Ferramenta de coleta de dados GitHub Token Donation

João Vitor Gazola[1], **Hudson Silva Borges**[1]

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

{joao.v.gazola, hudson.borges}@ufms.br

Abstract. *GitHub, one of the most popular tools for managing and sharing Git repositories, provides ways for anyone to access information about its platform, however, it restricts the types and amount of information that can be obtained, imposing barriers for researchers who need to collect data in large quantities over extended periods. To address this situation, it is necessary to somehow increase the imposed rate limits. That was when we propose creating an application that utilizes GitHub's authentication flow, allowing anyone registered on the website the option to support by donating their request limit for as long as they wish.*

Resumo. *O GitHub, uma das mais populares ferramentas de gestão e compartilhamento de repositórios Git, disponibiliza a quem quiser maneiras de obter às informações sobre sua plataforma, porém delimita quais informações e quanta informação pode ser obtida, restrições que se provam barreiras para pesquisadores que precisam coletar dados em quantidade por longos períodos. Para contornar essa situação é preciso aumentar de alguma forma o limite de taxas imposto. Foi proposta a criação de uma aplicação que utiliza o fluxo de autenticação do próprio GitHub para permitir que qualquer um com uma conta doe seu limite de requisições.*

1. Introdução

O GitHub desempenha um papel fundamental no mundo do desenvolvimento de software, fornecendo uma plataforma centralizada para colaboração, gerenciamento de versões e compartilhamento de código-fonte. Essa ferramenta desempenha um papel crucial ao facilitar a colaboração entre desenvolvedores, permitindo que equipes distribuídas trabalhem de maneira eficiente em projetos complexos.

Visando permitir que seus usuários possam obter informações sobre a plataforma através de seus próprios sistemas programados, o GitHub disponibiliza APIs (Interface de Programação de Aplicação) e, para evitar abusos ou ataques, limita a quantidade de informação que cada usuário obtém por tempo das mesmas.

Os limites se baseiam no número de requisições feitas por hora, sendo que cada requisição possui um limite na quantidade máxima de informações obtidas. É necessário consultar a documentação oficial do GitHub[1] para obter informações detalhadas e atualizadas, pois os limites podem ser ajustados pela plataforma ao longo do tempo conforme for necessário.

Esses limites têm se mostrado pequenos quando comparado com as necessidades de trabalhos acadêmicos como "Análise da Contribuição para Código entre Repositórios

[1]https://docs.github.com/

do GitHub"[Rocha et al. 2016], que teve como objeto de estudo mil repositórios para cada uma de 12 linguagens de programação. O trabalho "Entendendo o engajamento das comunidades front-end e back-end nos repositórios do GitHub"[Júnior et al. 2022] também menciona ter que limitar sua amostra de dados para atender os limites.

Este artigo tem como proposito endereçar estes obstáculos encontrados, introduzindo e descrevendo uma ferramenta desenvolvida para contornar esse limite sem haver abuso das diretrizes de uso. Para isso, aproveitamos o potencial do processo de autenticação oferecido pelo GitHub, feito para permitir que terceiros validem acessos em suas plataformas, conseguimos um *token* de acesso, sendo este o mesmo *token* que as APIs utilizam para aplicar seus limites de acesso.

Ao final do desenvolvimento obtivemos a ferramenta GitHub Token Donation [Gazola 2023], um sistema que realiza o processo de coleta e armazenamento dos *tokens* de acesso obtidos no processo de autenticação do GitHub, deixando-os disponíveis para que outros sistemas como o GitHub Proxy Server[2] [Borges 2020] os utilize em seus fluxos como desejarem, podendo assim aumentar o limite de requisições por hora para cada novo *token*.

2. GitHub e os desafios ao coletar seus dados

O GitHub é uma plataforma que armazena repositórios Git e permite interações dos seus usuários com os mesmos, assemelhando-se a uma rede social, só que para código. O nosso foco se volta para suas APIs que permitem aos desenvolvedores interagir e obter dados sobre todos os eventos do GitHub para suas aplicações.

Não é preciso muito para provar a importância do GitHub dentro do mundo acadêmico, uma simples busca pelo termo "GitHub"na ferramenta Google Scholar[3] retorna cerca de 182 mil resultados, isso apenas entre os anos de 2021 e 2023, muitos dos quais utilizaram dados retirados das APIs.

De acordo com sua documentação[4], para evitar abusos e manter a estabilidade do serviço, o GitHub impõe limites nas consultas à API, controlando o número de solicitações que um usuário pode fazer em um determinado período. Isso é feito ajuda garantir uma distribuição equitativa dos recursos e evita possíveis sobrecargas ou ataques de negação de serviço.

O limite primário para aplicações não relacionadas ao GitHub Enterprise Cloud em 2023 foi de 5 mil requisições por hora, junto disso, os limites secundários identificam como abuso, *tokens* que realizam muitas requisições em paralelo (Figura 1) ou por minuto.

Projetos como "GitHub REST API vs GHTorrent vs GitHub Archive: A Comparative Study"[Mombach and Valente 2018] buscam analisar os métodos de obtenção de dados do GitHub e, ferramentas como o GitHub Proxy Server[2] [Borges 2020] auxiliam a obtenção desses dados, organizando as consultas a API e utilizando um ou múltiplos *tokens* de acesso, balanceando a carga entre eles para impedir que o GitHub detecte abusos.

O GitHub Proxy Server[5] [Borges 2020] ajuda a mitigar o efeito dos limites de

[2]https://github.com/gittrends-app/github-proxy-server
[3]https://scholar.google.com/
[4]https://docs.github.com/
[5]https://github.com/gittrends-app/github-proxy-server

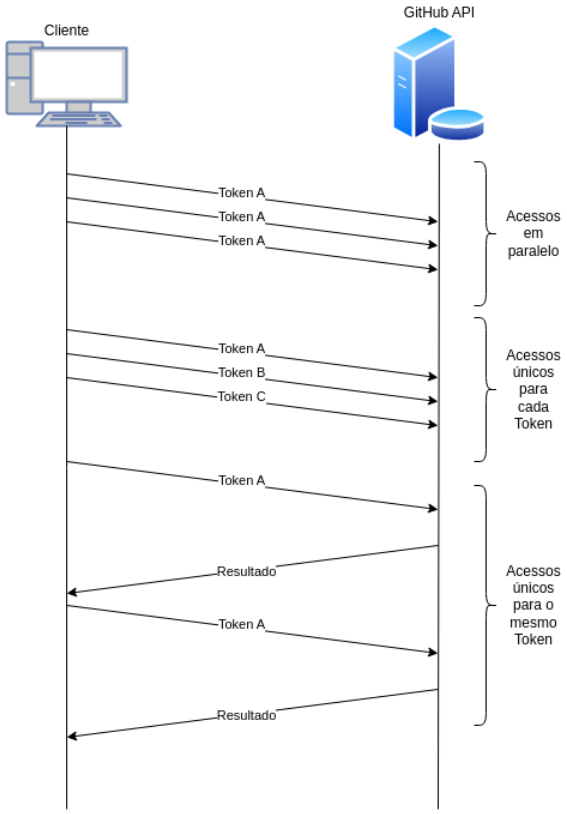

Figura 1. Exemplo abstrato de requisições únicas e em paralelo.

taxas, porém ele não fornece *tokens* de acesso, o que de fato aumentaria a quantidade de dados por medida de tempo, nisso surge a necessidade do projeto descrito neste artigo, o objetivo do mesmo é permitir que qualquer um que o use, tenha uma maneira de pedir que seus usuários cedam seus limites, doando *tokens* se assim desejarem.

3. *Tokens* de acesso *OAuth* para autenticação com GitHub

3.1. O que são os *tokens OAuth*?

Tokens OAuth são chaves utilizadas para validar acessos de aplicações terceiras em serviços que seguem o modelo de API, garantindo a segurança dos dados de seus usuários.

O protocolo *OAuth* permite que um *token* seja gerado e garanta acesso credenciado sem revelar as credenciais necessárias para gerá-los, esse *token* pode ser refeito ou excluído a qualquer momento pelo usuário e pode também possuir um tempo de expiração.

O GitHub utiliza o protocolo *OAuth* para que outras aplicações obtenham informações de seus usuários que concordarem, o GitHub se encarrega de informar a seus usuários que permissões estão sendo requisitadas pela aplicação terceira, para que assim decidam se permitem estes acessos.

Acessando as configurações de desenvolvedor nas configurações de conta do GitHub é possível gerenciar aplicações integradas ao GitHub e também *tokens* de acesso pessoais, tanto os gerados ao acessar outras aplicações ou até mesmo seus próprios *tokens*. Para usar um *token*, basta enviá-los junto de qualquer requisição HTTP (Protocolo de Transferência de Hipertexto) realizada na API.

3.2. APIs do GitHub

Para realizar uma requisição as APIs do GitHub, é preciso informar um *token OAuth* que o GitHub usa para identificar qual aplicação ou usuário está fazendo aquela requisição e quanto sobrou do limite de consumo da API para aquele *token*. Caso um *token* não seja fornecido, o GitHub ira contabilizar o limite se baseando no endereço de rede da máquina que fez o acesso, sendo este limite muito inferior.

Os dados do GitHub são organizados em uma estrutura de nós de informações interligados, com cada nó possuindo seu próprio identificador único e imutável. O GitHub implementa o limite de taxas em suas APIs limitando a quantidade de nós que cada usuário consegue obter por período, impedindo assim acessos que poderiam prejudicar o uso da plataforma.

O GitHub prove dois tipos de APIs para quem quer acessar seus dados, ambas possuem acesso ao escopo do GitHub e respondem a requisições do tipo HTTP, cabe ao desenvolvedor escolher a que melhor se encaixa no seu escopo.

A API do tipo REST[6] recebe requisições em pontos de acesso diferentes para cada artefato consultado, o que a torna menos eficaz para alguns casos, pois é preciso receber muitos nós de informações desnecessários junto das informações desejadas.

A API *GraphQL*[7], sendo a mais recente, consegue retornar dados de maneira mais eficiente que a sua predecessora, pois ela recebe requisições em apenas um ponto de acesso e, com a consulta inteira já descrita no corpo da requisição, podendo retornar apenas os nós necessários sem muito desperdício de limite.

3.3. Fluxo de autenticação do GitHub para coleta de *tokens*

O GitHub possui um fluxo de autenticação para outras plataformas que queiram permitir que seus usuários usem as credenciais das suas contas no GitHub para validar acesso, o desenvolvedor que queira esse fluxo em sua plataforma deve criar uma aplicação no menu de desenvolvedor de sua conta do GitHub, obtendo uma chave de acesso pública e privada.

Quando um novo usuário deseja cadastrar-se pelo GitHub em sua aplicação, ele deve ser redirecionado para o endereço de autenticação do GitHub levando a chave pública da aplicação, após inserir as suas credenciais, o GitHub informa ao usuário quais permissões estão sendo solicitadas pela aplicação (Figura 2), aceitando o usuário retorna para a aplicação com um código de acesso temporário.

[6]https://docs.github.com/en/rest?api
[7]https://docs.github.com/en/graphql

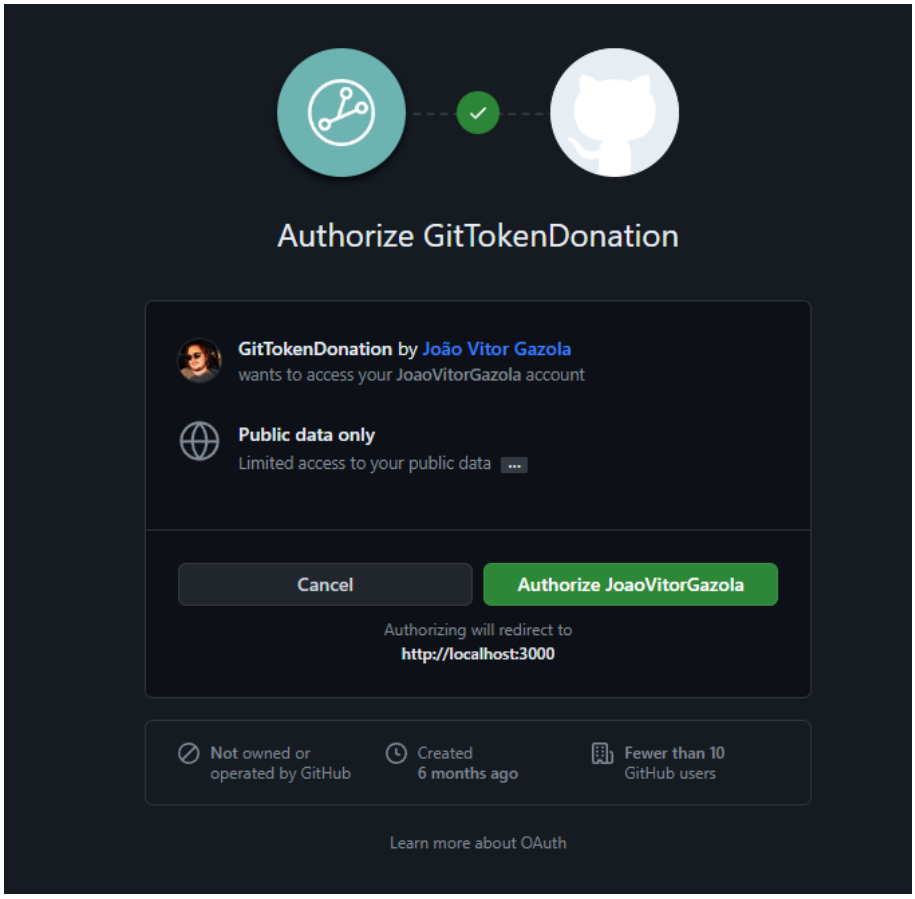

Figura 2. Tela de autenticação do GitHub

Após a autenticação, é responsabilidade da aplicação usar este código em conjunto com a sua chave privada para obter um *token OAuth* relativo ao usuário que fez o acesso, validando assim o seu acesso com a conta do GitHub. Com este *token* pode-se acessar todos os dados relativos às permissões aprovadas pelo usuário, gastando apenas o limite do mesmo, o que significa que cada *token* doado aumenta a quantidade de informações que pode ser obtida da API por unidade de tempo.

4. Ferramentas e metodologia

4.1. Node.js[OpenJS Foundation 2009]

Node.js[8] é um programa desenvolvido com base no interpretador V8 do Google para executar códigos JavaScript fora de um navegador. Sua arquitetura assíncrona e orientada por eventos, possibilita a execução eficiente de projetos que demandam alta concorrência e escalabilidade.

A arquitetura assíncrona do Node.js permite que várias operações sejam executadas simultaneamente, sem bloquear a execução do programa, o que é essencial para lidar com a concorrência em projetos de grande escala.

Desenvolvido com o foco em aplicações de rede, Node.js possui uma transmissão de dados de baixíssima latência, ele consegue isso graças à arquitetura *single-thread* (fio

[8]https://nodejs.org

de execução único), onde um laço de eventos fica responsável por lidar com novas chamadas ao sistema, economizando poder de processamento no processo.

Ao desenvolver uma aplicação de rede, arquiteturas com várias *threads* acabam sendo ineficientes e difíceis de usar, já no Node.js quase nenhuma tarefa de entrada ou saída é realizada diretamente, ao invés disso, seu funcionamento padrão realiza todas as operações sem realizar bloqueios, deixando que o desenvolvedor seja responsável por dizer quando a aplicação deve esperar. Essa abordagem auxilia aplicações de rede a sempre continuar executando, independente da quantidade de acessos simultâneos.

Sua arquitetura *single-thread* não significa que seus usuários não possam tirar proveito de arquiteturas com várias *threads*, pois é possível criar novas conforme necessário, e elas conseguem se comunicar facilmente, outra funcionalidade que facilita o uso de *multithreading* (múltiplos fios de execução) é dividir *sockets* entre *threads* através de seu módulo de *cluster* permitindo balancear a carga entre os núcleos de um processador.

Node.js prove o gerenciador de pacotes NPM (Gerenciador de Pacotes do Node.js) que possui suporte de uma comunidade de usuários muito ativa, existindo várias bibliotecas de código aberto e prontas para uso, economizando muito tempo no desenvolvimento e possibilitando a utilização de soluções já consolidadas por qualquer usuário de Node.js. Essas e outras informações sobre o Node.js estão disponíveis em sua documentação[9].

Para que *tokens* sejam obtidos, armazenados e acessados posteriormente existe a necessidade de uma aplicação de rede que acesse uma API externa capaz de manipular dados confidenciais aos usuários da rede, tanto para obter quanto para armazenar os *tokens*, sendo esta uma tarefa trivial para muitas das ferramentas escolhidas para o projeto que tem o Node.js como base.

4.2. Next.js [Vercel, Inc. 2016]

Next.js[10] é um *framework* de código aberto desenvolvido pela Vercel que utiliza a biblioteca React junto de funcionalidades como renderização do lado do servidor que não estão disponíveis ao React nativamente.

Ele é especialmente útil para este projeto, pois sua arquitetura divide o código entre código de servidor e cliente, permitindo que *tokens* sejam obtidos sem revelar chaves secretas de acesso à API do GitHub ou credenciais de acesso a banco de dados, processando informações no servidor sem que as mesmas sejam enviadas ao lado do cliente.

O Next.js tem integrado de forma simples a capacidade de criar rotas para APIs, facilitando tanto a comunicação entre o lado cliente com o lado servidor quanto do lado do servidor com banco de dados e APIs do GitHub, onde obteremos os *tokens*.

Outra funcionalidade disponível é a pré-renderização estática, o Next.js permite que as partes estáticas de uma página sejam geradas em tempo de compilação, ele consegue isso dividindo automaticamente o código necessário para renderizar as partes estáticas de uma página, resultando num tempo de carregamento mais rápido das páginas para o usuário final.

Apesar de seu foco em renderização hibrida, é possível também exportar páginas

[9]https://nodejs.org/docs/latest/api/
[10]https://nextjs.org/

que não necessitam de um servidor como arquivos estáticos, podendo no futuro adicionar um servidor ao projeto sem que haja necessidade de migração do código para outras tecnologias que resultaria em retrabalho.

O desenvolvedor não foi deixado de lado pelo Next.js, ele inclui em si um ambiente de desenvolvimento integrado, permitindo o desenvolvedor modificar o projeto sem precisar parar a sua execução e atualizando imediatamente ao identificar uma mudança, agilizando as iterações durante o desenvolvimento.

Sua documentação[11] ensina quais e como utilizar as funcionalidades implementadas, tornando-o um sistema fácil de se aprender.

Apesar da base do Next.js, o React, ser por conta própria ser uma biblioteca poderosa para construção de interfaces de usuário, não foi o bastante para atender nossa aplicação por inteiro, exigindo as funcionalidades apenas disponíveis com o Next.js.

4.3. React[Meta Open Source 2013]

React[12] é uma arquitetura e biblioteca de código aberto voltada a facilitar o desenvolvimento de aplicações *front-end*, mantido pelo Facebook e utilizado dentro de várias empresas de grande porte, o React possui uma base solida de suporte e segurança, e uma documentação[13] com muitos guias e informações para auxiliar o seu aprendizado.

Seu foco como ferramenta está em criar interfaces gráficas de páginas *web*, se mostrando muito útil no projeto tanto para criar uma interface para os usuários comuns que doaram *tokens*, quanto para visualizá-los em uma lista ordenada.

A maioria dos navegadores trabalha com um padrão internacional conhecido como DOM (Modelo de Objeto de Documentos), que organiza os objetos disponíveis em um servidor numa estrutura de árvore. O React trabalha modificando um DOM virtual e sincronizado com o DOM verdadeiro alterando apenas o necessário para renderizar as novas páginas, esse modelo permite uma abstração mais legível e facilitada das funcionalidades do DOM.

A cada renderização o React gera um novo DOM virtual, compara as diferenças com o DOM real usando algoritmos de *diffing* para assim, gerar um novo DOM virtual, composto do antigo DOM somando as alterações encontradas.

O React promove a modularização dos componentes e interfaces de usuários, permitindo que o desenvolvedor crie componentes isolados para cada parte necessária na interface a ser desenvolvida, resultando num código mais legível e organizado.

Sendo parte da comunidade de desenvolvimento Node.js, através dos gerenciadores de pacotes tiramos proveito de um grande acervo de bibliotecas com ferramentas prontas para uso, o React também recebe frequentes atualizações e tem um histórico de suporte a longo prazo, mantendo alta confiabilidade e estabilidade ao longo do tempo.

É necessário ressaltar que existem outros *frameworks* e bibliotecas, mesmo pelos gerenciadores de pacotes do Node.js, que não foram escolhidos para levar em

[11] https://nextjs.org/docs
[12] https://react.dev/
[13] https://react.dev/learn

consideração padrões de desenvolvimento de código aberto e históricos de suporte de longa data.

Apesar do React usar por padrão a linguagem JSX (JavaScript XML), muito semelhante ao HTML, a linguagem principal escolhida para o projeto foi o TypeScript, que o React tem suporte nativo.

4.4. TypeScript[Microsoft 2012]

TypeScript[14] é uma linguagem de programação criada pela Microsoft. Sua tipagem estática com base em JavaScript e código aberto foram os principais motivos que levaram a sua escolha para esse projeto. A linguagem funciona como um complemento ao JavaScript, podendo ser utilizado em muitos *frameworks* e bibliotecas JavaScript, no nosso caso, tanto o Node.js quanto o React.

Introduzindo tipagem estática ao JavaScript obtemos uma linguagem de desenvolvimento mais segura, pois isto permite definições de tipo para variáveis e parâmetros, reduzindo não só a possibilidade de erros durante o desenvolvimento como aumentando a manutenibilidade e a segurança do código.

Erros podem ser detectados em tempo de compilação, pois o próprio compilador informará sobre variáveis e parâmetros mal utilizados antes que o projeto possa rodar, diminuindo casos onde o projeto falharia em produção por má utilização de recursos ou falhas na lógica.

Este exemplo em JavaScript (sem tipagem estática) não apresenta erros, mas o resultado não é o esperado:

```
function soma(a, b) {
    return a + b;
}
console.log(soma(5, "10"));
```

Aplicando TypeScript (com tipagem estática) no código anterior, ele apresentaria erro de compilação:

```
function soma(a: number, b: number): number {
  return a + b;
}
console.log(soma(5, "10"));

//Argumento do tipo 'string' não é atribuível ao parâmetro
do tipo 'number'.
```

Graças à tipagem estática o código se torna mais legível e fica mais fácil dar manutenção, pois é possível entender a utilidade das variáveis e parâmetros de maneira mais clara sabendo o seu tipo. A produtividade do desenvolvedor também é aprimorada, pois o mesmo pode utilizar de ferramentas de autocompletar que se baseariam no tipo das variáveis disponíveis.

[14]https://www.typescriptlang.org/

4.5. MUI[Material UI SAS 2018]

MUI[15] (Material-UI) é uma biblioteca de componentes visuais para o React desenvolvida pelo Google. Ela fornece componentes visuais estilizados e reutilizáveis, facilitando a criação das interfaces de acesso necessárias no projeto.

A biblioteca permitiu diminuir o tempo gasto com a criação de uma interface intuitiva para o usuário, liberando mais tempo para ser gasto com outras partes, pois questões como legibilidade e responsividade, que de outro modo tomariam tempo para serem resolvidas, podem ser abordadas de maneira mais rápida com esta biblioteca.

A usabilidade de um sistema em dispositivos com telas de formatos diferentes é comprometida se o desenvolvimento do mesmo não leva responsividade em consideração, é por isso que os componentes visuais da MUI funcionam com layouts responsivos, reduzindo a necessidade de criar adaptações para telas com tamanhos variados.

Cada um de seus componentes, que são módulos independentes para variadas funcionalidades, possui sua seção na documentação[16], que explica todas as maneiras que se pode utilizá-lo e mostra o resultado de cada uma delas.

4.6. Banco de dados

Se tratando de banco de dados, não existe nenhuma exigência do projeto que incentive a escolha de uma solução específica, pois é necessário apenas uma tabela de dados com um campo para o identificador único e outro para o token e ambos não são particularmente extensos.

Dessa maneira, qualquer tecnologia de banco de dados seria aplicável, a escolhida para essa versão do projeto foi o MongoDB[17] [MongoDB, Inc. 2009], que durante o desenvolvimento foi o mais simples de se implantar no Next.js.

5. Ferramenta proposta

5.1. Descrição geral

O projeto GitHub Token Donation[18] [Gazola 2023] é uma ferramenta que segue a arquitetura cliente-servidor, e auxilia a coleta e armazenamento de *tokens* de acesso *OAuth* GitHub para quaisquer que sejam os usos necessários.

Seu desenvolvimento se deu pela necessidade de acelerar a análise apresentada nos estudos de Hudson Borges, Andre Hora e Marco Tulio Valente sobre a quantidade de estrelas obtidas por cada repositório [Borges et al. 2016a, Borges et al. 2016b, Borges and Valente 2018], mas qualquer aplicação que necessite ultrapassar o limite de requisições das APIs do GitHub pode precisar utilizar esta ferramenta.

O projeto está organizado com uma arquitetura cliente-servidor, cada lado é capaz de trocar informações entre si sem saber o que de fato o outro está fazendo, ambos se comunicam com APIs externas, mas apenas o lado do servidor é capaz de enxergar os dados confidenciais da aplicação, nunca os expondo para o cliente.

[15] https://mui.com/

[16] https://mui.com/material-ui/getting-started/

[17] https://www.mongodb.com/

[18] https://github.com/JoaoVitorGazola/GitHubTokenDonation

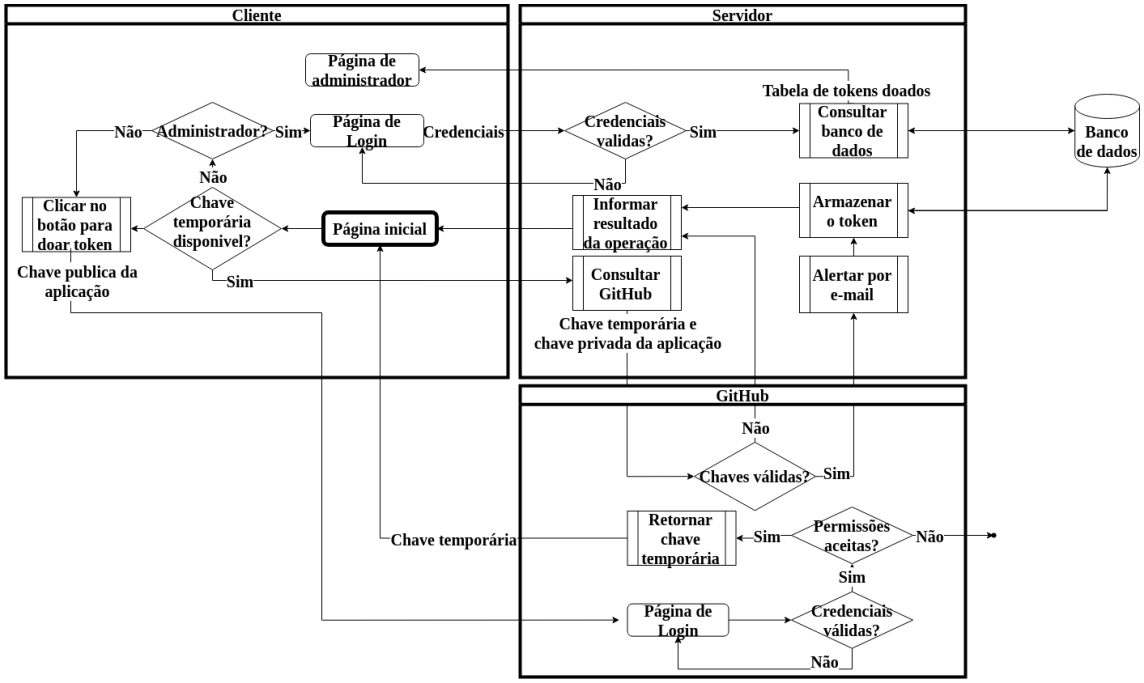

Figura 3. Diagrama de fluxo da ferramenta.

5.2. Analise detalhada do fluxo

A ferramenta consiste em uma aplicação com três telas, sendo elas a tela inicial, de acesso do administrador e a tabela de *tokens* obtidos, todas possuem uma barra de navegação lateral que permite a navegação entre si e exibe um nome e o ícone da aplicação.

A tela inicial (Figura 4) possui um título e mensagem que visam informar o leitor da necessidade de *tokens*, solicitando suporte com uma doação, a tela também mostra um botão que redireciona ao fluxo de acesso do GitHub e uma caixa de notificações que se faz visível quando necessária.

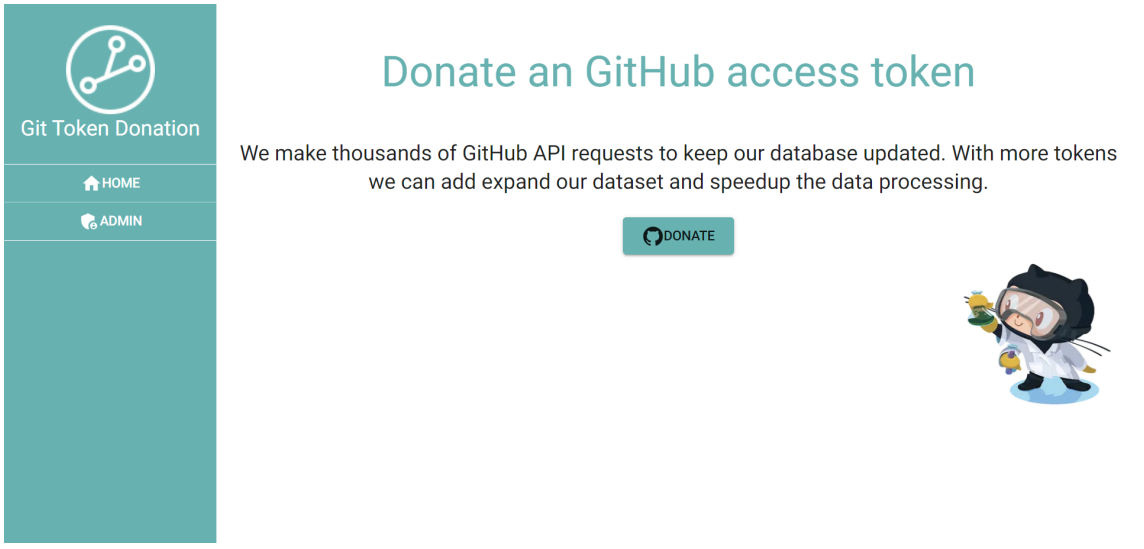

Figura 4. Página inicial da aplicação.

Ao ser redirecionado para o acesso do GitHub, o usuário leva consigo a chave pública da aplicação, assim o GitHub consegue identificar qual organização e permissões são relativas ao *token* temporário que ele deve gerar. Esse *token* é então devolvido a aplicação original junto do usuário.

Entretanto, este ainda não é o *token* de acesso *OAuth* que necessitamos, ao acessar a página inicial com ele dentro das variáveis da URL (Localizador Uniforme de Recursos), a aplicação entende que deve seguir um fluxo alternativo, para só assim obter o *token* correto. O lado do cliente envia a chave temporária para o lado do servidor que a utiliza em conjunto da chave privada da aplicação, para pedir ao GitHub o *token OAuth*.

O servidor salva o *token* recebido no banco de dados, envia um alerta no e-mail do administrador e informa o lado do cliente a situação do procedimento, para que o mesmo tenha um retorno visual informando no caso de erro ou sucesso (Figuras 5 e 6).

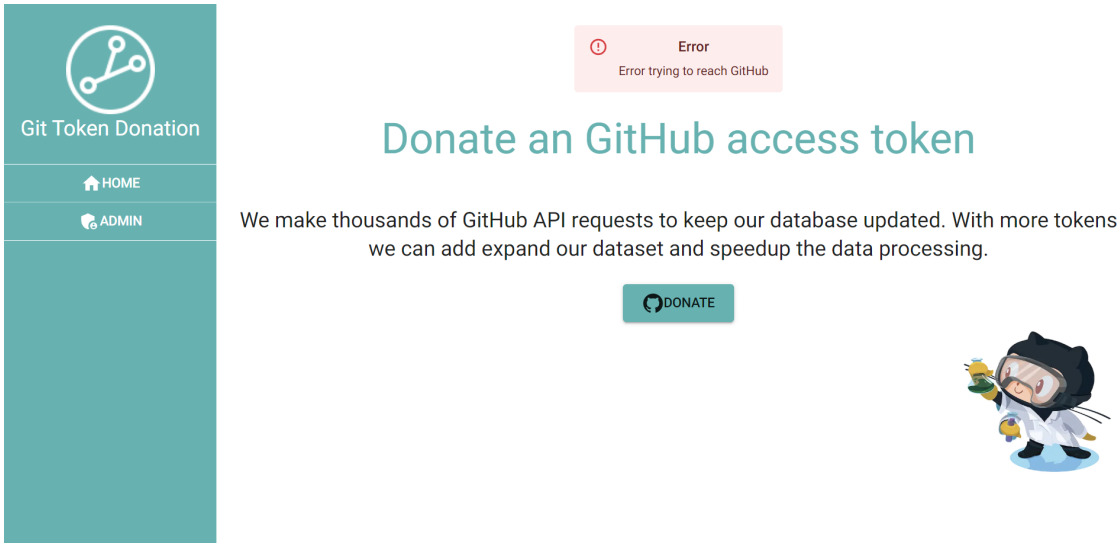

Figura 5. Página inicial com alerta de erro no fluxo da aplicação.

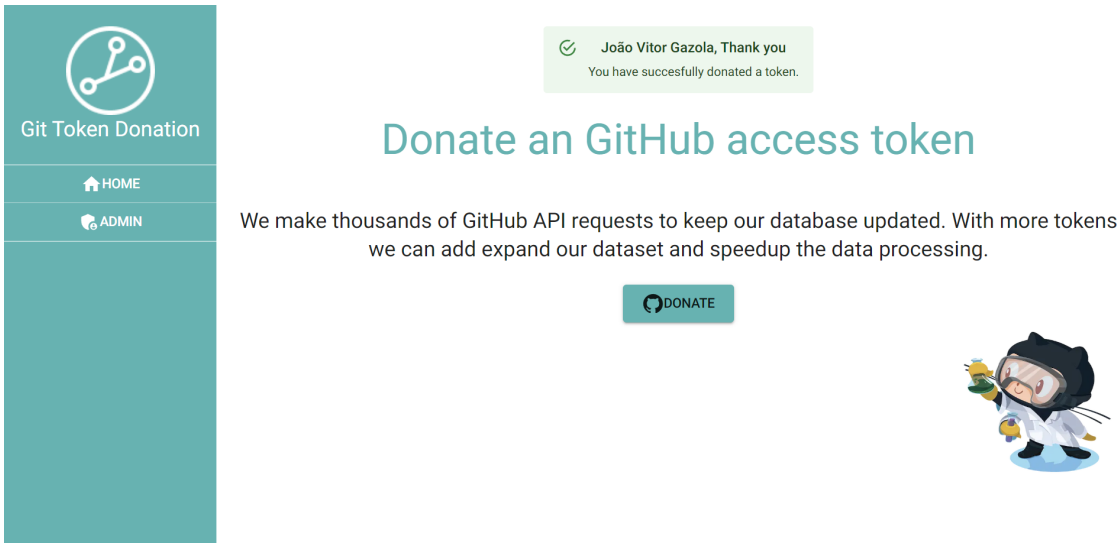

Figura 6. Página inicial com alerta de sucesso ao doar *token*.

A necessidade de se comunicar com um servidor vem principalmente da importância em manter a chave privada de aplicação escondida do usuário, mas também existe necessidade de armazenar os *tokens* de forma segura acessando um servidor de banco de dados, o que envolve mais informações que devem permanecer privadas.

Outro uso para a arquitetura cliente-servidor vem das telas que ainda não foram descritas, a tela de acesso de administrador (Figura 7) precisa validar com o servidor se as informações de acesso fornecidas coincidem com o configurado para só assim liberar o acesso de administrador e redirecioná-lo para a terceira tela.

A terceira tela (Figura 8) exibe uma tabela com todos os *tokens* doados que estão salvos no banco de dados, como são informações confidenciais, essa tela é de acesso exclusivo do administrador, para garantir isso, o usuário é redirecionado para a tela de acesso até que informe as credenciais corretas.

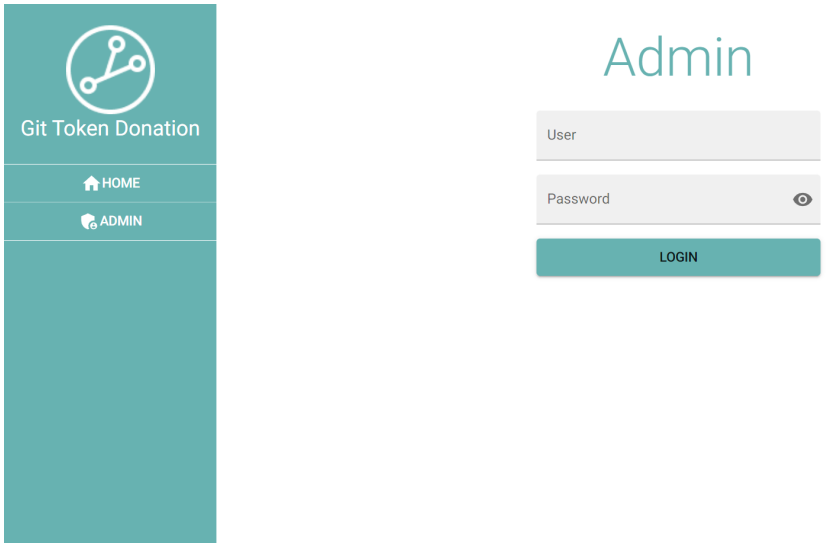

Figura 7. Página de acesso de administrador

Figura 8. Página de administrador com tabela de *tokens* doados.

5.3. Implantação

Para que o sistema seja implantado, é necessário qualquer computador com sistema operacional compatível com o Node.js de versão 20 ou superior, um banco de dados MongoDB que tenha permissão de acesso e os arquivos do projeto.

O desenvolvedor deve criar na raiz do projeto um arquivo de nome ".env.local", copiar nele o conteúdo do arquivo ".env.example"e preencher os valores de variáveis, para entender melhor que valores devem ser preenchidos em cada variável, elas foram divididas em grupos.

- **URLs:** Domínio escolhido para instância da aplicação (localhost:3000 no caso de instâncias locais) e URLs de APIs do GitHub.
- **GitHub secrets:** Chaves pública e privada da aplicação geradas no GitHub do desenvolvedor. Para gerá-las, basta acessar as configurações de conta e encontrar a área de opções de desenvolvedor, onde se é feito o cadastro e gestão da aplicação, alterando o ícone, nome da aplicação, quais permissões o token deve pedir e para qual URL redirecionar quando uma chave temporária for gerada.
- **Admin login info:** Informações para acesso do administrador e e-mail para qual enviar notificação de novo *token* recebido.
- **E-mail secrets:** Variáveis contendo configurações necessárias para que um e-mail informando um novo *token* seja enviado, é possível desativar os e-mails marcando a variável "SMTP"com o valor "false".
- **MongoDB secrets:** Variáveis de acesso a um banco de dados MongoDB.
- **Messages:** Todos os textos da interface para o cliente podem ser alterados aqui.

Tendo então o servidor Node.js instalado, e as variáveis do arquivo ".env.local"preenchidas, basta seguir o fluxo de implementação padrão do Next.js[19], executando os comandos a seguir na raiz do projeto a aplicação se fará disponível no navegador pelo endereço configurado.

```
npm install # Instala todos as dependencias necessárias
npm run build # Constroi os arquivos de produção
npm run start #Inicia servidor Node.js para a aplicação
```

6. Conclusões

O Objetivo deste projeto foi encontrar maneiras de auxiliar processos de coleta de dados das APIs do GitHub, a ferramenta desenvolvida auxilia nessa coleta aumentando a quantidade que se é possível retirar da API por período.

O objetivo original desse estudo era encontrar uma maneira de utilizar os recursos computacionais de quem permitisse para acessar às APIs com os *tokens* doados, repassando procedimentos de manuseio dos dados que pesam servidores quando realizados em larga escala, porém barreiras de confiabilidade de dados inviabilizaram essa parte do desenvolvimento.

[19]https://nextjs.org/docs/app/building-your-application/deploying

Melhorias futuras podem começar com a implementação de um sistema que permita múltiplos banco de dados, visto que atualmente apenas MongoDB é suportado. Outras melhorias podem envolver mais configurações e melhorias na interface gráfica do projeto.

Qualquer desenvolvedor pode obter o código desta ferramenta no repositório do GitHub e alterar o código da aplicação para implementar o uso dos *tokens* e obter os dados que necessita, seguindo as restrições da licença MIT (Instituto de tecnologia de Massachusetts).

O GitHub Token Donation está disponível em: https://github.com/JoaoVitorGazola/GitHubTokenDonation

Referências

[Borges et al. 2016a] Borges, H., Hora, A., and Valente, M. T. (2016a). Predicting the popularity of GitHub repositories. In *Proceedings of The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*. ACM.

[Borges et al. 2016b] Borges, H., Hora, A., and Valente, M. T. (2016b). Understanding the factors that impact the popularity of github repositories. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 334–344.

[Borges and Valente 2018] Borges, H. and Valente, M. T. (2018). What's in a GitHub star? understanding repository starring practices in a social coding platform. *Journal of Systems and Software*, 146:112–129.

[Borges 2020] Borges, H. S. (2020). GitHub Proxy Server. `https://github.com/gittrends-app/github-proxy-server`.

[Gazola 2023] Gazola, J. V. (2023). GitHub Token Donation. `https://github.com/JoaoVitorGazola/GitHubTokenDonation`.

[Júnior et al. 2022] Júnior, A. A., Meireles, L., Figueira, L., Carmo, V., Marques-Neto, H., and Xavier, L. (2022). Entendendo o engajamento das comunidades front-end e back-end nos repositórios do github. In *Anais do X Workshop de Visualização, Evolução e Manutenção de Software*, pages 26–30, Porto Alegre, RS, Brasil. SBC.

[Material UI SAS 2018] Material UI SAS (2018). Material UI. `https://mui.com/material-ui/`. Acessado durante o ano de 2023 e 2024.

[Meta Open Source 2013] Meta Open Source (2013). React. `https://react.dev/`. Acessado durante o ano de 2023 e 2024.

[Microsoft 2012] Microsoft (2012). TypeScript. `https://www.typescriptlang.org/`. Acessado durante o ano de 2023 e 2024.

[Mombach and Valente 2018] Mombach, T. and Valente, M. T. (2018). Github rest api vs ghtorrent vs github archive: A comparative study.

[MongoDB, Inc. 2009] MongoDB, Inc. (2009). MongoDB. `https://www.mongodb.com/pt-br`. Acessado durante o ano de 2023 e 2024.

[OpenJS Foundation 2009] OpenJS Foundation (2009). Node.js. `https://nodejs.org/`. Acessado durante o ano de 2023 e 2024.

[Rocha et al. 2016] Rocha, L., Silva, T. H., and Moro, M. (2016). Análise da contribuição para código entre repositórios do github. In *Anais do XXXI Simpósio Brasileiro de Bancos de Dados*, pages 103–108, Porto Alegre, RS, Brasil. SBC.

[Vercel, Inc. 2016] Vercel, Inc. (2016). Next.js. `https://nextjs.org/docs`. Acessado durante o ano de 2023 e 2024.