, quando extraída do corpo do pedido recebido. A Figura 5.5 exemplifica os passos relacionados à execução de uma consulta, desde o recebimento de um pedido até o envio de uma resposta contendo os resultados.

A função *Aggregate* da camada de Serviços é uma interface para o *framework* de mesmo nome do SGBD MongoDB. Ele permite realizar consultas baseadas em diferentes passos de processamento de dados. Por exemplo, se há uma relação na consulta, é realizado o passo de "*lookup*"(do inglês, pesquisa), que funciona para juntar documentos vindos de coleções diferentes. Neste caso, a chave estrangeira é determinada programaticamente a partir do nome da relação. Por fim, se há filtros na consulta, estes são adicionados à etapa "*match*"(do inglês, comparar), responsável por restringir os resultados somente aos documentos cujos atributos filtrados correspondem ao valor estabelecido.

```
route.get('/run/saved/:queryID', async (req: Request, res: Response, next: NextFunction) => {
  try {

    const mongoServiceInstance = Container.get(MongoService);
    let query = await mongoServiceInstance.GetQuery(req.params.queryID);

    query = JSON.parse(query.query) as Query;

    const pipeline = createPipeline(query);
    const queryResult = await mongoServiceInstance.Aggregate(query.collection, pipeline, query.project);

    return res.json(queryResult).status(200);

  } catch (e) {
    console.log(e);
    return next(e);
  }
});
```

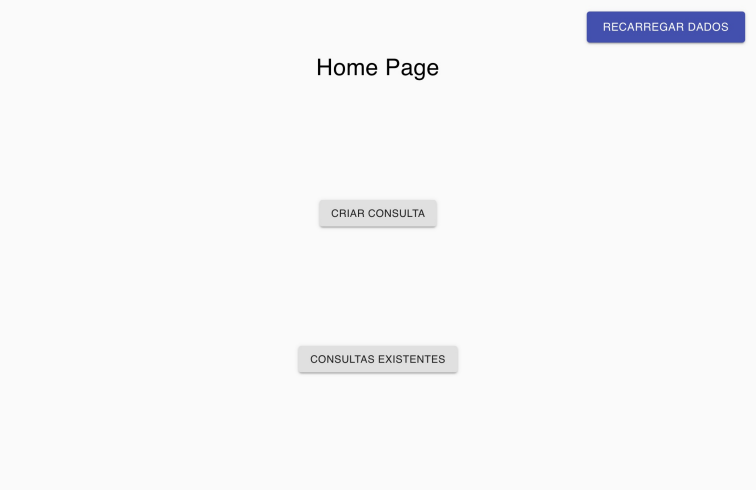
Figura 5.5 – Rota responsável pela execução de uma Consulta pré-definida

Há uma etapa de processamento no método *Aggregate* que também auxilia na transformação dos resultados para a Primeira Forma Normal. Se trata da etapa de "*unwind*" (do inglês, "desenrolar"), que permite emitir um documento para cada valor de um campo multi-valorado, repetindo-se todos os outros campos. No caso de existir um relacionamento na consulta, este método é utilizado para dissolver a relação de Um para Muitos.

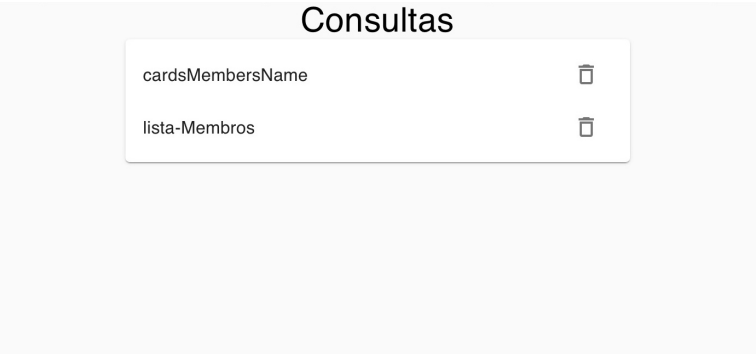
5.3.2 Funcionalidades do *Front-End*

Como a interação dos usuários com a aplicação se dá através do Front-End, também a maioria das funcionalidades mostradas na Figura 4.2 são iniciadas através da interface de usuário, que é responsabilidade deste módulo. A Figura 5.6a retrata a tela inicial da aplicação, a partir da qual se pode recarregar os dados, navegar para a página de criação de consultas ou para a página que lista as consultas existentes. A Figura 5.6b mostra esta página, onde as consultas pré-definidas podem ser executadas ou deletadas.

A funcionalidade de criação de consultas é implementada em quatro passos, como



(a) Tela Inicial da Aplicação



(b) Tela de Consultas Pré-Definidas

Figura 5.6 – Interface Inicial da Aplicação

descrito na Seção 4.2.2. Inicialmente, o Componente responsável por esta página faz uma chamada à API e recebe uma lista de todas as coleções presentes no Banco de Dados, que também contém seus atributos e relações possíveis.

O restante desta seção é organizado em exemplos, a fim de discutir a criação e execução de algumas consultas complexas relativas ao projeto SOS PME, apresentadas previamente na Seção 3.2.

a) **Obter as datas de início e fim de cada atendimento finalizado:**

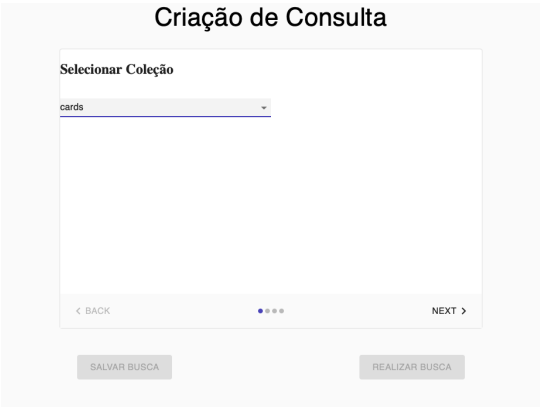
A Figura 5.7 mostra a criação da consulta com o objetivo de obter as datas de início e fim de cada atendimento finalizado. Esta consulta visa exemplificar os passos necessários para a criação de uma consulta a atributos simples de um mesmo tipo de entidade. A Figura 5.9 mostra os resultados desta consulta em forma de tabela.

1. O primeiro passo é selecionar a coleção principal a ser consultada. Neste caso, se seleciona a coleção *cards* (Figura 5.7a), que são os elementos da plataforma *Trello* que representam os atendimentos.

2. A seleção de uma relação (Figura 5.7b) é um passo opcional, e não é necessário para esta consulta.

3. No passo seguinte, como mostra a Figura 5.7c, são selecionados os atributos dos cartões que se deseja visualizar: "*dateCreated*" e "*dateFinished*" (datas de criação e de término, respectivamente), "*name*" (nome do atendimento como aparece no cartão) e "*listName*" (nome da lista que contém o cartão). Este passo é obrigatório. As ações de Salvar e Realizar Busca só se tornam habilitadas quando existe pelo menos um atributo selecionado, refletindo a necessidade de haver pelo menos um campo a ser consultado.

4. No passo de filtragem dos atributos, que também é opcional, só é possível filtrar atributos que farão parte do resultado final. Ele não permite filtrar atributos aninhados, já que toda comparação realizada é exata. A Figura 5.7d mostra a definição do filtro desta consulta, que só permite que cartões da lista de atendimentos concluídos façam parte do resultado.



(a) Seleção de Coleção - cards

(b) Seleção de Relação

(c) Seleção de Atributos

(d) Criação de Filtros

Figura 5.7 – Passos de criação da consulta 1

A Figura 5.8 traz a consulta MongoDB criada por estes passos, executada pelo módulo *Back-End*. A partir da tela de resultados (Figura 5.9), também é possível exportar os dados.

```
db.collection("cards")
    .aggregate([
        {
         "$match": { "listName": "Atendimentos concluídos" }
        }])
    .project({
        "dateCreated": 1,
        "dateFinished": 1,
        "listName": 1,
        "name": 1,
        "_id": 0
})
```

Figura 5.8 – Código MongoDB da consulta 1

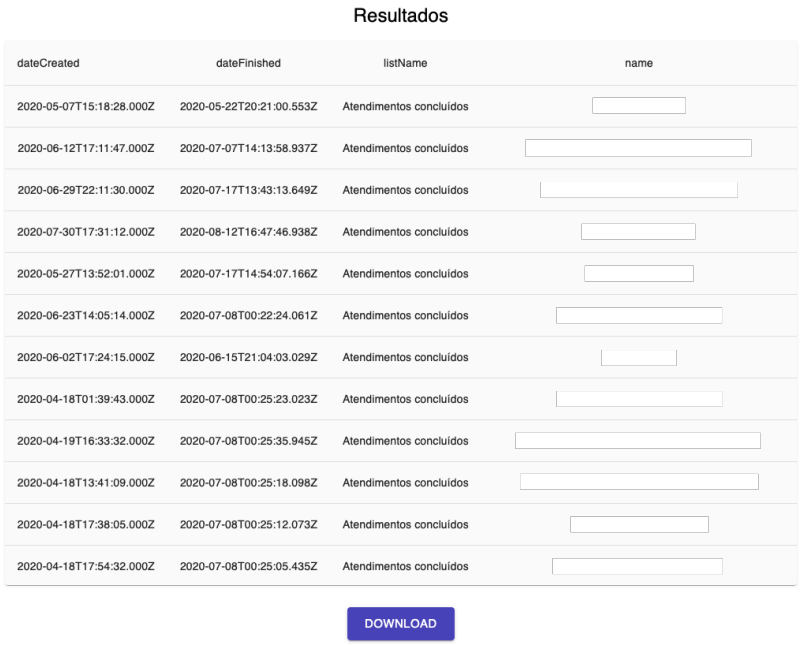


Figura 5.9 – Resultados (anonimizados) da consulta 1

b) **Obter a relação de demandas de cada atendimento:**

A consulta pelas demandas de cada atendimento exemplifica a visualização de um atributo multi-valorado, que precisa ser "aplainado"na visualização de seu resultado.

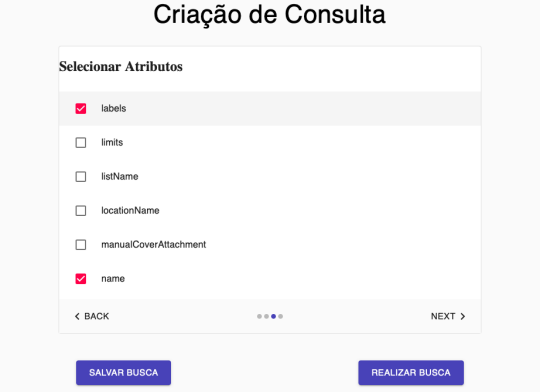


Figura 5.10 – Seleção de Atributo multi-valorado (consulta 2)

1-2. Os dois primeiros passos desta consulta são idênticos aos da consulta anterior (Figuras 5.7a e 5.7b).

3. No passo de seleção de atributos (Figura 5.10), são selecionados o nome de cada cartão e suas "*labels*" (etiquetas), que representam as diferentes demandas de cada atendimento.

4. Nesta consulta, não é realizado o passo de criação de filtros.

```
db.collection("cards")
    .aggregate([])
    .project({
        "labels": 1,
        "name": 1,
        "_id": 0
})
```

Figura 5.11 – Código MongoDB da consulta 2

A consulta executada é representada na Figura 5.11. Nenhum passo de processamento de dados é passado para a função "*aggregate*", pois não foi definida nenhuma relação com outra coleção, nem foram filtrados atributos.

A Figura 5.12 mostra os resultados desta consulta. Ela ilustra também o resultado da formatação dos resultados em dados tabulares. Pode-se observar como o atributo multi-valorado "labels" é aplainado, como parte da transformação destes dados para a Primeira Forma Normal. A partir deste atributo são criadas 8 diferentes colunas, o que significa que o documento com o maior número de etiquetas continha 8 delas. Aqueles com menos etiquetas ficam sem valores correspondentes.

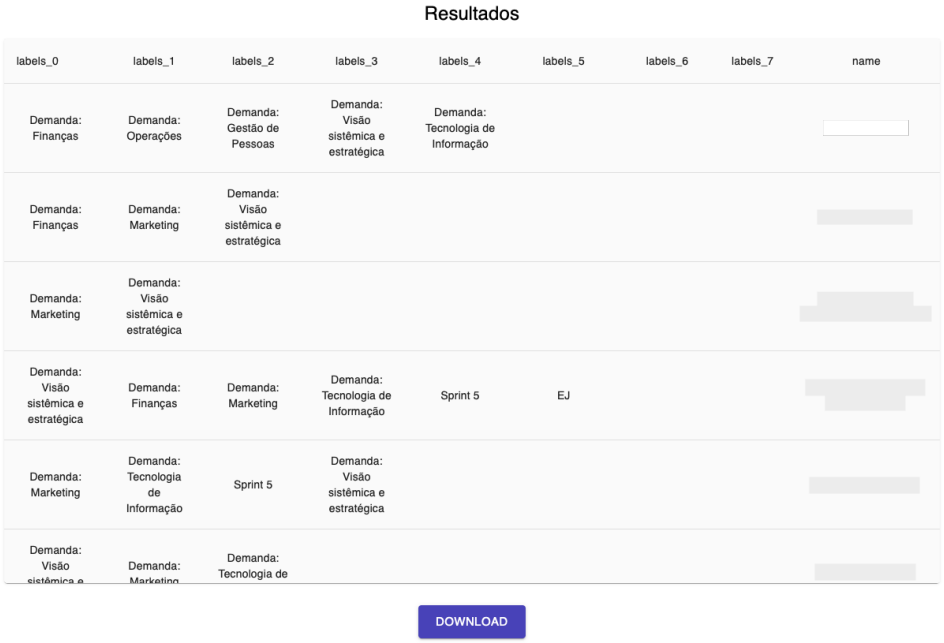


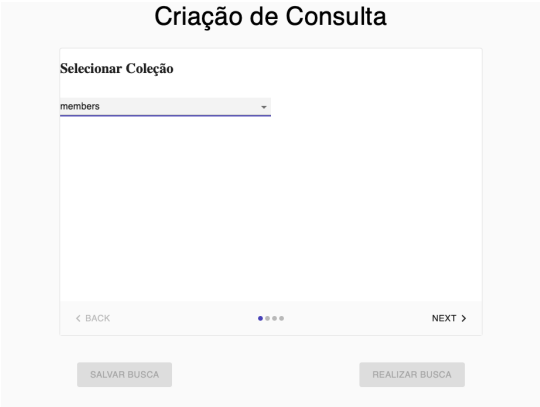
Figura 5.12 – Resultado de consulta 2 (às demandas)

c) **Gerar relatórios de participação dos atendimentos concluídos:**

Esta consulta retrata uma atividade central para a gestão do projeto SOS PME: obter a relação entre cada voluntário e os atendimentos finalizados dos quais ele participou, a fim de gerar relatórios. A Figura 5.13 mostra o passo-a-passo de sua criação.

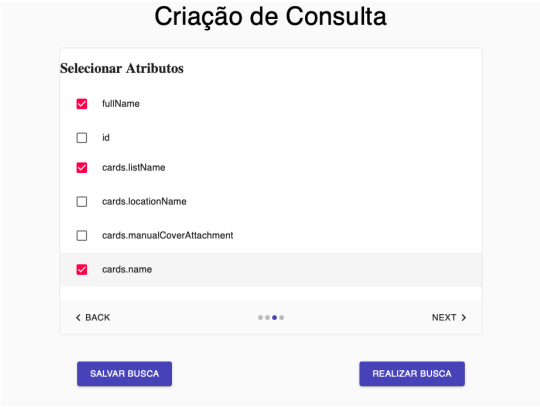
1. No primeiro passo, é selecionada a coleção de "*members*" (Figura 5.13a). Para o projeto SOS PME, os membros de um cartão do seu quadro Trello representam os voluntários responsáveis por aquele atendimento.
2. A Figura 5.13b mostra que, para esta consulta, os cartões são selecionados como elementos secundários.
3. No passo de seleção de atributos (Figura 5.13c) pode-se ver que os atributos relativos à coleção secundária são identificados pelo seu nome. Neste caso, foram selecionados o "*fullName*" (nome completo) dos membros, bem como "*cards.name*", o nome de cada cartão/atendimento, e "*cards.listName*", a lista a qual o cartão pertence.
4. Como mostra a Figura 5.13d, só fazem parte do resultado final os cartões localizados na lista de atendimentos concluídos.

Os resultados desta consulta estão ilustrados na Figura 5.14. A fim de manter os



(a) Seleção de Coleção - members

(b) Seleção de Relação - cards



(c) Seleção de Atributos

(d) Criação de Filtros

Figura 5.13 – Passos de criação da consulta 3

participantes do projeto anônimos, foram utilizadas cores para identificá-los. Assim, pode-se ver que os resultados são organizados pelos elementos da coleção primária selecionada. Os valores de "*fullName*", dos membros, se repetem múltiplas vezes por conta do desenrolamento do relacionamento de Um para Muitos entre um membro e os seus cartões. Isto é possível através da consulta MongoDB retratada na Figura 5.15.

Na função "*aggregate*", o passo "*lookup*" permite realizar a junção dos documentos da coleção de membros com os da coleção de cartões, enquanto o passo "*unwind*" determina que cada valor de "cards" seja emitido como um documento separado.

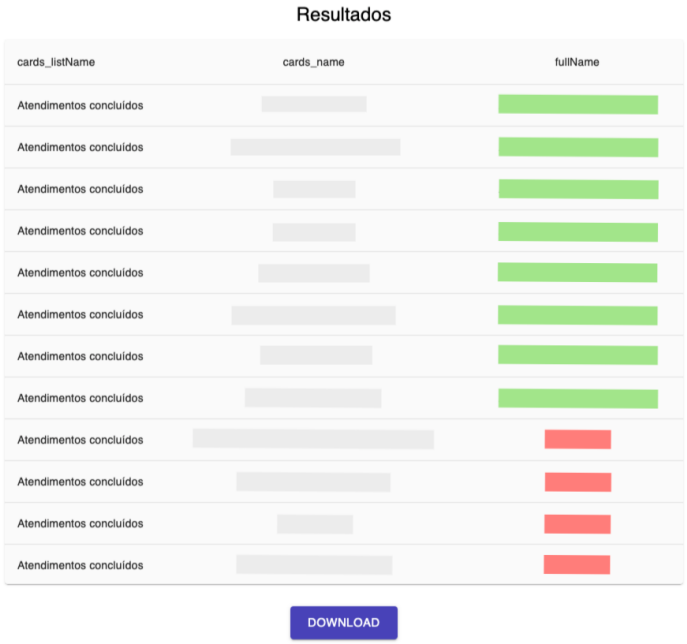


Figura 5.14 – Tela de resultados (anonimizados) da consulta 3

```
db.collection("members")
    .aggregate([
        {
         "$lookup": {
             "from": "cards",
             "localField": "id",
             "foreignField": "idMembers",
             "as": "cards" }
        },
        {
        "$unwind": "$cards"
        }
        ])
    .project({
        "fullName": 1,
        "cards.listName": 1,
        "cards.name": 1,
        "_id": 0
})
```

Figura 5.15 – Código MongoDB da consulta 3

6 CONCLUSÃO

O projeto SOS PME foi criado em Março de 2020 com o objetivo de auxiliar pequenas e médias empresas a superar novas dificuldades causadas pela pandemia da COVID-19. A gestão interna do projeto visa coletar informações referentes aos atendimentos oferecidos às empresas para fins de análise, buscando possibilitar uma troca de aprendizados entre as equipes de voluntários envolvidas. Considerando as dificuldades enfrentadas pela gestão do projeto em extrair estas informações do quadro *Trello* utilizado para o acompanhamento dos atendimentos, este trabalho se propôs a desenvolver uma solução que permita consultar estes dados, carregar versões atualizadas deles e exportá-los de maneira apropriada para uso em análises gerenciais.

A fim de oferecer uma maneira simples de se realizar atividades complexas de consulta sobre os dados do projeto, foi desenvolvida uma aplicação *web* voltada a não especialistas em Bancos de Dados. A aplicação permite obter as informações relevantes às análises consideradas difíceis pela gestão do projeto SOS PME. Além disso, as funcionalidades de se executar consultas previamente definidas e da transformação dos resultados para um formato tabular contribuem para que a ferramenta possa ser facilmente utilizada pela gestão do projeto SOS PME para viabilizar a exploração dos dados por outras ferramentas analíticas.

Elencamos, junto à gerência do projeto, consultas consideradas úteis do ponto de vista de gestão, e demonstramos através de exemplos que as mesmas são atendidas através de nossa solução. A preocupação em organizar a aplicação segundo bons princípios de desenvolvimento de software, como separação de conceitos e padrões arquiteturais, permite que a solução seja melhorada e estendida.

A solução proposta neste trabalho ainda possui espaço para melhorias no seu funcionamento. Embora a infraestrutura da aplicação permita que novas funcionalidades sejam rapidamente adicionadas tanto ao módulo *Front-End* quando ao módulo *Back-End*, ainda seria necessário o envolvimento de um especialista para se realizar consultas mais complexas, por exemplo com a criação de filtros sobre atributos não-textuais e/ou aninhados. Além disso, o método utilizado para se inferir a data de conclusão de um atendimento é baseado no histórico de ações do quadro *Trello*, que a qualquer momento só mantém as 1.000 ações mais recentes ocorridas, o que eventualmente pode levar a uma data incorreta sendo atribuída à conclusão de um atendimento. Também existem documentos anexados aos cartões *Trello*, que podem conter informações importantes relativas às lições aprendi-

das com os atendimentos, mas não há nenhum método de extraí-los através da aplicação. Apesar de existir código no módulo *Back-End* que realiza o *download* destes arquivos e realiza processamento de linguagem natural sobre eles, esta funcionalidade não é atualmente disponibilizada.

Tendo em vista os pontos acima mencionados, também existem trabalhos a serem futuramente completados. A interface visual do módulo *Front-End*, por exemplo, pode ser melhorada a fim de ser mais clara. Dada a natureza *web* da aplicação, também se poderia procurar soluções para os problemas de segurança com o acesso remoto ao Banco de Dados e disponibilizá-la na *internet*. Por fim, seria importante implementar no futuro um sistema de atualização periódica dos dados da plataforma *Trello*, para que não seja mais necessário importar um arquivo.

REFERÊNCIAS

ATWOOD, J. **Understanding Model-View-Controller**. 2008. Available from Internet: <https://blog.codinghorror.com/understanding-model-view-controller/>.

BUSCHMANN, F. et al. **Pattern-Oriented Software Architecture**. Hoboken, New Jersey: Wiley, 1996.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. São Paulo, São Paulo: Pearson, 2005.

ESPINHA, R. G. **Kanban: O que é e TUDO sobre como gerenciar fluxos de trabalho**. 2019. Available from Internet: <https://artia.com/kanban/>.

PAIVA, R. **O que é e como funciona o Map Reduce usado pelo Google**. 2011. Available from Internet: <http://blog.werneckpaiva.com.br/2011/08/como-funciona-o-map-reduce-usado-pelo-google/>.

REDMOND, E.; WILSON, J. R. **Seven Databases in Seven Weeks**. Dallas, Texas: The Pragmatic Bookshelf, 2012.

SZALAI, A. **Padrões de Projeto - Arquitetura em Camadas**. 2019. Available from Internet: <http://blog.askm.com.br/2019/02/01/padroes-de-projeto-arquitetura-em-camadas/>.

TRELLOBLOG. **How WIRED UK Uses Trello To Publish Articles For A Global Audience**. 2020. Available from Internet: <https://blog.trello.com/wired-uk-template-story>.