PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

VisMaker: uma aplicação orientada a perguntas para visualização de dados

Matheus Bispo da Silva

Projeto Final de Graduação

Centro Técnico Científico - CTC
Departamento de Informática
Curso de Graduação em Engenharia da Computação

Rio de Janeiro, julho de 2022

Matheus Bispo da Silva

VisMaker: uma aplicação orientada a perguntas para visualização de dados

Relatório de Projeto Final, apresentado ao programa de graduação da PUC-Rio como requisito parcial para a obtenção do título de Engenheiro de Computação ou Bacharel em Informática ou Bacharel em Ciência da Computação.

Orientadora: Simone Diniz Junqueira Barbosa

Rio de Janeiro

Dezembro de 2022.

Resumo

DA SILVA, Matheus Bispo. BARBOSA, Simone Diniz Junqueira. **VisMaker: uma aplicação orientada a perguntas para visualização de dados**. Rio de Janeiro, 2022. 22p. Relatório de Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O objetivo deste trabalho é a refatoração de uma aplicação chamada VisMaker. Dessa forma, busca-se reconstruir a ferramenta utilizando linguagens de programação mais adequadas ao contexto, bem como a aplicação de uma arquitetura eficiente, promovendo assim tanto ganho de performance quanto adaptabilidade e facilidade de futuras implementações.

Palavras-chave

Recomendação de visualização. Exploração visual de dados. Ferramenta de visualização. Visualização de informações.

Abstract

DA SILVA, Matheus Bispo. BARBOSA, Simone Diniz Junqueira. **VisMaker: a question-oriented application for data visualization**. Rio de Janeiro, 2022. 22p. Final Paper Report – Department of Informatics, Pontifical Catholic University of Rio de Janeiro - PUC-Rio.

The main objective of this project is the refactoring of a software called VisMaker. Thus, we look for a complete reconstruction of the program using both more adequate programming languages and an efficient architecture, promoting not only performance upgrade, but also adaptability and ease of future feature implementations.

Keywords

Visualization recommendation. Visual data exploration. Visualization tool. Information visualization.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. Introdução

"The ability to **take** data – to be able to **understand** it, to **process** it, to **extract value** from it, to **visualize** it, to **communicate** it – is going to be a hugely important skill in the next decades." - Hal Varian, Chief Economist, Google, 2009.[1]

É notório que a informação (mais especificamente, os dados), ao longo da história, agiu como um dos principais agentes nos processos da evolução tecnológica humana. Desde o método empírico de descoberta do tratamento de doenças na antiguidade até o reconhecimento de padrões no mercado financeiro nos dias atuais, por exemplo, pode-se perceber que a aquisição de dados e a forma como estes foram ou são analisados têm sido fundamentais para a realização de grandiosas descobertas. [2]

Atualmente, acompanhamos um processo de constante digitalização dos dados, em um mundo cuja onda de informações geradas e exibidas é cada vez maior e mais rápida – o que conhecemos hoje como *Big Data*.

Embora a tecnologia hoje nos ofereça excelentes ferramentas de visualização de dados, pode-se perceber que, mesmo entre os profissionais da área, é exigido um conhecimento técnico avançado para que possam ser obtidos os insights desejados. [3]

Dentro desse contexto, foi imaginado o **VisMaker** [4], uma aplicação voltada para a visualização de dados. Com uma interface intuitiva, o usuário não precisa de conhecimento técnico avançado, bastando que ele escolha a pergunta que precisa ser respondida. A partir desse ponto, a própria ferramenta se encarrega de analisar a informação e gerar um conjunto de visualizações adequadas para que o utilizador possa obter as respostas à sua pergunta.

Neste projeto, é proposta a completa refatoração da aplicação, com o objetivo de solidificar a estrutura do projeto, de modo a utilizar ao máximo e de maneira mais eficiente as tecnologias (linguagens, frameworks, arquiteturas e demais componentes de infraestrutura de software) disponíveis no momento, fazendo com que o VisMaker torne-se uma ferramenta mais completa e que, ao final, futuras implementações, bem como a própria manutenção da aplicação em si, torne-se uma tarefa bem mais fácil.

2. Situação atual

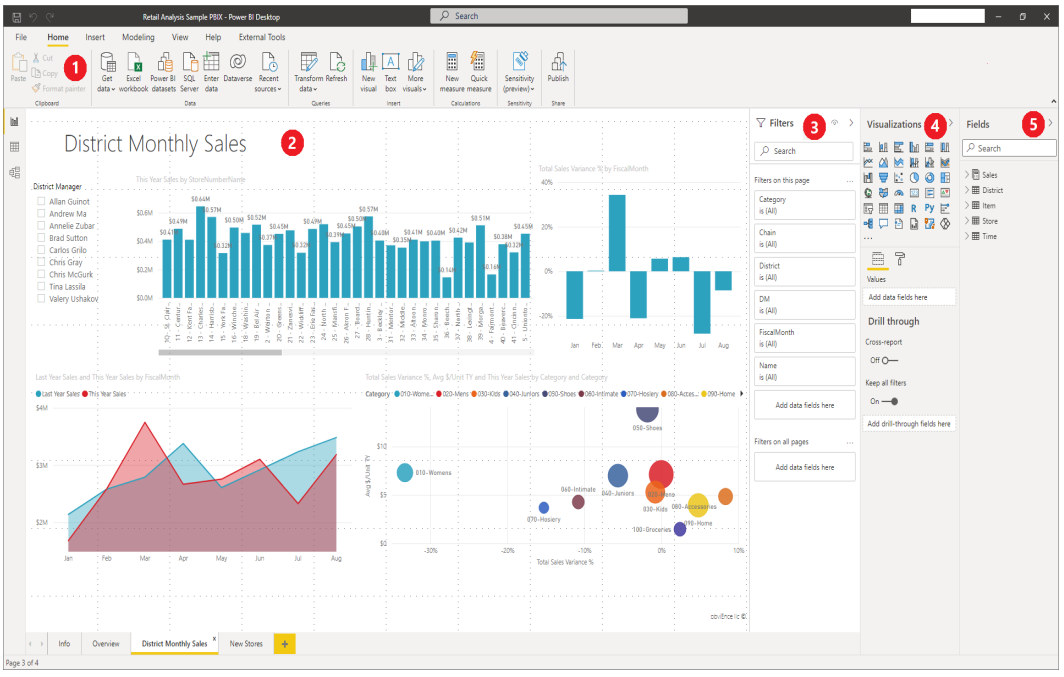
A aplicação aqui proposta possui algumas inspirações às quais pode ser relacionada, que serão apresentadas e comparadas a seguir.

Ao falar de Visualização de Dados, não podemos deixar de pensar em uma das principais ferramentas utilizadas no mercado: o Power BI.

Com uma interface limpa, intuitiva e um leque robusto de funcionalidades (desde filtros até as poderosas fórmulas DAX), o Power BI é uma ferramenta avançada de exploração e análise de dados, contando com uma integrada seção de tratamento das fontes de dados inseridas.

Na Figura 1, pode-se ter uma visão geral da interface de visualização do Power BI [5], contendo, à direita, as colunas correspondentes às dimensões disponíveis, os gráficos (juntamente com os seus campos de configuração) e os filtros do relatório.

Figura 1 - Interface do Power BI



De modo semelhante, outra ferramenta profissional de amplo uso no mercado é o Tableau.

Possuindo uma interface semelhante ao Power BI, o Tableau é uma ferramenta tão robusta e eficiente quanto ele. Nesta ferramenta, é válido ressaltar a funcionalidade de relacionamento das fontes de dados, que é feita de maneira bastante intuitiva (ao contrário do Power BI).

Além disso, a fase de tratamento dos dados é realizada, se necessário, numa ferramenta à parte (Tableau Prep), fazendo com que o software principal tenha um desempenho melhor e exija menos poder de processamento.

Na Figura 2 é demonstrada a interface do Tableau, que remete bastante à utilizada pelo Power BI. A partir desta análise, pode-se inferir que tal modelo de interface possui um ótimo desempenho com relação à experiência e utilização do usuário, podendo assim servir como inspiração para o VisMaker.

Complementarmente, a Figura 3 exibe a interface de relacionamento entre fontes de dados, exemplificando a funcionalidade disponibilizada pelo software.

Figura 2 - Interface do Tableau

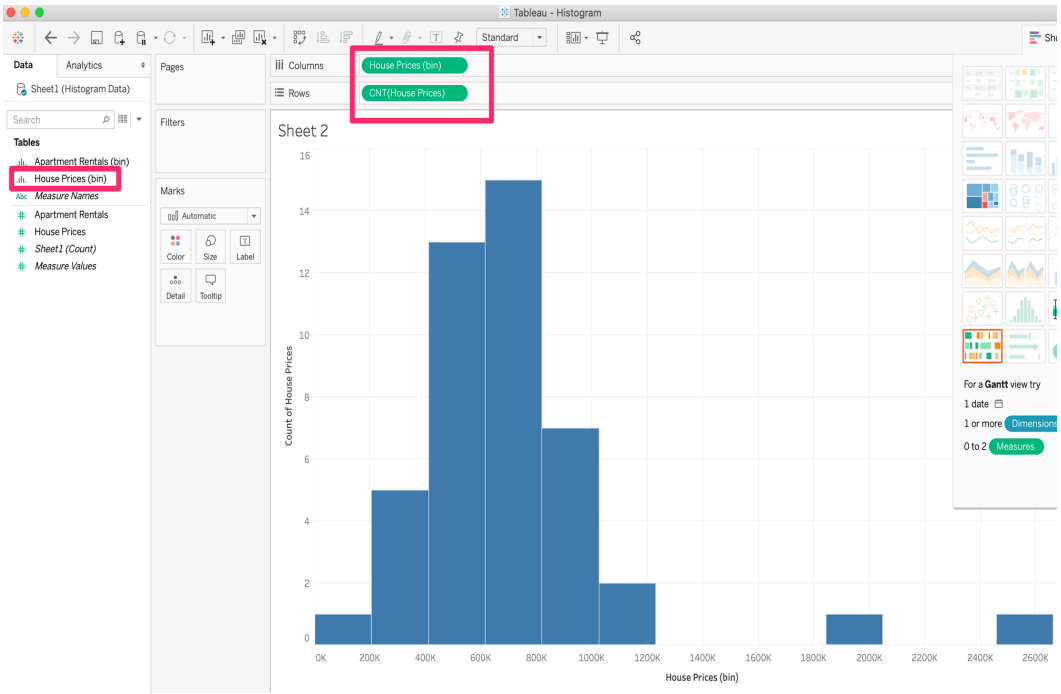
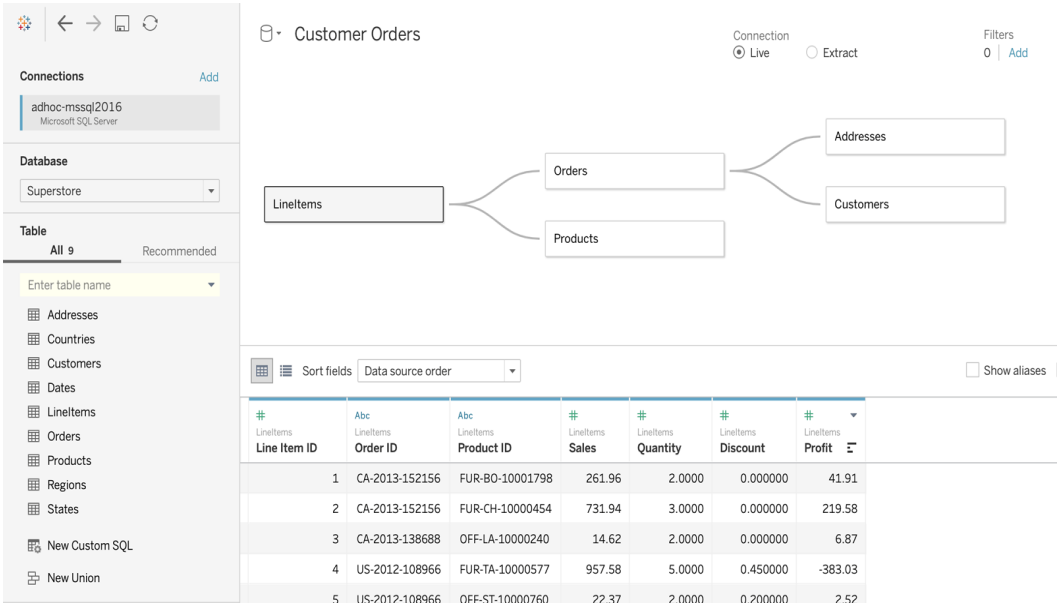


Figura 3 - Relacionamento entre Fontes de Dados no Tableau



Uma terceira ferramenta bastante interessante disponível de maneira gratuita é o Metabase, uma aplicação open-source para visualização de dados que contempla um sistema de gerenciamento de usuários juntamente com uma interface intuitiva que visa, justamente, responder às perguntas dos usuários.

No Metabase, os gráficos criados possuem nomes gerados automaticamente (com possibilidade de modificação) de modo bastante intuitivo. Além disso, a ferramenta possui um sistema de sugestão de gráficos com base nas fontes que foram disponibilizadas.

Na Figura 4, é exibida a interface de interação para geração de um gráfico no software em questão. A “pergunta simples” (Simple question) irá requisitar do programa um gráfico sugerido baseado na dimensão escolhida. Já a “pergunta customizada” (Custom question) permite mais modificações nos gráficos, incluindo manipulação por SQL.

Em sequência, a Figura 5 exibe na prática a funcionalidade de “pergunta simples”. Ao escolher a dimensão “Category”, o Metabase gerou um gráfico agrupando a quantidade de registros pela categoria correspondente. O próprio software se encarregou de analisar as informações relacionadas à categoria para gerar uma visualização que fizesse mais sentido. Além disso, ele também se encarrega de gerar automaticamente um nome intuitivo para o gráfico (na imagem, “Weekly orders by category for the past year”).

Figura 4 - Interface do Metabase

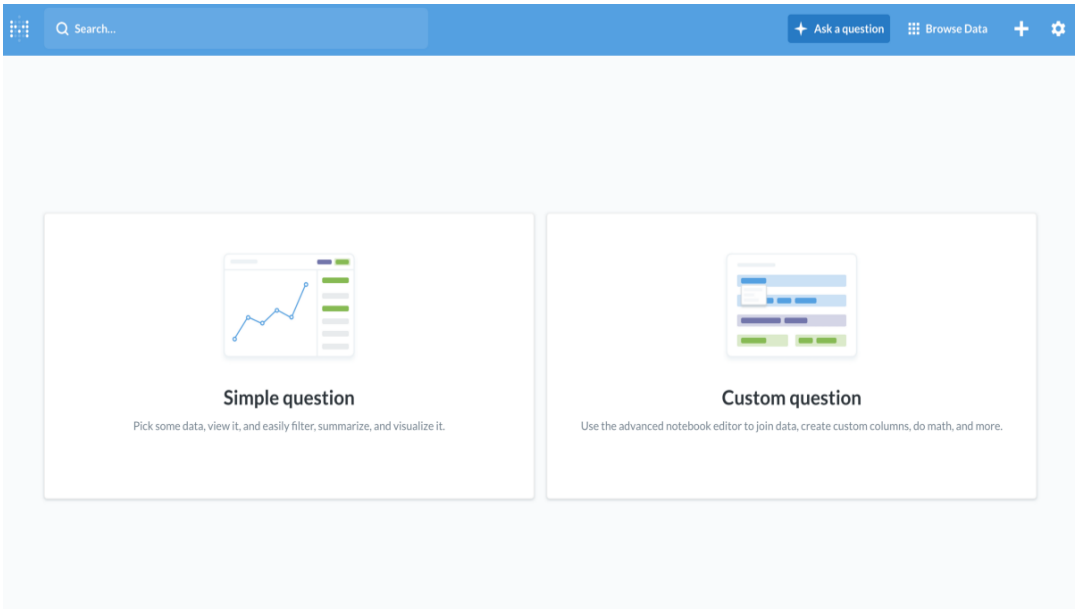
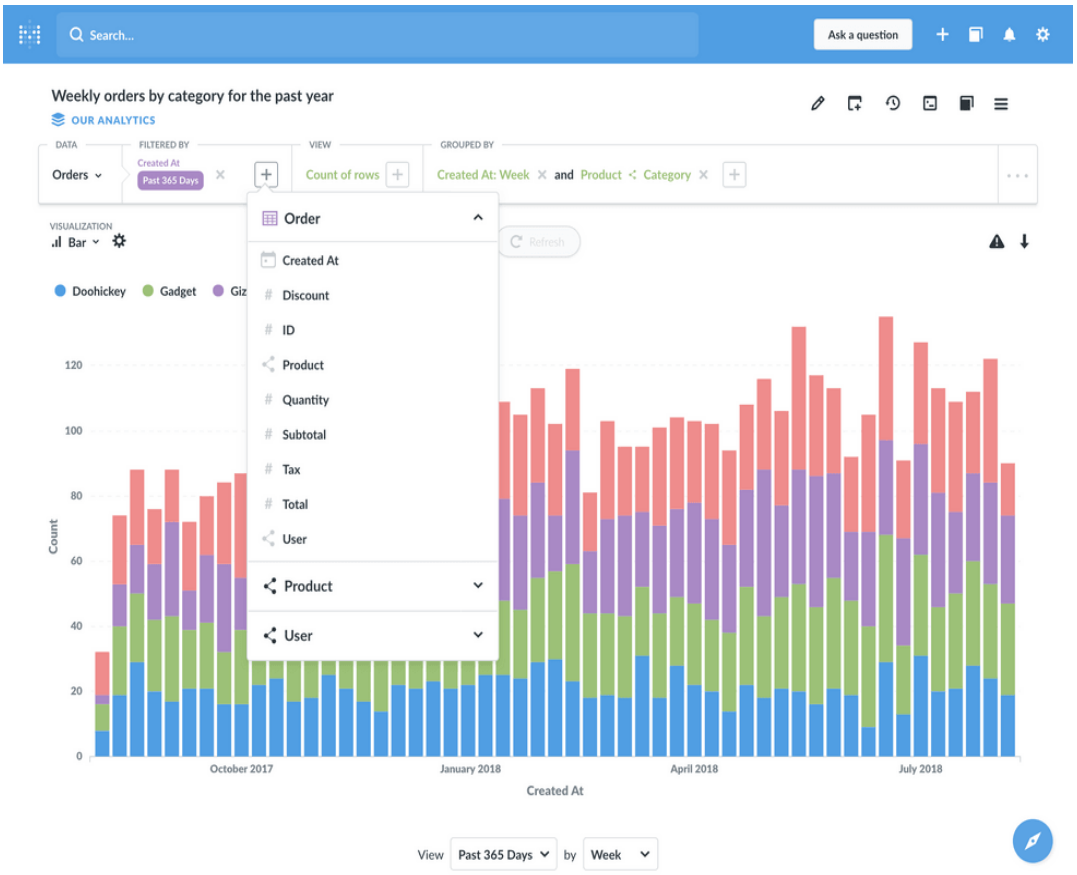


Figura 5 - Criação de gráficos no Metabase



Por fim, o principal trabalho relacionado é a primeira versão do VisMaker.

Possuindo uma interface simples e intuitiva, o VisMaker 1 é uma ferramenta de enorme potencial para a obtenção de insights analíticos sobre os dados em questão.

Atualmente, o software conta com uma usabilidade simples e semelhante ao das ferramentas mais poderosas citadas anteriormente (Power BI e Tableau).

Além disso, o VisMaker explora um ponto chave de fundamental importância: tornar a exploração de dados acessível e mais facilmente interpretável.

Dessa forma, o programa consegue fornecer diversas sugestões de correlação entre os dados que foram inseridos, juntamente com a percepção visual de quais parâmetros estão envolvidos.

Na Figura 6 é exibida a interface final do VisMaker 1, onde é possível notar semelhança principalmente com o Power BI. Neste quesito, percebe-se a influência do trabalho relacionado, utilizado de forma a complementar e viabilizar, com suas qualidades, o projeto de graduação realizado.

Ademais, a Figura 7 demonstra a principal funcionalidade da ferramenta: a orientação a perguntas. A partir da análise das dimensões, o VisMaker expande o recurso exemplificado no Metabase para uma série de insights que podem ser obtidos correlacionando as informações presentes no conjunto de dados.

Figura 6 - Interface atual do VisMaker

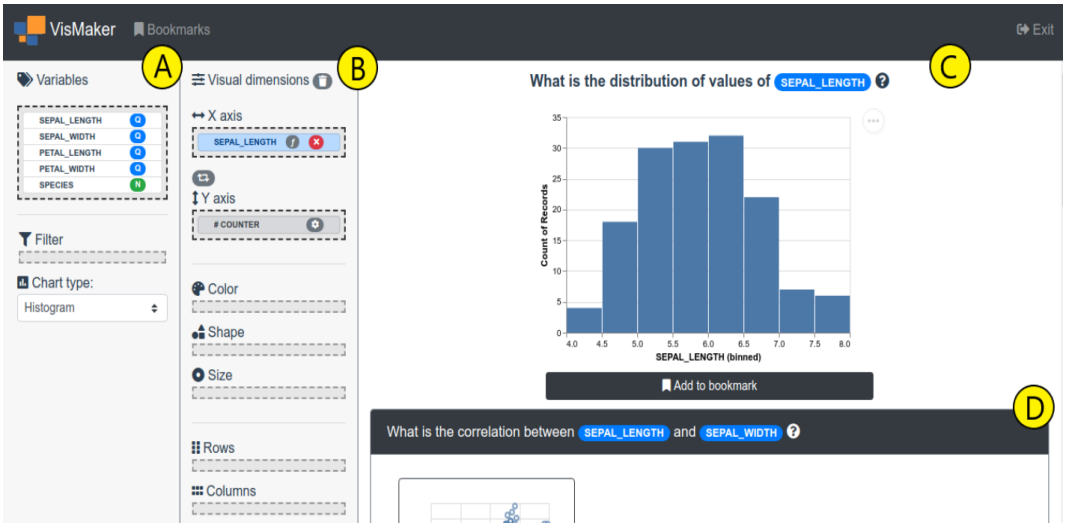
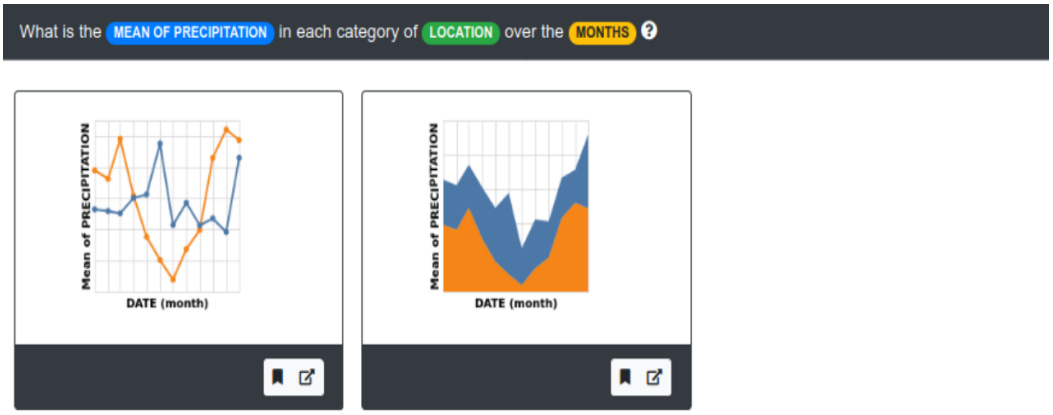


Figura 7 - Insights fornecidos pelo VisMaker



Dessa forma, pode-se perceber que o VisMaker consegue reunir de maneira simples as maiores qualidades das ferramentas anteriormente citadas, desenvolvendo-se no quesito de exploração de dados – um dos pilares que sustenta a motivação de sua criação.

3. Objetivos

Atualmente, a arquitetura e modelagem utilizadas na elaboração do VisMaker tornaram seu potencial limitado.

Visando uma melhora expressiva de desempenho e potencial da ferramenta, bem como o alongamento de sua vida útil, o principal objetivo deste Projeto Final é a remodelagem da arquitetura do VisMaker.

Com a crescente demanda por softwares, diversas práticas e convenções foram criadas nas últimas décadas para um bom gerenciamento e desenvolvimentos de aplicativos.

Dessa forma, podemos utilizar boas práticas de projetos de software para a recriação do programa, separando o Front End do Back End para gerir o fluxo de informação de maneira eficiente.

Além disso, também é possível tornar o código completamente orientado a objetos, trazendo uma modularização bastante eficiente para a aplicação e facilidades expressivas, tais como:

- Fácil detecção de problemas;
- Grande reusabilidade e fácil modificação através de herança;
- Flexibilidade de adaptação a quaisquer classes existentes;
- Facilmente extensível.

Com estas perspectivas em mente, é possível produzir um produto final com enorme capacidade de extensão de *features*, consagrando-se como uma excelente ferramenta de visualização de dados e um promissor objeto de estudos para futuras graduações em informática.

Assim, era esperado que, durante a disciplina de Projeto Final I, haja um amplo foco na modelagem do software, garantindo que toda a estrutura imaginada foi arquitetada de maneira coerente, sem a implementação de todas as funcionalidades da ferramenta.

Já no Projeto Final II, uma vez que o VisMaker esteja corretamente modelado e modularizado, serão feitas as implementações da maioria das suas funcionalidades.

4. Desenvolvimento

Dado o escopo e os objetivos previamente explicitados, a intenção para o desenvolvimento do VisMaker é de utilizar as seguintes tecnologias:

4.1. Front End

Para toda a interface a ser desenvolvida, será utilizada a linguagem Javascript, mais especificamente com o **ReactJS** [8], uma poderosa biblioteca open-source e amplamente utilizada para o desenvolvimento de aplicações web (como os sites da Netflix e Airbnb, por exemplo), sendo mantida por diversas fontes, desde programadores individuais até grandes corporações (dentre elas, o Facebook e o Instagram).

4.2. Back End

Para toda a interação, exploração e tratamento dos dados no back end será utilizada a linguagem **Python** [9], amplamente utilizada em trabalhos de análise e ciência de dados, possuindo tanto uma gama enorme de bibliotecas voltadas para esse propósito – em especial, a biblioteca Pandas [10] – quanto frameworks de imensa utilidade para a realização deste projeto – como o Flask [11], por exemplo.

4.3. Construção da Aplicação

Com o objetivo de reunir todas as estruturas previamente criadas, será utilizado o **Docker** [12], uma tecnologia de conteinerização capaz de encapsular os códigos criados e fazê-los conversar entre si, produzindo ao final um “pacote” (imagem) que pode ser executado ou mesmo modificado de maneira simples e dinâmica.

4.4. Banco de Dados

Por fim, seria criado um banco de dados – utilizando, a princípio, o MySQL [13] – para armazenar não apenas as tabelas, para que sejam facilmente consultadas a partir das requisições, mas também as demais informações estruturais da aplicação (usuários e demais configurações, por exemplo).

5. Atividades realizadas

5.1. Estudos preliminares

Ao início da disciplina de Projeto Final, o aluno não possuía conhecimento prévio em nenhuma tecnologia de Front-End, uma vez que não houve qualquer contato com as mesmas durante o período de graduação ou mesmo no estágio e no trabalho, tornando esse desenvolvimento o maior desafio do projeto em questão.

Já com relação às tecnologias de Back-End, alguns estudos foram realizados com o objetivo de relembrar conceitos das bibliotecas Pandas e Numpy, do Python. Não houve ênfase neste ponto, bem como no banco de dados, por serem tecnologias mais familiares ao escopo do aluno.

5.2. Estudos conceituais e de tecnologia

Durante a disciplina de Projeto Final foi necessário um investimento considerável de tempo para o entendimento do funcionamento de um software do ponto de vista do Front-End.

Sendo assim, foi iniciada uma jornada de aprendizado para a aquisição dos conceitos mais básicos de HTML, CSS, Javascript e ReactJS (React Nativo).

Além disso, foi necessário aprender o funcionamento de um software completo cuja comunicação ocorre frequentemente entre os servidores de Front e End, uma vez que o conhecimento prévio do aluno foi somente até a utilização de APIs já desenvolvidas.

5.3. Testes e Protótipos para aprendizado e demonstração

Para a elaboração do projeto, visto que não havia conhecimento prévio com boa parte da tecnologia utilizada e com o modelo de aplicação (full stack) que se desejava desenvolver, foram realizados alguns testes sem o comprometimento de gerar um protótipo ao final, mas sim com o único intuito de entender o funcionamento e o comportamento das tecnologias.

Esses testes foram:

- **Busca de ponta-a-ponta:**

Consistiu basicamente em testes para verificar se um simples input (POST) via form no Front End poderia ser enviado para o Back End, de modo a ser inserido no banco de dados ou mesmo ser realizado algum tipo de validação em cima do mesmo.

Essa seria a lógica utilizada para criar a autenticação de usuários dentro da aplicação.

- **Lidando com dados:**

Foram feitos alguns testes para verificar a possibilidade e o desempenho da aplicação como um todo ao lidar com o input de arquivos.

Uma vez que a ideia inicial era receber os arquivos e inseri-los no banco de dados para realizar posteriores manipulações, foi necessário explorar a biblioteca Pandas e Numpy para lidar com os dados a nível de Back End.

Foram realizados inputs de datasets contendo desde 150 registros até 578 mil, onde verificou-se que, apesar de uma lentidão perceptível entre o tempo de execução dos dois extremos, não houve erro de compilação ou impossibilidade técnica de obter o resultado esperado.

- **Respostas mais rápidas:**

A fim de tentar otimizar a passagem de informação entre as pontas, principalmente devido à natureza assíncrona percebida ao utilizar o ReactJS, foram realizados testes projetados para verificar a possibilidade de utilizar a própria resposta do Back End ao método POST do Front End, evitando assim que o último tivesse que fazer um POST de retorno para que a informação alcançasse o destino pretendido.

A partir dos testes, foi observado que havia a possibilidade de capturar o retorno da rota do Flask que recebeu o POST, e dessa forma utilizar-se desse recurso tanto para fazer validações (quanto ao sucesso ou fracasso da tentativa realizada) quanto para enviar ou receber arquivos.

- **Drag and Drop**

Para aplicar uma funcionalidade visual bastante utilizada em ferramentas de visualização de dados, foi necessário testar que tipo de eventos poderiam ser capturados e utilizados ao adicionar uma biblioteca de arrastar e soltar componentes no Front End.

Essa tarefa foi relativamente complicada pois as principais bibliotecas que implementam essa funcionalidade possuem uma documentação bastante inacessível (no sentido de complexa) para aqueles que estão iniciando com o Javascript e ReactJS. É necessária uma constante manipulação de arrays e muitas vezes os exemplos compõem uma quantidade grande de componentes complexos.

- **Criar gráficos**

Foram testadas algumas bibliotecas gráficas do ReactJS para entender seu funcionamento, sua extensão (ou seja, a quantidade de recursos gráficos oferecidos) e a complexidade de sua manipulação.

5.4. Método

O método utilizado para o desenvolvimento da aplicação foi uma variação do TDD (Test Driven Development) [14], uma metodologia voltada para o desenvolvimento de software com base em testes.

No TDD, são executados ciclos de repetição onde cada funcionalidade é desenvolvida a partir da falha do teste realizado para tentar executar a função requisitada.

Neste projeto final, foi seguido o mesmo raciocínio: seriam aplicados testes a uma determinada funcionalidade, e enquanto ela não fosse de fato implementada com sucesso, seriam feitos ciclos de repetição para refatorar o módulo em questão.

Essa abordagem trouxe tanto pontos positivos, como a garantia de funcionamento da funcionalidade trabalhada, quanto um ponto bastante negativo: o tempo necessário para avançar, visto que, devido ao nível técnico iniciante do aluno com relação às tecnologias de Front End, os testes e os ciclos demoravam muito mais do que o previsto para serem concluídos.

Dessa forma, a metodologia de execução do projeto pode ser resumida em:

- Passo 1: Implementar tela de login;
- Passo 2: Implementar tela de envio de arquivo;
- Passo 3: Implementar tela da aplicação.

5.5. Plano de Ação

A partir da metodologia explicitada acima, os cronogramas originais elaborados para as disciplinas de Projeto Final I e Projeto Final II podem ser vistas abaixo:

Tabela 1 - Cronograma do Projeto Final I

Atividades de Projeto Final I	Março				Abril				Maio				Junho				Julho			
Proposta																				
Estudar a modelagem																				
Criar a modelagem																				
Revisão da Modelagem																				
Migrar a interface																				
Testar a interface																				
Criação do back end																				
Testar o back end																				
Conteinerizar a aplicação																				
Implementar funcionalidades básicas																				
Revisar aplicação e resolver pendências																				

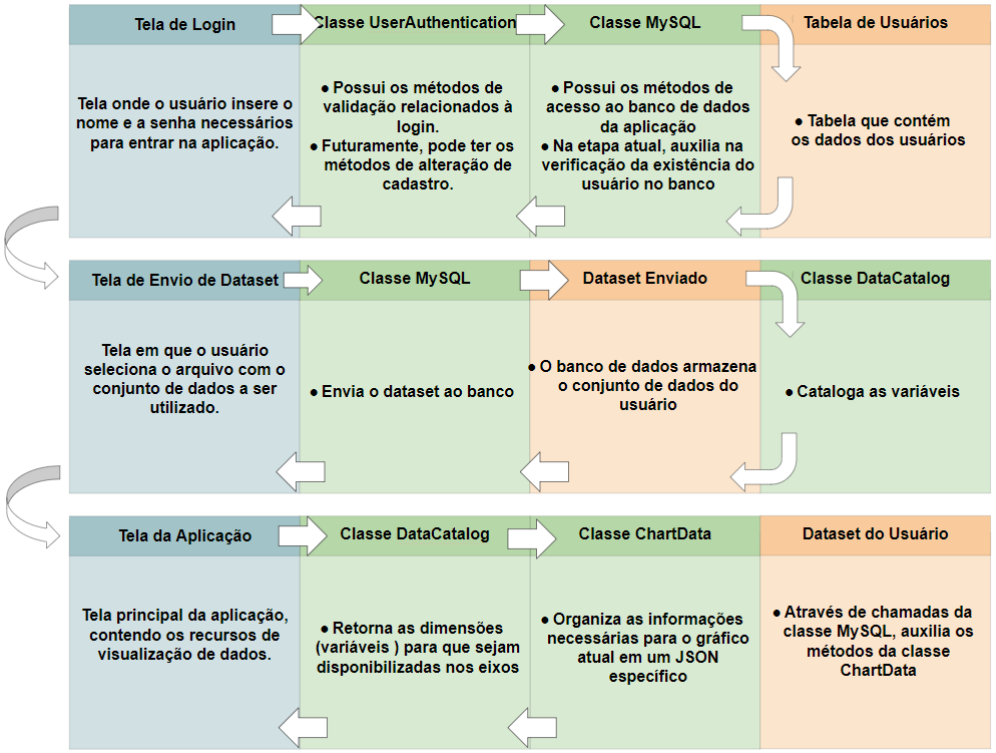
Tabela 2 - Cronograma do Projeto Final I

Atividades de Projeto Final II	Agosto				Setembro				Outubro				Novembro				Dezembro			
Implementar as demais funcionalidades																				
Testar as funcionalidaes																				
Testes de exaustão																				
Revisão de código																				
Aplicação de revisões (se necessário)																				
Modelagem do banco de dados																				
Criação do banco de dados																				
Implementação do banco na aplicação																				
Implementação de funcionalidades relacionadas ao banco																				
Revisão de código																				
Montagem e ensaio da Apresentação Final																				

6. Projeto e especificação do sistema

A fim de montar uma exemplificação visual de funcionamento, foi elaborado um fluxograma para exemplificar a estrutura pensada para a aplicação até o momento.

Figura 8 - Fluxograma da aplicação



O esquema de cores da figura 4.1 é configurado por:

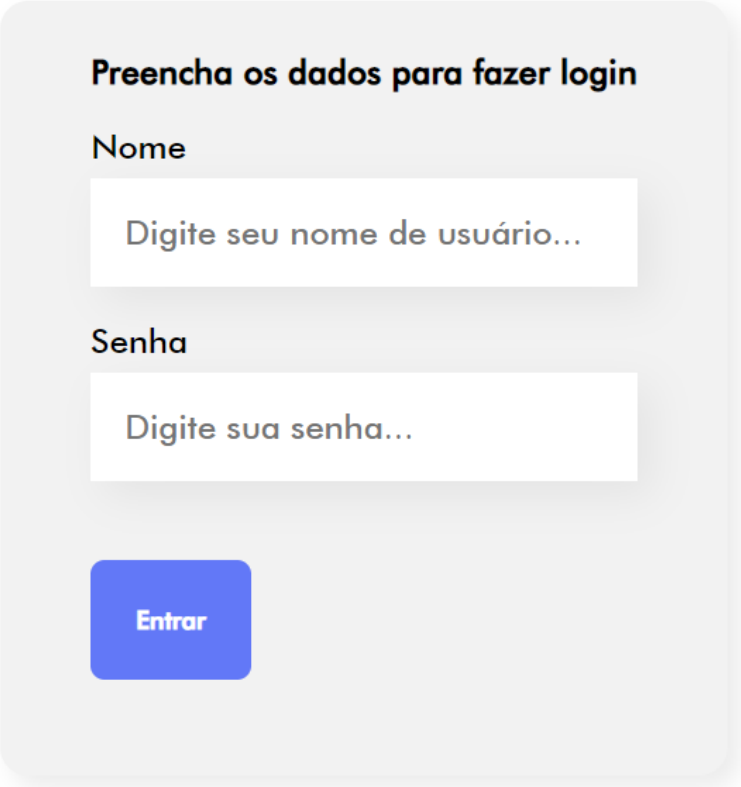
- Azul: front-end
- Verde: back-end
- Laranja: banco de dados

As etapas serão detalhadas a seguir:

6.1. Tela de Login

A primeira etapa de implementação a ser realizada, em conformidade com a metodologia escolhida, seria a tela de login. A versão final desta tela pode ser visualizada na Figura 9.

Figura 9 - Tela de Login



A primeira tela da aplicação possui apenas um formulário que solicita ao usuário o seu nome e senha para que seja efetuado o login.

Por conta da autenticação não ter sido o foco do projeto, foi criada uma tabela users dentro do banco de dados. Essa criação é feita sempre que o servidor do Back End é levantado, através de um método dentro da classe **MySQL**. Através da biblioteca SQLAlchemy [15], é feita a conexão com o banco e é executada uma query simples de “CREATE TABLE users IF NOT EXISTS” e

“INSERT INTO TABLE VALUES”, como documentado na referência da linguagem SQL [16], em seguida inserindo os valores de usuário padrão (usuário “admin” com senha “admin”).

Quando clicado em enviar, é feito um POST ao Back End para que ele requisite o banco de dados, verificando se a correspondência de usuário e senha existe. Se existir, o usuário é redirecionado para outra tela a partir da utilização do React Router [17], um componente do React capaz adicionar à aplicação de página única (SPA) a possibilidade de transitar entre rotas, gerando a possibilidade de criar diversas páginas. Caso não haja correspondência, o formulário é limpo e nada acontece.

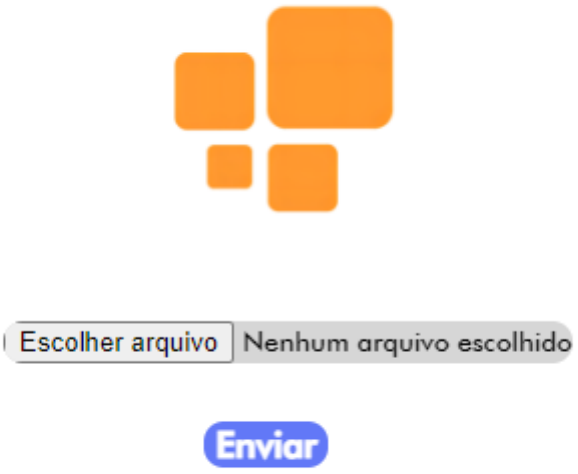
6.2. Tela de Envio do Arquivo

A segunda etapa da aplicação é o envio de um conjunto de dados a partir de um arquivo .CSV.

Esta etapa é bem direta, necessitando apenas que o usuário escolha o arquivo e clique e enviar.

Vale ressaltar que não é verificado aqui se o tipo de arquivo enviado é, de fato, um .CSV.

Figura 10 - Tela de Envio do Arquivo



Novamente, ao clicar em enviar, o conjunto de dados do usuário é enviado para o Back End, onde é transformado em um Dataframe e, em seguida, enviado para o banco de dados com a utilização da classe MySQL. O React Router entra novamente em ação e direciona o usuário, finalmente, à página da aplicação.

6.3. Tela da Aplicação

Figura 11 - Tela de Login

A etapa final consiste na tela em que o usuário consegue interagir com a aplicação, através do componente React Beautiful DnD [18]. A partir da interação de arrastar e soltar, o usuário consegue escolher as variáveis para os eixos x e y.

Nessa etapa é feita uma primeira requisição ao Back End para que sejam retornadas as dimensões do dataset do usuário e, assim, a lista de variáveis seja preenchida.

Houveram alguns bugs de implementação, como será comentado na próxima sessão.

Além disso, para a parte gráfica, inicialmente foi utilizado o componente CavasJS [19].

O componente anterior possuía todos os tipos de gráficos necessários, bem como uma configuração bastante amigável e intuitiva do output gerado. A lógica implementada foi de que, sempre que duas variáveis fossem adicionadas (ou seja, os eixos x e y estivessem preenchidos), uma requisição seria enviada

ao Back End, onde a classe ChartData montaria o JSON específico para a montagem daquele gráfico.

Porém, devido à uma limitação encontrada no componente acima, foi necessário fazer uma mudança para o componente Recharts [20], que por sua vez resultou em um processo completamente diferente de implementação trazendo consigo outras limitações, impedindo assim que a funcionalidade de orientação à perguntas fosse implementada no VisMaker.

7. Implementação e Avaliação

A partir dos testes iniciais, foram-se obtendo resultados satisfatórios nas primeiras duas telas.

Porém, uma grande dificuldade encontrada na terceira tela foi a naturalidade com a qual processos assíncronos são implementados no ponto de vista de uma aplicação Front End. Essa ocorrência foi de especial dificuldade para a implementação e entendimento de um desenvolvedor que só havia experienciado desenvolvimentos de Back End até o momento, o que resultou em alguns bugs principalmente na funcionalidade de arrastar e soltar.

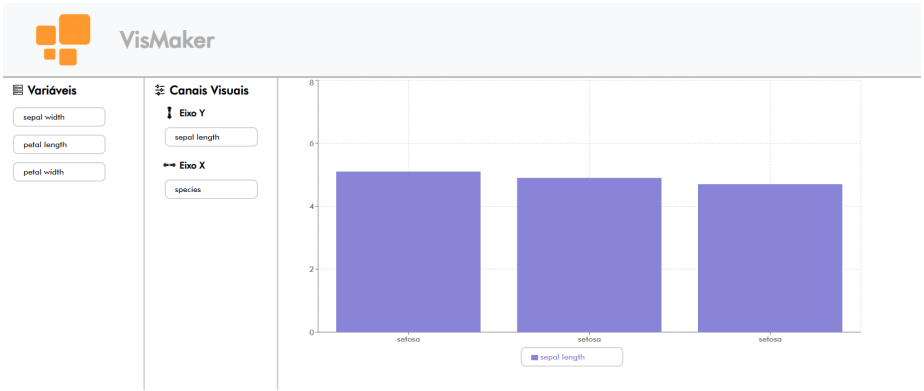
Além disso, uma triste ocorrência acabou com que a aplicação não cumprisse seu propósito final.

Através do CanvasJS, foram criados vários gráficos e experienciados vários testes com um dataset público de casos de covid na União Europeia [21]. Nesse dataset, tanto o eixo x quanto o eixo y eram variáveis numéricas (quantitativas).

Para minha surpresa, ao tentar criar gráficos com uma variável de texto, o gráfico não me retornava resultado. A partir de vários testes, deduzi que esse é o comportamento desse componente.

Esse foi um grande problema, pois não consegui encontrar outra biblioteca que suprisse as necessidades e se encaixasse na lógica que foi implementada antes, fazendo com que, ao trocar para o componente Recharts, a funcionalidade de orientação à perguntas fosse pardida e o VisMaker cumprisse apenas o papel de uma simples ferramenta de visualização de dados.

Figura 12 - Funcionamento atual do VisMaker



Por fim, também não foi entregue ao final deste projeto o VisMaker em uma versão conteinerizada do Docker.

Isso porque o grande objetivo de trazer esse recurso para o VisMaker seria sua implementação na nuvem, através do Cloud Run [22] dentro do Google Cloud Platform [23], trazendo assim a aplicação em um ambiente o mais próximo possível a um cenário de Produção.

Infelizmente, após diversos testes iniciais, não consegui produzir uma URL pública e acessível de dentro da nuvem para que a aplicação pudesse ser utilizada dentro desse ambiente.

8. Conclusão e Considerações Finais

A escolha do tema me deixou bastante animado para trabalhar no projeto. Hoje já atuo na área de dados e sempre tive o interesse de começar a desenvolver aplicações full stack. Por conta disso, vi essa como uma oportunidade perfeita.

Senti bastante dificuldade com o aprendizado de toda a carga de conteúdo necessária para o desenvolvimento da aplicação, mas ao longo do caminho fui percebendo que as tarefas foram ficando mais intuitivas e que o aprendizado estava, de fato, persistindo.

Infelizmente, por um descuido em relação à biblioteca utilizada, uma boa parte do desenvolvimento foi perdido e o VisMaker acabou por não atingir a sua funcionalidade máxima. Porém, fiquei bastante satisfeito com todo o projeto que consegui montar, das funcionalidades que consegui implementar e com todo o imenso aprendizado que tive ao longo deste projeto final.

Se tivesse a oportunidade de começar agora, focaria mais em desenvolver por conta própria projetos de front end para que mais dúvidas fossem geradas e, consequentemente, fossem sanadas mais rapidamente. Fiquei meio perdido no início, e tentei associar muito do conhecimento a partir de projetos já criados. Por conta disso, acabei exercitando menos a mente e tive mais dificuldades ao longo do caminho. Outro ponto importante seria a utilização de testes mais completos para verificação do cumprimento de requisitos pelas bibliotecas, pois através do acontecido neste projeto, uma grande parte do trabalho acabou por ser perdido.

Acredito que o VisMaker, apesar de não ter trazido sua principal funcionalidade nesse projeto, encontrou um caminho de desenvolvimento que possibilita a sua expansão e uma grande gama de aprendizado para os próximos que tiverem contato com o tema.

Continuarei trabalhando nele e, graças a esse projeto, me sinto pronto para iniciar outros desenvolvimentos.

9. Referências bibliográficas

[1] VARIAN, Hal. **Data Storytelling: The Essential Data Science Skill Everyone Needs**. Disponível em: <https://www.forbes.com/sites/brentdykes/2016/03/31/data-storytelling-the-essential-data-science-skill-everyone-needs/?sh=5687b52a52ad>. Acesso em: 06 de abril de 2022.

[2] REICHENBACH, Hans. **Experience and Prediction: An Analysis of the Foundations and the Structure of Knowledge**. Chicago, IL, USA: University of Chicago Press (1938). DOI: 10.2307/2180454.

[3] WANG, Peter. **Why Data Visualization is One of the Hardest but Most Important Tasks**. Anaconda, 2021. Disponível em: <https://www.anaconda.com/blog/why-data-visualization-is-one-of-the-hardest-but-most-important-tasks>. Acesso em: 30 de junho de 2022.

[4] BARBOSA, Simone Diniz Junqueira; LIMA, Raul de Araújo. **VisMaker: a Question-Oriented Visualization Recommender System for Data Exploration**. Disponível em: <https://arxiv.org/pdf/2002.06125.pdf>. Acesso em: 06 de abril de 2022.

[5] **Power BI**. Disponível em: <https://powerbi.microsoft.com/pt-br/>. Acesso em: 01 de julho de 2022.

[6] **Tableau**. Disponível em: <https://www.tableau.com/pt-br>. Acesso em: 01 de julho de 2022.

[7] **Metabase**. Disponível em: <https://www.metabase.com/>. Acesso em: 01 de julho de 2022.

[8] **ReactJS**. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 01 de julho de 2022.

[9] **Python**. Disponível em: <https://www.python.org/>. Acesso em: 01 de julho de 2022.

[10] **Pandas**. Disponível em: <https://pandas.pydata.org/docs/>. Acesso em: 01 de julho de 2022.

[11] **Flask**. Disponível em: <https://flask.palletsprojects.com/en/2.1.x/>. Acesso em: 01 de julho de 2022.

[12] **Docker**. Disponível em: <https://www.docker.com/>. Acesso em: 01 de julho de 2022.

[13] **MySQL**. Disponível em: <https://www.mysql.com/>. Acesso em: 01 de julho de 2022.

[14] **Test-driven development**. Disponível em: <https://www.devmedia.com.br/test-driven-development-tdd-simples-e-pratico/18533>. Acesso em: 19 de dezembro de 2022.

[15] **SQLAlchemy**. Disponível em: <https://www.sqlalchemy.org/>. Acesso em: 19 de dezembro de 2022.

[16] **Linguagem SQL**. Disponível em: <https://www.w3schools.com/sql/default.asp>. Acesso em: 19 de dezembro de 2022.

[17] **React Router**. Disponível em <https://reactrouter.com/en/main>. Acesso em: 19 de dezembro de 2022.

[18] React Beautiful Dnd. Disponível em: <https://github.com/atlassian/react-beautiful-dnd>. Acesso em: 19 de dezembro de 2022.

[19] **CanvasJS**. Disponível em: <https://canvasjs.com/docs/charts/integration/react/>. Acesso em: 19 de dezembro de 2022.

[20] **Recharts**. Disponível em: <https://recharts.org/>. Acesso em: 19 de dezembro de 2022.

[21] Data on COVID-19 vaccination in the EU/EEA. Disponível em: <https://www.ecdc.europa.eu/en/publications-data/data-covid-19-vaccination-eu-eea>. Acesso em: 19 de dezembro de 2022.

[22] **Cloud Run**. Disponivel em: <https://cloud.google.com/run/docs>. Acesso em: 19 de dezembro de 2022.

[23] **Google Cloud Platform**. Disponivel em: <https://cloud.google.com/docs>. Acesso em: 19 de dezembro de 2022.

10. Demais referências

Link para o Github:
https://github.com/bispomat/VisMaker.git