Aplicação web para coleta e visualização de dados de repositórios do GitHub

Yuri S. L. Shiino[1], **Hudson S. Borges**[1]

[1]Universidade Federal do Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brasil

{yuri.seiti, hudson.borges}@ufms.br

Abstract. *GitHub, the largest collaborative development platform, provides a number of APIs for building scripts and applications that integrate with or extend its functionality. Some of these APIs provide data about repositories and the users who have favorited them, in the form of "stars,"which serve as a measure of the community's interest in the project. There are sites that show the evolution of a repository's star count over time, but they have some limitations. The main one is that they work with a sample of the data, due to the limitations of GitHub's REST APIs, and also the fact that they don't return any data beyond the forty thousandth star, which affects the accuracy of the analysis to be made. With this in mind, it was proposed to develop an application using GitHub's GraphQL APIs as an alternative that does not have these limitations, allowing for a complete and accurate assessment of the number of stars in a given repository over time, enabling researchers and enthusiasts to gain different insights into their communities.*

Resumo. *O GitHub, a maior plataforma de desenvolvimento colaborativo, disponibiliza um conjunto de APIs para a criação de scripts e aplicações que integram ou ampliam suas funcionalidades. Algumas dessas APIs fornecem dados sobre os repositórios e os usuários que o favoritaram, por meio das "Estrelas", que serve como uma métrica de interesse da comunidade no projeto. Existem sites que mostram a evolução da quantidade de estrelas de um repositório ao longo do tempo, porém com algumas limitações. A principal delas é trabalhar com uma amostragem dos dados, devido a restrições das APIs REST do GitHub, e também o fato de que não retornam dados além da estrela de número quarenta mil, afetando a precisão das análises a serem feitas. Com isso em mente, foi proposto o desenvolvimento de uma aplicação que utilize as APIs GraphQL do GitHub como uma alternativa que não apresenta essas limitações, permitindo uma avaliação completa e precisa do número de estrelas de um determinado repositório ao longo do tempo, possibilitando que pesquisadores e entusiastas obtenham diferentes insights sobre suas comunidades.*

1. Introdução

O GitHub se destaca como a principal plataforma de desenvolvimento colaborativo do mundo e desempenha um papel fundamental no desenvolvimento de software moderno. Para se ter uma ideia, cerca de 100 milhões de desenvolvedores utilizam a plataforma [1], recebendo mais de 500 milhões de visitas mensais [2]. Essa plataforma, amplamente adotada por desenvolvedores e pesquisadores, oferece não só recursos de controle de versão,

mas também funcionalidades sociais que permitem a colaboração, a comunicação e a interação entre a comunidade.

Por exemplo, uma das funcionalidades que é frequentemente utilizada como métrica de popularidade e engajamento no GitHub são as "Estrelas", que representam a quantidade de usuários que marcaram um determinado projeto como favorito. As estrelas servem como um indicador do interesse e do reconhecimento da comunidade em relação a um repositório, retornando um indício sobre a adoção, impacto e o alcance do projeto, como visto em [3].

Esses dados, que refletem o engajamento da comunidade, são essenciais para análises mais profundas sobre o impacto e a adoção de projetos no GitHub. Entretanto, buscar dados sobre os repositórios para realizar análises mais profundas é uma tarefa difícil. O GitHub fornece um conjunto de API para disponibilizar esses dados para pesquisadores e entusiastas. Contudo, elas possuem algumas limitações que afetam as soluções existentes.

Neste artigo, é proposto o desenvolvimento de uma aplicação web que utilize a API GraphQL do GitHub, podendo fornecer uma forma de coleta de dados que permita uma análise mais precisa e facilitada da evolução do número de estrelas de repositórios ao longo do tempo. Essa abordagem contorna as limitações das soluções existentes, fazendo que pesquisadores, desenvolvedores e entusiastas obtenham insights valiosos sobre sua comunidade.

Este artigo está estruturado da seguinte forma: a Seção 2 discute as limitações das duas APIs REST e GraphQL do GitHub e apresenta os desafios na coleta de dados. A Seção 3 apresenta a proposta de desenvolver uma ferramenta online de coleta de dados do GitHub. A Seção 4 explica as tecnologias utilizadas para o desenvolvimento desta aplicação. A Seção 5 apresenta a aplicação desenvolvida e suas funcionalidades. A Seção 6 apresenta a avaliação de desempenho da aplicação e discute os principais resultados. A Seção 7 conclui o artigo e apresenta trabalhos futuros.

2. GitHub e os desafios ao coletar seus dados

Existem duas principais formas de ter acesso aos dados do GitHub: a API REST [4] e a API GraphQL [5]. A primeira é considerada mais simples de ser utilizada, visto que muitos desenvolvedores tem familiaridade com esse tipo de API. Ela faz uso de protocolos HTTP, sendo que muitas consultas utilizam o método GET, permitindo o uso eficiente de cache nas aplicações. A segunda tem uma curva de aprendizado maior, porém trás novas possibilidades na busca dos dados. Ela possui maior flexibilidade nas consultas e permite diferentes dados com uma única consulta, porém possui um tempo de resposta maior.

A API REST do GitHub possui uma limitação significativa para a análise de repositórios populares: ela não retorna dados além da 40.000ª estrela de um repositório. Essa limitação tem grande impacto na análise de repositórios populares, impedindo que se tenha uma visão geral da evolução do número de estrelas.

Algumas das ferramenta populares que são afetadas por essa limitação é o site star-history.com [6] e o site starchart.cc [7], que recuperam amostragens até a estrela de número 40.000, e a próxima limitação é a quantidade atual de estrelas, afetando a qualidade de possíveis análises. Essa restrição faz com que repositórios altamente populares

percam informações de grandes períodos, impossibilitando a visualização de mudanças de popularidade ou tendências específicas que podem ter ocorrido após esse limite ser atingido. Isso acaba afetando a qualidade de estudos comparativos e pesquisas acadêmicas que dependem de análises completas.

A API GraphQL não possui esse tipo de limitação. Entretanto, desenvolver ferramentas eficientes que fazem uso dessa API não é uma tarefa simples, visto que tem uma curva de aprendizado mais difícil, devido uso de queries complexas para realizar as buscas.

3. Construindo uma ferramenta online de mineração

Trabalhos acadêmicos que utilizam dados do GitHub tipicamente são baseados em *scripts* ou procedimentos que devem ser executados localmente para realizar a coleta dos dados através das APIs. Embora seja funcional, isso acaba dificultando o acesso à ferramenta, visto que a instalação local é dependente do sistema operacional utilizado e configurações específicas de cada ambiente de trabalho.

A proposta deste trabalho consiste em avaliar a viabilidade de desenvolver uma ferramenta web que seja capaz de realizar a coleta, armazenamento e visualização dos dados diretamente no navegador. Isso permite que os usuários possam ter visões completas e precisas sem a necessidade de instalar e configurar um software específico para isso em seu computador.

Existem alguns desafios para a construção de uma ferramenta web com essa finalidade, sendo o maior deles a coleta, armazenamento, manipulação e visualização de um grande volume de dados diretamente no navegador. Por conta disso, nossa proposta foca na coleta e visualização de dados de estrelas dos repositórios. A partir disso, a ferramenta poderá ser expandida para diferentes visualizações e métricas, visto que a API GraphQL do GitHub fornecem inúmeras outras informações pertinentes aos repositórios.

Para isso, foi utilizada a biblioteca Gittrends-app, desenvolvida por Hudson Borges, que permite coletar os dados sobre um repositório e os usuários que o marcaram com uma estrela (*stargazers*) de forma sequencial, fazendo uso da API GraphQL do GitHub. Essa API não possui limitações que impactariam a qualidade da análise, e permite utilizar um sistema de cache para otimizar o consumo desta API. Esta biblioteca está disponível em npm [8].

4. Tecnologias utilizadas

4.1. Gittrends-app

Gittrends-app [8] é uma solução desenvolvida para a coleta dos dados dos repositórios através da API GraphQL. Com ela, é possível buscar informações sobre um repositório específico, como dados de *releases*, *tags*, *issues*, *pull requests*, entre outros, utilizando o nome do autor e o nome do repositório, e a partir do ID do repositório obtido, fazer buscas sequenciais para obter informações sobre todas as estrelas dadas neste repositório.

A Gittrends-app também possui suporte a um sistema de cache, compatível com diversos tipos de banco de dados. Para fazer uso desse sistema, foi criado uma abstração que verifica se já existem dados salvos de um determinado repositório antes de fazer as

requisições para a API do GitHub, evitando assim fazer chamadas desnecessárias, visto que a API tem possui um limite de 5.000 requisições por hora para usuários comuns.

4.2. TypeScript

TypeScript [9] é linguagem de programação *open source* desenvolvida pela Microsoft que funciona como uma extensão do JavaScript. A linguagem adiciona recursos como tipagem estática, melhora o uso de conceitos de programação orientada a objetos, como encapsulamento e herança, e possui uma melhor integração com as IDEs, sendo capaz de fornecer um feedback imediato sobre erros e melhorando a experiência de desenvolvimento.

Todos esses recursos fazem com que o ambiente de desenvolvimento seja mais seguro e controlado. A tipagem estática de variáveis, parâmetros e retornos de funções, juntamente com a análise sintática e semântica do TypeScript, faz com que seja minimizado a quantidade de erros durante o desenvolvimento, além de aumentar a produtividade e a facilidade de dar manutenção no projeto.

O processo de transpilação, realizado pelo compilador do TypeScript, transforma o código fonte em JavaScript puro, mantendo total compatibilidade com os ambientes de execução existentes, permitindo que o TypeScript seja implementado aos poucos em projetos já em andamento. Durante este processo, são realizadas verificações extensivas que garantem a integridade dos tipos e a correta implementação das interfaces definidas, contribuindo para a robustez do sistema final.

4.3. React

React [10] é uma biblioteca *open source* de JavaScript, criada e mantida pelo Facebook (Meta) em 2013. A motivação por trás de seu desenvolvimento envolve a complexidade da manipulação do DOM (Document Object Model), a falta de modularidade, grande quantidade de reutilização de código, dificuldade no gerenciamento de estados e baixa performance de renderização.

O React implementa uma representação virtual do DOM e utiliza um algoritmo de reconciliação para minimizar a quantidade de operações e fazer atualizações em lote, reduzindo assim de forma significativa a manipulação do DOM real, melhorando a performance da aplicação como um todo.

Além disso, permite a criação de componentes reutilizáveis e estabelece um fluxo de dados unidirecional (top-down), estabelecendo uma hierarquia clara de componentes, que acaba facilitando o processo de desenvolvimento, manutenção e escalabilidade do código, bem como o gerenciamento de estados e melhorando a experiência ao debugar.

O React foi especialmente útil no desenvolvimento desta aplicação, visto que permite re-renderizar somente os componentes que sofreram alteração, permitindo a visualização dos dados em tempo real enquanto são buscados pela API, tudo isso sem prejudicar a performance do restante da página.

4.4. Express

Express [11] é framework *open source* que tem como objetivo simplificar o processo de criação de endpoints, fornecendo uma camada de abstração que permite a criação de

APIs de um modo rápido e escalável. Com ele, é possível definir rotas para diferentes endpoints, e cada rota pode estar associada a um ou mais middlewares, que são executados ao acessar tal rota.

Os middlewares são funções que recebem até quatro parâmetros, o objeto da requisição, o objeto da resposta, que são ambos obrigatórios, um objeto de erro e uma função de callback, ambos opcionais, que chama outro middleware. Com isso, é possível criar endpoints com uma grande variedade de funcionalidades.

Além disso, Express possui recursos como gerenciamento de sessão, tratamento de erros e suporte a requisições assíncronas.

Para a aplicação proposta, o Express é usado especificamente para a autenticação do usuário, visto que precisamos obter um *token* de acesso através do protocolo OAuth, este que só funciona através de um servidor proxy. Todas as outras requisições são feitas diretamente do lado do cliente, utilizando o *token* obtido para poder realizar as chamadas para a API do GitHub.

4.5. PouchDB

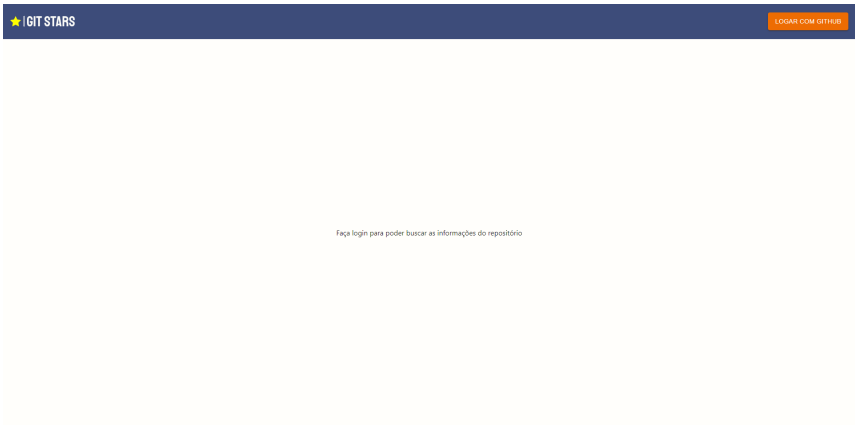
PouchDB [12] é um banco de dados que funciona diretamente no navegador. Ele faz uso da API IndexedDB, fornecida pelos navegadores, que permite armazenar dados localmente no computador do usuário.

O PouchDB armazena os dados em formato JSON, salvando cada objeto como um documento independente, atribuindo um ID único e dispensando a necessidade de ter um esquema definindo os campos e seus tipos, fazendo com que seja uma boa opção de armazenamento de dados persistente.

Na aplicação desenvolvida, o PouchDB é utilizado para o sistema de cache, no qual as informações coletadas ficam armazenadas no IndexedDB e são válidas pelo período de uma semana, otimizando assim o uso da API do GitHub.

5. Git Stars: Uma ferramenta de coleta de dados e visualização de popularidade

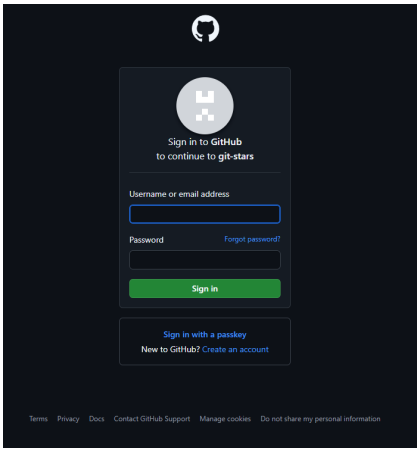
O Git Stars [13] é a aplicação web desenvolvida para fornecer visualizações detalhadas da evolução das estrelas em repositórios do GitHub. Ao entrar no site, o usuário é recebido com a tela inicial da aplicação, informando que o usuário precisa estar logado para utilizar a ferramenta. A Figura 1 apresenta a tela inicial da aplicação.



Fonte: Próprio Autor

Figura 1. Tela inicial antes do login

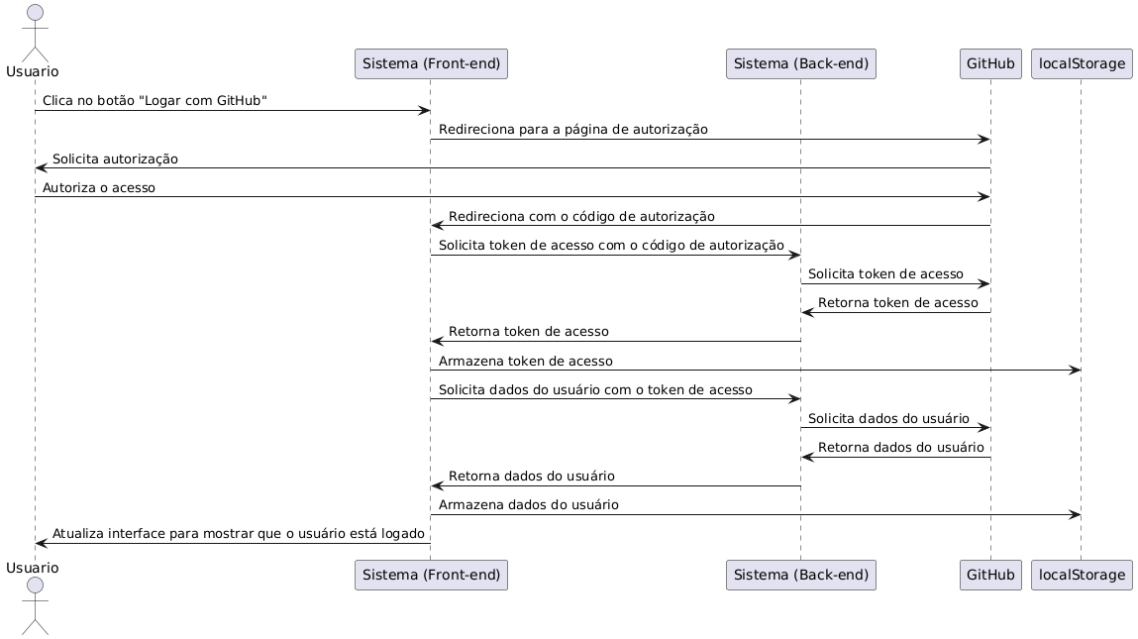
Ao clicar no botão “LOGAR COM GITHUB”, no canto superior direito, o usuário é redirecionado para a tela de autenticação do GitHub, onde digita seu login e senha (Ilustrado na Figura 2). Assim, é enviado um código para o navegador, que utilizamos para obter acesso ao *token* de acesso deste usuário. Este token é armazenado no *local storage* do navegador, permitindo fazer chamadas para as APIs do GitHub, podendo assim fazer a visualização de dados de repositórios.



Fonte: Próprio Autor

Figura 2. Tela de autenticação do GitHub

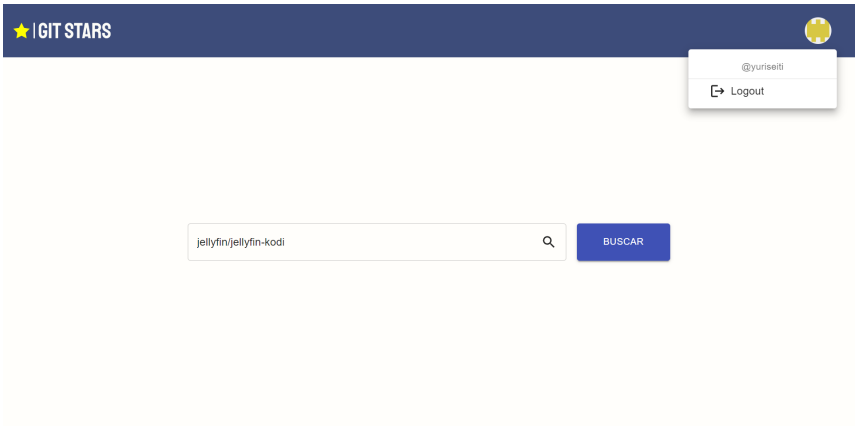
Na Figura 3 observa-se o fluxo de autênticação. Ele ilustra como o sistema gerencia o processo de login do usuário por meio da API do GitHub.



Fonte: Próprio Autor

Figura 3. Diagrama de sequência do fluxo de autenticação

Após fazer login, o usuário é redirecionado para a tela inicial da aplicação, onde é recebido com uma barra de pesquisa já com um repositório preenchido, para que sirva de exemplo de como deve ser o formato inserido para realizar uma busca. No canto superior direito, é exibido a foto do usuário, que ao ser clicada exibe um menu com o *username* e um botão para deslogar, exibido na Figura 4.

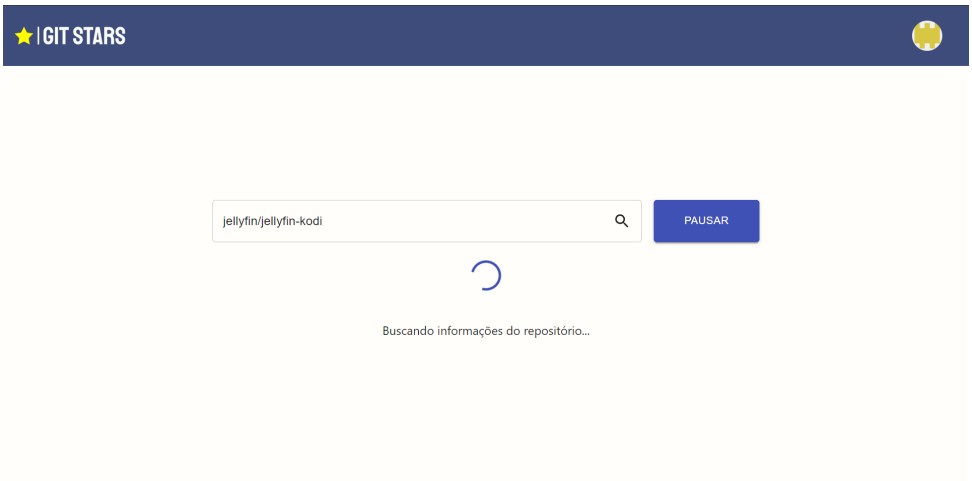


Fonte: Próprio Autor

Figura 4. Tela inicial com usuário logado

Ao clicar em ”Buscar”, é verificado se existem dados em cache para aquele repositório. Para serem considerados válidos, podem ter sido criados à no máximo uma semana. Caso existam e sejam válidos, os dados são utilizados. Caso contrário, é feita uma chamada para a API do GitHub para buscar informações do repositório e é exibido

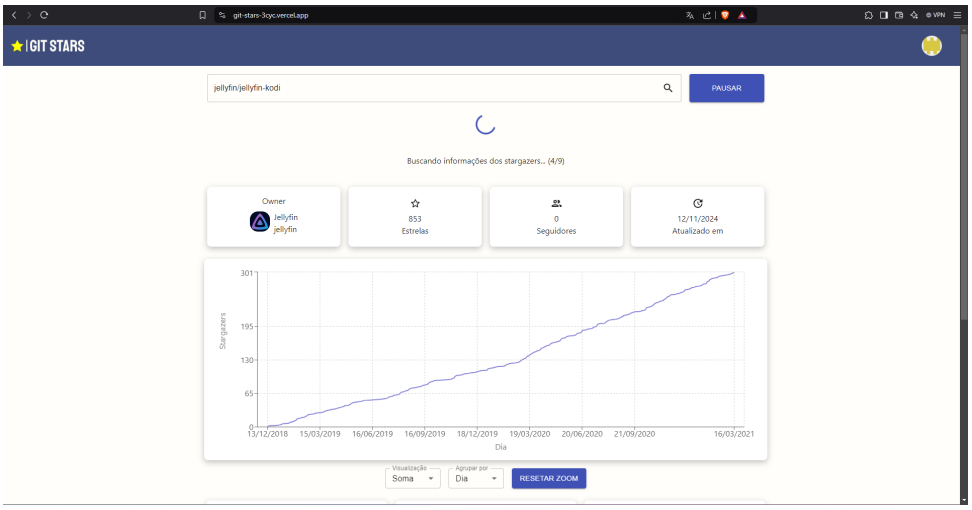
o status atual da busca, como demonstrado na Figura 5. Os dados são salvos em cache, através do PouchDB.



Fonte: Próprio Autor

Figura 5. Aplicação buscando dados do repositório

Em seguida, são exibidos os dados referentes ao repositório e se dá início à busca sequencial para trazer os dados dos usuários que favoritaram o repositório com uma estrela. As atualizações nos gráficos e nos rankings são feitas enquanto a coleta de dados está sendo realizada, permitindo acompanhar o gráfico sendo montado em tempo real, como mostrado na Figura 6. Mais uma vez, os dados recebidos são salvos em cache conforme as requisições vão sendo feitas, permitindo que o usuário saia da aplicação e retome o processo de coleta dos dados em outro momento.



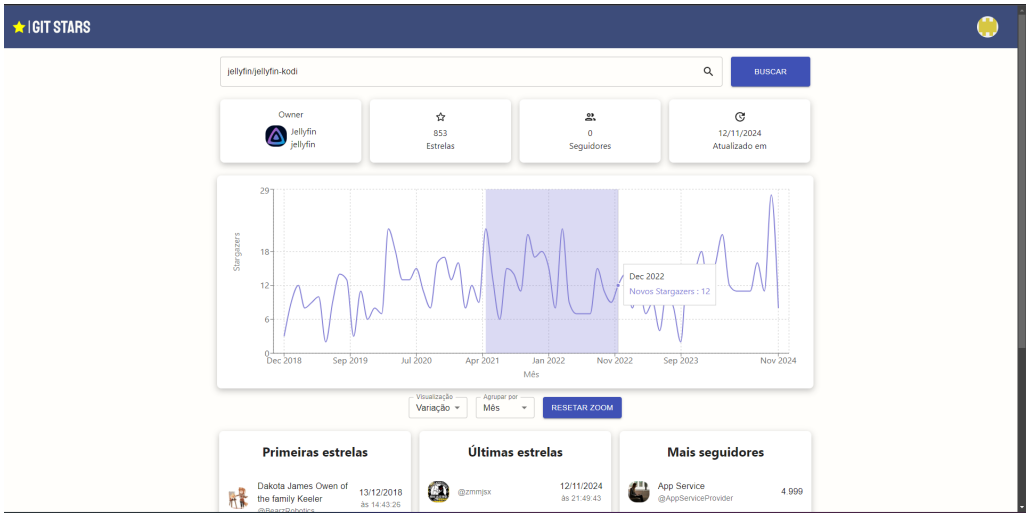
Fonte: Próprio Autor

Figura 6. Aplicação buscando dados dos stargazers

É possível modificar a forma como os dados são apresentados no gráfico, utilizando duas opções principais de visualização: o modo de soma e o modo de variação. O

modo de soma exibe o número total acumulado de stargazers em uma data específica, permitindo que o usuário tenha uma visão clara da evolução de stargazers ao longo do tempo. Este modo é particularmente útil para entender o crescimento geral do projeto e identificar marcos importantes, como quando o repositório atingiu certos números significativos de estrelas.

Já o modo de variação foca na quantidade de novos stargazers que se juntaram em um determinado período, fornecendo uma perspectiva mais dinâmica sobre o crescimento ou a flutuação no interesse ao longo do tempo. Nesta visualização é possível identificar picos de interesse, que podem estar relacionados a lançamentos importantes, viralização do projeto em plataformas sociais ou outros eventos, como demonstrado na Figura 7.



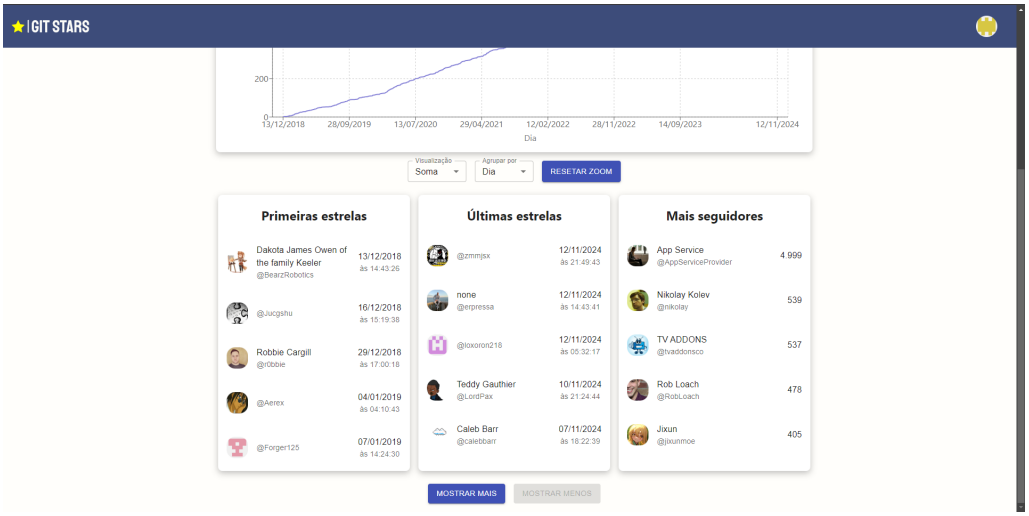
Fonte: Próprio Autor

Figura 7. Recursos do gráfico

Além dessas opções de visualização, o gráfico também oferece a funcionalidade de agrupar os dados em diferentes intervalos de tempo, sendo eles dia, semana, mês ou ano. Essa flexibilidade permite que os usuários analisem as tendências e padrões em diferentes escalas, permitindo melhores visões temporais. Por exemplo, a visualização diária pode ser útil para análises detalhadas de períodos específicos, enquanto a visualização anual pode revelar tendências de longo prazo no crescimento do projeto.

O usuário também pode ampliar uma sessão específica do gráfico ao clicar e arrastar sobre o período desejado. Essa funcionalidade facilita a análise de repositórios que possuem um grande número de stargazers, permitindo analisar de forma detalhada períodos específicos de interesse.

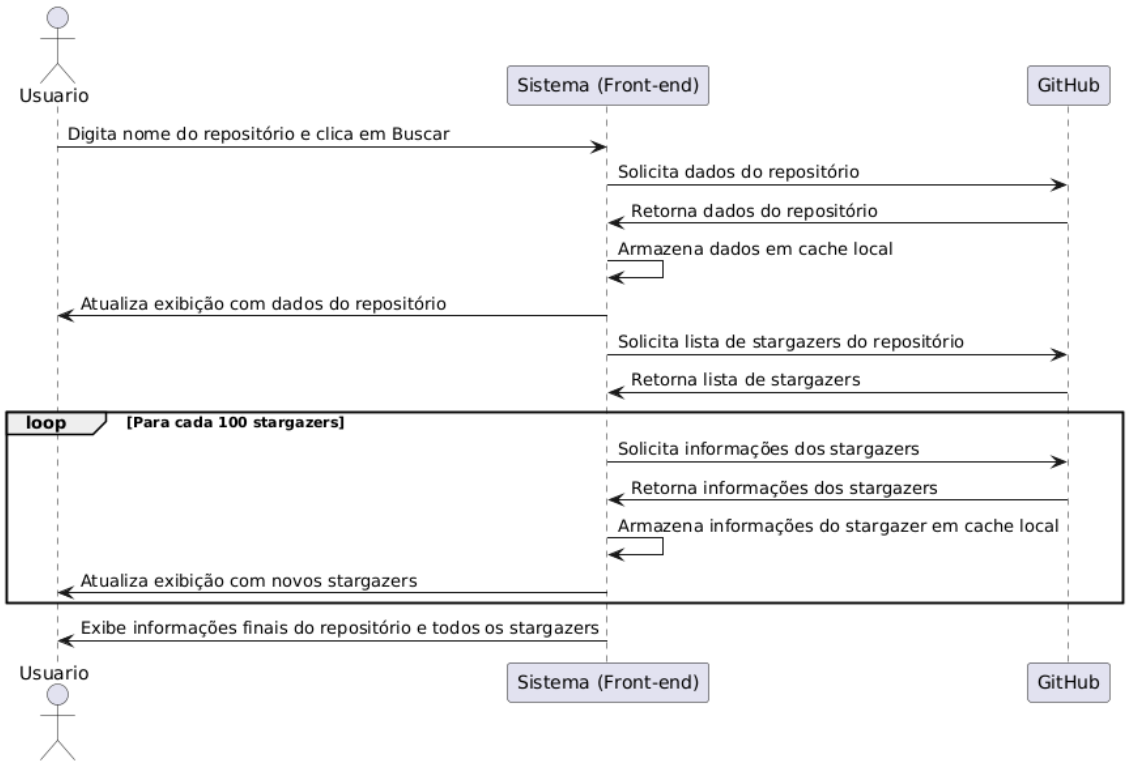
Abaixo do gráfico, demonstrado na Figura 8, é possível ver algumas informações sobre a comunidade deste repositório, com a possibilidade de expandir e restaurar o tamanho original das listas. São exibidos três rankings: os primeiros usuários a dar estrela no repositório, os últimos usuários que deram estrela e os usuários com o maior número de seguidores que favoritaram o repositório.



Fonte: Próprio Autor

Figura 8. Rankings dos usuários

Na Figura 9, observa-se o fluxo completo de busca de dados, que detalha como o sistema interage com o usuário e a API do GitHub.



Fonte: Próprio Autor

Figura 9. Diagrama de sequência do fluxo de busca de dados

6. Avaliação e Discussões

Para avaliar o desempenho da ferramenta, foram coletados dados durante a mineração de dados de diversos repositórios, exibidos na Tabela 1, sendo os dados: estrelas, tempo, tempo médio por requisição, espaço para armazenamento em disco e uso de memória RAM.

Essa avaliação tem também como objetivo verificar a viabilidade da abordagem utilizada, assim como identificar os pontos fortes e também os pontos de melhoria. As métricas escolhidas ajudam a determinar a escalabilidade e custo operacional da ferramenta.

Os testes foram realizados coletando dados de diferentes repositórios, com diferentes níveis de popularidade, variando entre 800 e 50.000 estrelas. A escolha desses repositórios foi feita de forma aleatória, buscando abranger diferentes faixas de popularidade para verificar o desempenho em diferentes cenários de uso. Os repositórios foram escolhidos independentemente da linguagem de programação principal e fazendo parte de diferentes domínios. Os testes foram feitos em um ambiente controlado, com conexão de internet estável e utilizando um computador com configurações comuns, simulado um cenário real de uso da ferramenta.

Na Tabela 1 é atribuída uma letra para cada repositório, a fim de facilitar a leitura dos gráficos criados com os dados de desempenho obtidos e a quantidade de estrelas e a linguagem principal utilizada em cada repositório. É possível analisar a amplitude na quantidade de estrelas entre os repositórios escolhidos. Uma grande variação nessa amplitude permite avaliar o desempenho da ferramenta com diferentes volume de dados, permitindo melhor validação da sua eficiência e escalabilidade.

Tabela 1. Tabela de Repositórios e Letras Correspondentes

Letra	Repositório	Estrelas	Linguagem principal
A	jellyfin/jellyfin-kodi	849	Python
B	timescale/pgai	1.782	PLpgSQL
C	Azure-Samples/cognitive-services-speech-sdk	2.899	C#
D	toss/es-toolkit	6.709	TypeScript
E	DataExpert-io/data-engineer-handbook	11.897	Makefile
F	fishaudio/fish-speech	14.166	Python
G	twentyhq/twenty	20.044	TypeScript
H	ManimCommunity/manim	25.919	Python
I	maybe-finance/maybe	33.542	Ruby
J	PowerShell/PowerShell	45.541	C#

No gráfico da Figura 10 pode-se analisar o tempo de coleta de dados dos repositórios. Observa-se um aumento crescente e linear no tempo necessário para buscar todos os dados do repositório conforme o número total de estrelas, mostrando que há previsibilidade e constância ao no tempo necessário para coletar esses dados.

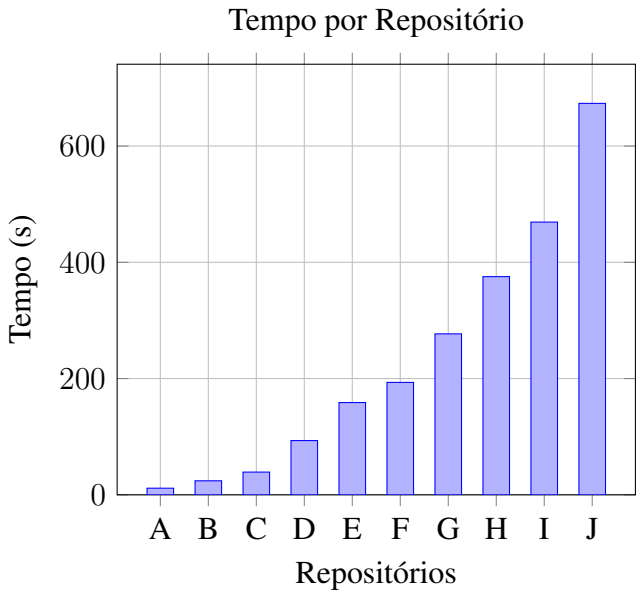


Figura 10. Tempo por Repositório

No gráfico da Figura 11 é possível observar que o tempo médio por requisição se mantém estável, com uma variação máxima de 200 milissegundos entre os repositórios escolhidos. Isso indica que a performance da ferramenta não é afetada pela quantidade de estrelas de um repositório.

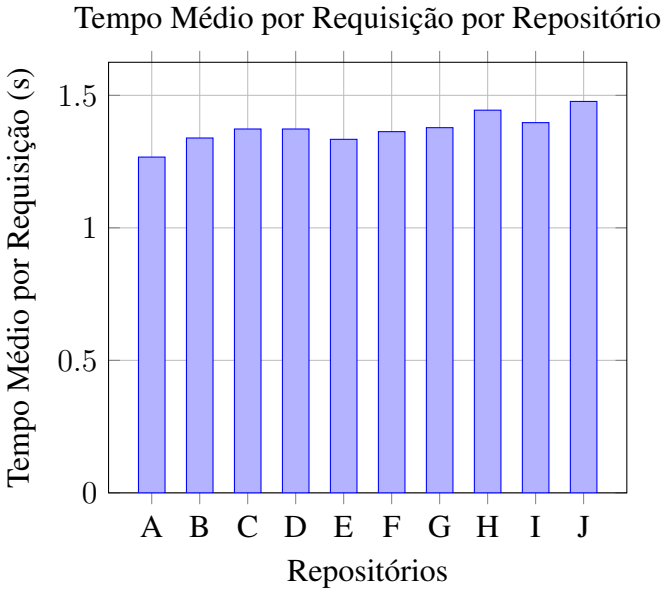


Figura 11. Tempo Médio por Requisição por Repositório em Segundos

No gráfico da Figura 12 é possível observar o espaço alocado em disco para armazenar as informações coletadas, onde existe um padrão predominantemente crescente no espaço alocado, com uma exceção do terceiro para o quarto repositório, onde houve uma redução de 9,14 MB para 8,39 MB, mesmo com o aumento do número de estrelas.

Esse crescimento demonstra como o aumento na quantidade de estrelas impacta diretamente na quantidade de dados armazenados, evidenciando uma dificuldade técnica para a implementação de um banco de dados externo, que precisa ter grande capacidade para suportar um grande número de repositórios populares.

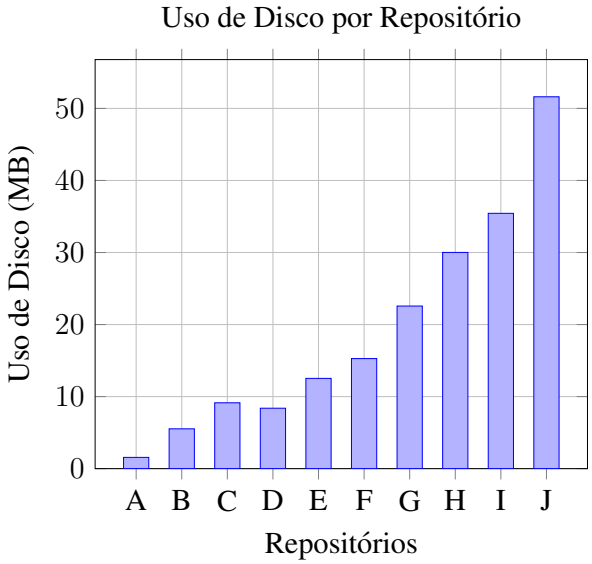


Figura 12. Uso de Disco por Repositório em Megabytes

Neste último gráfico, exibido na Figura 13, é possível observar que a alocação máxima de memória durante o processo de coleta de dados não apresenta um correlação direta com o tamanho dos repositórios. Existe uma oscilação significativa entre o quinto e o oitavo repositório (94,4 MB para 210,1 MB). Entretanto, a alocação de memória se mantém em níveis aceitáveis, indicando que a ferramenta pode ser utilizada mesmo em computadores com pouca quantidade de memória RAM.

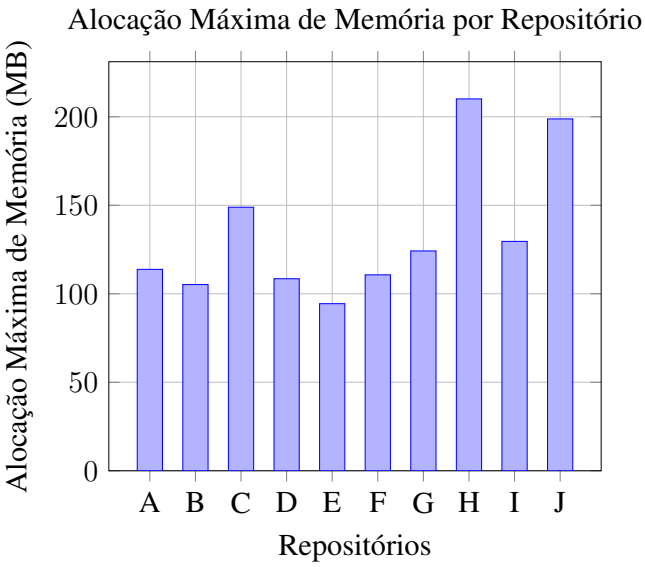


Figura 13. Alocação Máxima de Memória por Repositório

7. Conclusão

O objetivo deste trabalho foi desenvolver uma ferramenta que permita a coleta e visualização de dados sobre repositórios do GitHub diretamente no navegador, sem a necessidade da instalação de softwares específicos, contornando a limitação de obtenção de dados acima da estrela 40.000, enfrentada pelas soluções existentes que fazem uso das APIs REST.

Para isso, foi desenvolvida a aplicação Git Stars, que faz uso da biblioteca Gittrends-app, possibilitando assim a coleta de dados utilizando as APIs GraphQL do GitHub. A aplicação é capaz de obter dados de forma completa e precisa, faz uso de cache para armazenamento local e permite a visualização em tempo real dos dados enquanto são coletados, tudo isso diretamente do navegador.

Os resultados dos testes aplicados podem ser promissores, mostrando a capacidade da ferramenta de coletar informações de repositórios com grande quantidade de estrelas. Os tempos médios por requisição e uso de memória RAM permaneceram em níveis aceitáveis, mesmo durante a coleta de dados de repositórios grandes. O uso de disco cresce de acordo com a quantidade de estrelas, o que é esperado. Esses resultados demonstram a viabilidade de abordagem tomada.

Para trabalhos futuros, existem algumas melhorias e expansões que podem ser consideradas, como a visualização de outras métricas, como *forks*, *issues* e *pull requests*, mostrando o nível de atividade no repositório, a possibilidade de comparar vários repositórios simultaneamente, a implementação de um sistema de cache em um banco de dados externo e a implementação da funcionalidade de baixar um arquivo CSV com os dados coletados. Estas melhorias fazem de Git Stars uma ferramenta promissora para visualização de dados de repositórios do GitHub.

Referências

[1] GitHub. *100 Million Developers and Counting*. GitHub, Inc., 2024. URL: https://github.blog/news-insights/company-news/100-million-developers-and-counting/ (acesso em 20/11/2024).

[2] HypeStat. *GitHub Website Statistics*. 2024. URL: https://hypestat.com/info/github.com (acesso em 20/11/2024).

[3] Hudson Borges e Marco Tulio Valente. "What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform". Em: *Journal of Systems and Software* 146 (dez. de 2018), pp. 112–129. ISSN: 0164-1212. DOI: 10.1016/j.jss.2018.09.016. URL: http://dx.doi.org/10.1016/j.jss.2018.09.016.

[4] *GitHub REST API Documentation*. GitHub, Inc. URL: https://docs.github.com/en/rest?apiVersion=2022-11-28 (acesso em 26/11/2024).

[5] *GitHub GraphQL API Documentation*. GitHub, Inc. URL: https://docs.github.com/en/graphql (acesso em 26/11/2024).

[6] *Star History*. Star History. 2024. URL: https://star-history.com/ (acesso em 20/11/2024).

[7] *Starchart*. Starchart. URL: https://starchart.cc/ (acesso em 20/11/2024).

[8] Hudson Silva Borges. *@gittrends-app/core*. 2024. URL: https://www.npmjs.com/package/@gittrends-app/core (acesso em 20/11/2024).

[9] Microsoft. *TypeScript: Typed JavaScript at Any Scale*. 2024. URL: https://www.typescriptlang.org/ (acesso em 26/11/2024).

[10] Inc. Facebook. *React: A JavaScript library for building user interfaces*. 2024. URL: https://react.dev (acesso em 26/11/2024).

[11] Node.js Foundation. *Express - Framework web minimalista para Node.js*. 2024. URL: https://expressjs.com/pt-br/ (acesso em 26/11/2024).

[12] PouchDB. *PouchDB, the Javascript Database that Syncs!* 2024. URL: https://pouchdb.com/ (acesso em 26/11/2024).

[13] Yuri Seiti Loio Shiino. *Git Stars*. 2024. URL: https://git-stars-project.vercel.app/ (acesso em 22/11/2024).