HealthyEnv: a tool to assist in health assessment of software repositories

Diego Winter, Guilherme Avelino e Charles Miranda
{diegowinter,gaa,charlesmiranda}@ufpi.edu.br
Universidade Federal do Piauí
Teresina, Piauí, Brasil

ABSTRACT

Most of the software developed today, whether proprietary or open source, has as its base infrastructure for its construction software components developed by open source communities. In this context, the concern with the quality, reliability and maintainability of such software components is of great importance. To evaluate and monitor software and its development community, tools are needed to assist in the process. Thus, in this work we present a tool, called HealthyEnv, which helps in the evaluation of the health of software projects, providing not only a series of metrics automatically computed from the software repository, but also reference values to assist in the interpretation of results.

RESUMO

A maior parte do software desenvolvido atualmente, seja proprietário ou open source, tem como infraestrutura base para sua construção componentes de software desenvolvidos por comunidades open source. Nesse contexto, é de grande importância a preocupação com a qualidade, confiabilidade e manutenibilidade de tais componentes de software. Para avaliar e monitorar um software e sua comunidade de desenvolvimento se fazem necessárias ferramentas que auxiliem no processo. Dessa forma, neste trabalho apresentamos uma ferramenta, denominada HealthyEnv, que auxilia na avaliação da saúde de projetos de software, provendo não apenas uma série de métricas computadas automaticamente a partir do repositório de software, como também valores de referência para auxiliar na interpretação dos resultados.

Link para o vídeo: https://youtu.be/oymuwH3NKCY

CCS CONCEPTS

• **Software and its engineering**;

KEYWORDS

Software repository mining, machine learning, software quality.

ACM Reference Format:
Diego Winter, Guilherme Avelino e Charles Miranda. 2022. HealthyEnv: a tool to assist in health assessment of software repositories. In *XXXVI Brazilian Symposium on Software Engineering (SBES 2022), October 5–7, 2022, Virtual Event, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3555228.3555279

SBES '22, October 5–7, 2022, Virtual Event, Brazil

ACM ISBN 978-1-4503-9735-3/22/10.
https://doi.org/10.1145/3555228.3555279

1 INTRODUÇÃO

As organizações de desenvolvimento de software incorporam projetos de terceiros em seus projetos, pois é um modelo viável de produção de software [6]. Entretanto, a qualidade do software incorporado pode diminuir ou comprometer a saúde do projeto que o está incorporando, criando a preocupação com o gerenciamento da saúde de um software a fim de evitar problemas que poderiam levar a falhas graves em sistemas controlados por eles [9].

Nesse contexto, é de extrema importância saber como tais softwares são desenvolvidos e se são adequados para o uso ou integração em outro software. Observar o que engloba o ciclo de vida e a motivação dos contribuidores ajuda a entender a importância de uma comunidade que mantém um projeto. Cada contribuidor assume um papel e, dependendo da organização de tais papéis, muito pode se dizer sobre a saúde de um projeto [3]. Definir a saúde de um software significa verificar se ele se enquadra em padrões de qualidade e manutenibilidade. Esta avaliação muitas vezes não pode ser feita de modo manual, devido à complexidade e diversidade de projetos, havendo a necessidade de ferramentas para esta finalidade.

Apesar de existirem ferramentas capazes de extrair informações sobre projetos de software, como valores obtidos a partir de aplicação de métricas de qualidade de software, há a ausência de valores de referência que nos auxiliem no momento da avaliação. Um exemplo é o conjunto de ferramentas do GrimoireLab [2], do projeto CHAOSS [1], que fornece ferramentas para a mineração, enriquecimento e visualização de dados de repositórios de software, contudo não apresentando valores de referência dos dados que estão sendo visualizados. Valores de referência são importantes pois servem como base para avaliar se o valor obtido para uma dada métrica é adequado ou não. Por exemplo: ao realizar um exame de sangue é comum que para cada característica avaliada seja fornecido valores de referência, os quais variam de acordo com o perfil da pessoa. Com tais valores de referência, é possível observar que um valor de colesterol total de 180mg/dL está dentro da faixa desejável para um adulto maior de 20 anos, mas é considerado alto para uma pessoa com idade inferior. A mesma lógica se encaixa na avaliação da saúde de projetos de software. A definição de tais valores de referência requer o estudo de uma grande quantidade de projetos e é importante que considere as diferentes categorias de projetos de software de forma a prover valores de referência específicos para o grupo em que o projeto em avaliação se encaixe.

Dado o problema, nesta ferramenta propomos a aplicação de métricas em um conjunto de projetos de software, compondo um

dataset onde, dado um projeto selecionado para análise, deve ser obtido um grupo de projetos similares para compor valores de referência para a avaliação. O principal objetivo destes valores de referência é auxiliar na interpretação dos resultados ao permitir comparar os valores obtidos com os de projetos populares semelhantes. A ferramenta também inclui um dashboard interativo para que as análises possam ser feitas de forma intuitiva.

Este trabalho está organizado como a seguir. A Seção 2 apresenta conceitos relacionados ao trabalho. A Seção 3 apresenta a ferramenta proposta e todos os detalhes sobre seu funcionamento e implementação. Na Seção 4 temos exemplos de uso da ferramenta. A Seção 5 apresenta alguns estudos relacionados ao tema. E por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são introduzidos alguns conceitos que serão abordados nas seções seguintes, para facilitar o entendimento.

2.1 Métricas de software

Métricas de software são responsáveis por fazer medições em características de um software. Elas desempenham um papel importante ao auxiliar no controle de qualidade e estimativa de esforço. Porém o potencial da medição de software nem sempre é explorado da maneira correta, sendo necessária uma melhoria de como essa medição está sendo realizada [4]. Em outras palavras, deve ser proposto uma forma mais adequada de aproveitar melhor o que pode ser proporcionado pelas métricas de software.

Um projeto que possui bastante importância no campo da avaliação de repositórios de software é o CHAOSS - Community Health Analytics Open Source Software [1], mantido pela Linux Foundation. O CHAOSS nos traz um conjunto de ferramentas e métricas para auxiliar na definição da saúde de projetos e comunidades de desenvolvimento. Algumas métricas podem nos trazer informações sobre questões mais técnicas do projeto, enquanto outras podem focar mais no processo de desenvolvimento e na comunidade em si.

Várias contribuições do CHAOSS serão utilizadas nesta ferramenta, dentre algumas ferramentas e métricas formalizadas por eles. As métricas do CHAOSS que iremos utilizar estão detalhadas na Tabela 1. Algumas métricas podem abranger muitos pontos. Por exemplo, a métrica *Code changes lines* obtém diversas informações sobre linhas de código. Em casos como esse, as métricas foram subdivididas, como por exemplo, *Code changes lines: lines added*, para que esses aspectos possam ser avaliados separadamente.

2.2 Aprendizado de Máquina

Aprendizado de Máquina (AM) é um subcampo da Inteligência Artificial que faz o uso de algoritmos para utilizar informações passadas a estes, a fim de fazê-los aprender sem programá-los para tal. Existem três principais tipos de aprendizado: por reforço, supervisionado e o não supervisionado. No aprendizado por reforço, o aprendizado ocorre após uma série de reforços, sendo recompensas ou punições. Já o supervisionado tem como base a observação das informações passadas na entrada e as informações de saída. Por fim, que é o tipo utilizado neste trabalho, o aprendizado não supervisionado gira em torno das informações de entrada [8].

O AM não supervisionado tem como sua principal atividade a clusterização que é, em resumo, a detecção de grupos nas entradas, que sejam potencialmente úteis. No nosso caso em específico, criar estes grupos nos auxilia a identificar projetos semelhantes para que assim possamos utilizar na avaliação as informações obtidas.

3 FERRAMENTA PROPOSTA

A HealthyEnv foi desenvolvida para auxiliar na visualização da saúde de projetos de software através de valores de referência obtidos pela comparação com projetos semelhantes. Nosso dataset tem a possibilidade de ser expandido através de submissões pela comunidade que deseja avaliar um projeto que não esteja nele. A avaliação acontece através de gráficos do tipo *boxplot*, onde neles são mostrados algumas informações que atuam como valores de referência, como média, mediana e quartis. O principal público-alvo para a HealthyEnv são desenvolvedores de software que desejam avaliar seus próprios projetos ou avaliar projetos que desejam integrar em seus programas. A ferramenta pode ser utilizada através do endereço https://healthyenv.vercel.app e um exemplo do uso se encontra em um vídeo no endereço https://youtu.be/oymuwH3NKCY.

O projeto está estruturado em uma parte responsável pelo fornecimento e processamento das informações (back-end) e uma parte responsável pela exibição dos resultados (front-end). O back-end consiste de um web service que deve conter as informações do dataset e que deve executar um algoritmo baseado em algoritmos de AM conhecidos que obtém os projetos semelhantes através do critério de distância. Os projetos semelhantes, assim como o projeto a ser analisado é retornado com os valores para suas métricas para a análise. Já o front-end consiste em uma aplicação Web para auxiliar no processo de análise. As subseções seguintes irão abordar em detalhes as partes que compõem a arquitetura da ferramenta.

3.1 Construção do dataset

Para a HealthyEnv, o dataset é um dos componentes principais, pois é nele que as informações sobre os repositórios são armazenadas. Dentre as informações que são armazenadas, temos os atributos que são utilizados para a identificação de semelhança entre os projetos (linhas de código, forks, issues abertas, contribuidores e commits) e também os valores de cada métrica, tudo isso para cada projeto. Como o foco principal da ferramenta é a análise comparativa, manter essas informações internamente sem que seja necessário requisitar externamente (do GitHub, por exemplo) é essencial para permitir que as avaliações sejam realizadas.

O dataset iniciou com 50 projetos de cada uma das 10 linguagens mais populares segundo o relatório anual do GitHub, totalizando 500 projetos, como um passo inicial para validação da ideia da ferramenta. As linguagens inclusas são: JavaScript, Python, Java, TypeScript, C#, PHP, C++, Shell, C e Ruby. A medida que projetos vão sendo submetidos para análise pela comunidade, esse dataset se expande. Todos os projetos do dataset inicial passaram por um processo de enriquecimento, onde as informações vindas dos logs do Git e do repositório do GitHub são armazenadas no Elasticsearch. Projetos submetidos pela comunidade também devem passar por este processo antes de serem incluídos no dataset da HealthyEnv.

Tabela 1: Metricas extraídas dos repositórios

Métrica	*Working group*	Descrição
Truck Factor	Risk	Calcula o fator de distribuição do trabalho entre os contribuidores do projeto
Code changes commits	Evolution	Obtém a quantidade de total de commits
Code changes commits: average lines per commit	Evolution	Obtém uma média de linhas por commit
Code changes commits: average files per commit	Evolution	Obtém uma média de arquivos por commit
Code changes lines: lines added	Evolution	Obtém a quantidade total de linhas adicionadas
Code changes lines: lines removed	Evolution	Obtém a quantidade total de linhas removidas
Median issues age	Evolution	Verifica a mediana de tempo que uma issue permaneceu aberta
Average issues age	Evolution	Verifica o tempo médio que as issues permanecem abertas
Max issues age	Evolution	Obtém o tempo máximo que uma issue permanece aberta
Issues active	Evolution	Obtém a quantidade de issues abertas no projeto
Issues closed	Evolution	Obtém a quantidade de issues fechadas no projeto
Median time to first response	Common	Calcula a mediana de tempo para uma primeira resposta em uma issue
Average time to first response	Common	Calcula a média de tempo para uma primeira resposta em uma issue
Median time to close	Common	Calcula a mediana de tempo para fechar uma issue
Average time to close	Common	Calcula a média de tempo para fechar uma issue
Contributors	Common	Obtém o número de contribuidores do projeto

A partir do dataset enriquecido, as métricas selecionadas para compor a ferramenta são extraídas e armazenadas, para serem utilizadas posteriormente nas análises. As informações finais (informações gerais dos repositórios e valores de métricas) são armazenadas em um banco de dados MySQL para posteriores consultas pela API da ferramenta, que deverá fornecer dados processados para a aplicação Web.

É interessante detalhar que o processo de enriquecimento do dataset é feito utilizando a ferramenta GrimoireLab [2] do projeto CHAOSS. Esta ferramenta se trata de um conjunto de outras ferramentas capazes de fazer a mineração de repositórios e análise visual das informações obtidas, apoiando a análise de software. A Figura 1 detalha o processo de criação do dataset inicial da HealthyEnv, processo que também deve valer para projetos que são submetidos pela comunidade.

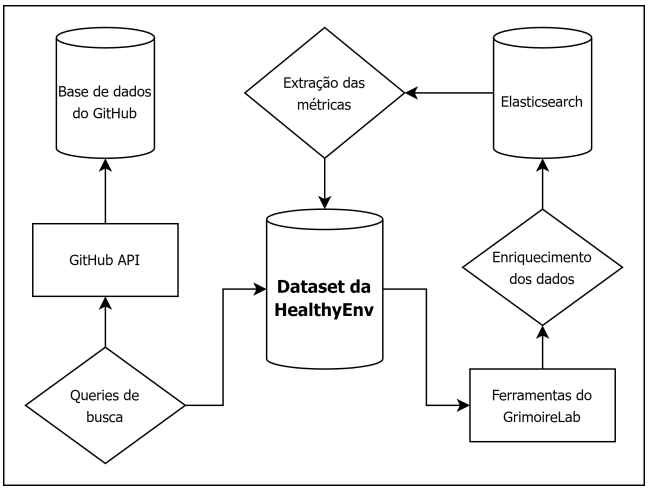

Figura 1: Diagrama da construção do dataset inicial.

3.2 Algoritmo proposto

O algoritmo responsável por fazer a obtenção de projetos semelhantes tem o mesmo fundamento do principal algoritmo de Aprendizado de Máquina não supervisionado, o k-means, que é obter a semelhança através da distância euclidiana entre os pontos. Como possuímos 5 atributos para serem utilizados na obtenção da semelhança entre os projetos (linhas de código, forks, issues abertas, contribuidores e commits), utilizamos um algoritmo de redução de dimensionalidade, o PCA (Principal Component Analysis), para que possamos reduzir para duas *features* e permitir a análise em um plano bi-dimensional. Os dados de tais atributos, antes de passar pela redução de dimensionalidade, passam por uma padronização, para que haja equilíbrio durante a aplicação do algoritmo sem que alguns pontos sejam favorecidos por grandes diferenças nos valores.

A ideia central do algoritmo é fazer o cálculo da distância entre um ponto selecionado e os demais pontos, onde este ponto selecionado deve representar o repositório que vai ser analisado. A partir de todas as distâncias obtidas, deve ser selecionado apenas os n pontos mais próximos, sendo esse valor de n a quantidade desejada de repositórios similares para compor os valores de referência para a avaliação.

Como dito anteriormente, o cálculo da distância se trata da distância euclidiana entre os pontos. A fórmula que mostra como este cálculo é feito é a seguinte:

$$d_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

O algoritmo segue alguns passos para que a obtenção das distâncias aconteça. Os passos executados pelo algoritmo e o pseudocódigo (Algoritmo 1) são mostrados a seguir.

- Define-se um valor para n, que será a quantidade de repositórios próximos para utilizar na análise, e um repositório do dataset para analisar;
- Para cada repositório que não seja o passado para análise, calcular a distância euclidiana entre este repositório e o repositório a ser analisado, conforme a fórmula mostrada anteriormente;

- Faz-se a ordenação de todas as distâncias obtidas em ordem crescente;
- Seleciona-se os n primeiros repositórios da lista ordenada. Estes devem representar os repositórios mais semelhantes ao selecionado.

Algorithm 1: Algoritmo proposto

```
Input  : D: Map representando o dataset, R: repositório
         para análise, N: valor
Output : Lista de N repositórios de D mais semelhantes a R
1 begin
2     for i = 0 até D.length do
3         if D[i] ≠ R then
4             D[i]['distância'] ← distânciaEuclidiana(D[i], R);
5         end
6     end
7     D ← D.sortBy(['distância']);
8     return D.slice(0, N);
9 end
```

3.3 Arquitetura da ferramenta

Como introduzido anteriormente, a ferramenta está dividida entre um back-end e um front-end. O back-end tem a responsabilidade de fornecer as informações para que o front-end as exiba. O back-end consiste de uma API REST que tem acesso direto ao banco de dados (dataset) da ferramenta, e possui as funcionalidades de obtenção de projetos próximos para que seja feito o retorno de tudo que é necessário para fazer a análise de um determinado projeto. Já o front-end consiste em uma aplicação Web para que as análises sejam feitas diretamente pelo browser.

Para a implementação do back-end, optamos pela linguagem Python, pois foram utilizadas funcionalidades da biblioteca scikit-learn para o anteriormente citado pré-processamento dos dados. Para que o back-end seja uma API REST, foi utilizado o framework Flask, junto com o ORM (Object-Relational Mapping) SQLAlchemy para que dados pudessem ser adicionados e recuperados do banco de dados.

Já no front-end, utilizando a linguagem TypeScript com o framework Next.js (que utiliza a biblioteca React), implementamos uma aplicação Web que é capaz de listar os datasets disponíveis e seus repositórios, bem como iniciar uma análise. A análise é realizada por gráficos do tipo *boxplot*, que são capazes de mostrar a variação dos dados através de caixas que representam os quartis, bem como extremos. Mais detalhes que mostram a ferramenta em ação podem ser vistos na Seção 4.

Na Figura 2 temos um panorama geral da arquitetura da ferramenta. O front-end possui três atividades principais: a submissão de um projeto, a seleção de um projeto para análise e apresentação dos resultados. Ao submeter um repositório, é gerada uma solicitação para que no back-end possa ser feito todo o procedimento de mineração pela ferramenta, onde os dados obtidos devem ser armazenados no dataset. Quando selecionado um projeto dentre todos os disponíveis na ferramenta, inclusive os adicionados pela comunidade, o back-end fica responsável por consultar o banco de dados para que possam ser obtidos os repositórios de referência. Após esse processo, estas informações são passadas ao front-end para que os resultados sejam apresentados.

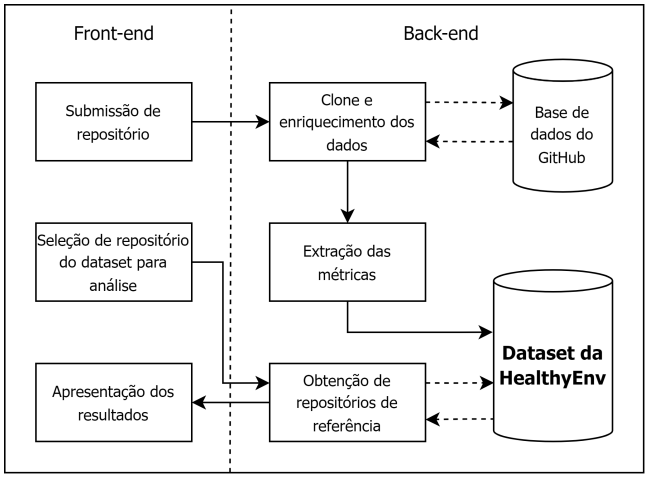

Figura 2: Esquema de funcionamento da ferramenta.

Na Tabela 2, temos uma visão geral dos endpoints disponíveis na API para fazer a consulta de informações. A API pode trabalhar com múltiplos datasets, por isso a existência de um endpoint que ajuda a identificá-los. Esta funcionalidade traz flexibilidade para que seja possível trabalhar com diferentes grupos de repositórios no futuro, fazendo com que a expansão deles aconteça por determinada característica, como por exemplo, de uma linguagem específica ou que utiliza um framework específico.

4 EXEMPLO DE USO

A Figura 3 apresenta a página inicial da HealthyEnv, contendo algumas informações iniciais e algumas opções. O usuário deve prosseguir escolhendo alguma opção situada na barra superior ou no botão "Explorar datasets"

Figura 3: Página inicial da HealthyEnv.

Como pode ser visto na Figura 4, na tela *Datasets e análise*, podemos fazer a seleção de um dataset para visualizar os repositórios que estão disponíveis nele. Ao lado do seletor de datasets, temos a opção de atribuir a quantidade de repositórios próximos ao que irá

Tabela 2: Endpoints disponíveis na API

Endpoint	Método	Descrição
/datasets	GET	Lista os datasets disponíveis.
/datasets/<id>/repos	GET	Lista os repositórios por id do dataset.
/datasets/<id>/cluster/<repo>?near_n=<n>	GET	Lista o grupo de n repositórios próximos de repo com informações para análise.
/datasets/<id>/submit	POST	Solicita a inclusão de um repositório
/metrics	GET	Lista as métricas disponíveis.
/metrics/categories	GET	Lista as categorias de métricas disponíveis.
/requests/<email>	GET	Lista as solicitações feitas por um email

ser selecionado para a realização da análise. No canto direito, ainda na mesma faixa de opções, temos uma caixa de texto para filtragem dos repositórios que estão carregados abaixo. Cada repositório da lista é exibido com seu nome no padrão usuário/repositório, com algumas das suas características sendo exibidas abaixo.

Figura 4: Tela de datasets e análise.

Com a quantidade de repositórios definida e um repositório escolhido na lista, podemos clicar nele para nos direcionarmos à página de análise. Conforme a Figura 5, é nesta página onde podemos visualizar todas as informações sobre o grupo obtido, localização do repositório no plano, seus semelhantes e os considerados não semelhantes pelo critério de distância. Abaixo, ao rolar a página, podemos ver as métricas aplicadas no grupo obtido, separadas por categorias, exibidas na forma de gráficos do tipo *boxplot*. Este tipo de gráfico nos permite visualizar a distribuição dos pontos e alguns valores, como mediana e quartis, para usarmos como valores de referência. Os gráficos mostram também onde está situado o projeto analisado, representado pelo ponto de cor laranja, conforme pode ser visto nos exemplos exibidos na figura 6. Quando uma métrica está acima ou abaixo da mediana, dependendo da métrica, o gráfico fica de cor esverdeada, caso contrário, de cor avermelhada. Isso auxilia ao usuário a ter um feedback instantâneo sobre a situação do projeto que ele escolheu para análise perante os semelhantes.

Vale ressaltar que algumas métricas são boas com valores altos, outras com valores baixos. A HealthyEnv mostra o feedback de cores levando em consideração isto. Se uma métrica, como por exemplo o

Figura 5: Tela de análise de um repositório.

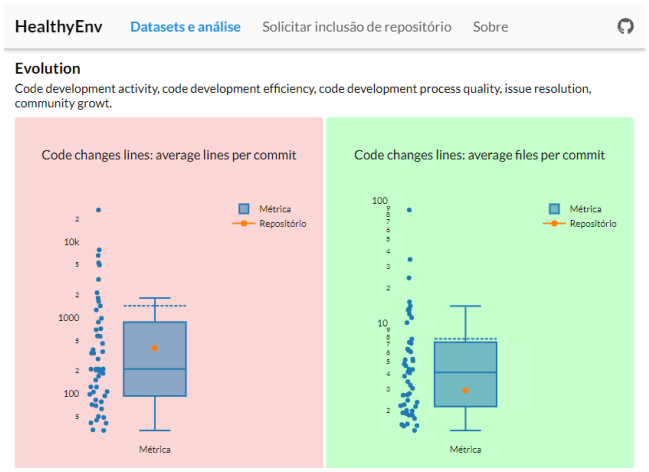

Figura 6: Gráficos de métricas aplicadas ao grupo obtido.

Truck Factor, que é uma métrica boa com valores altos, estiver com o valor 3 e a mediana com o valor 2, esta métrica é considerada boa para a HealthyEnv. Se o valor estivesse 1, estaria abaixo da mediana, sendo considerado pela ferramenta um ponto de melhoria. Nem toda métrica de cor avermelhada significa algo totalmente ruim, mas sim um ponto de atenção ao fazer a comparação.

Na barra superior, ao clicar em "Solicitar inclusão de repositório"o usuário chegará em uma tela com a possibilidade de submeter um repositório para que seja processado e adicionado pela equipe que mantém a ferramenta. Nessa tela, podemos também inserir o e-mail utilizado no cadastro da submissão para que possa ser acompanhado

a situação de todas as submissões enviadas com tal e-mail. Ao utilizar a funcionalidade de acompanhar as solicitações, o usuário é redirecionado para uma nova tela, conforme a Figura 7, onde ele consegue visualizar todas as suas solicitações e os status delas. As solicitações podem ter os seguintes status: Recebida - quando a solicitação foi registrada, Em andamento - quando o repositório está passando pelo processo de mineração para ser incluído no dataset, e Finalizada - quando já foi enviado um feedback para o e-mail sobre o sucesso ou falha na inclusão. A HealthyEnv inicialmente processa essas solicitações de modo manual pela equipe, mas é planejado que todo o processo seja automatizado e controlado por contas de usuário.

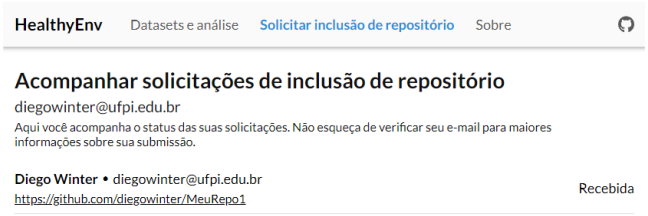

Figura 7: Página de acompanhamento de submissões.

Se o usuário desejar visualizar mais informações sobre a ferramenta, bem como as licenças de código aberto utilizadas, basta utilizar a opção "Sobre"da barra superior. Por fim, para visualizar o repositório da HealthyEnv, basta clicar no ícone do GitHub no final da barra superior.

5 TRABALHOS RELACIONADOS

Dentre os trabalhos relacionados, o mais interessante de ser mencionado aqui é o trabalho realizado por Goggins e outros [5]. Ele contextualiza o uso de métricas para avaliação de repositórios de software, questionando as limitações do modo que os projetos eram comumente avaliados, que consistia em ser apenas a observação de quantias relacionadas com a atividade de um projeto. Isso traz um questionamento que pode fazer com que muitos desenvolvedores se identifiquem, que é a escolha de um componente de software observando números, como popularidade em um *package manager* por exemplo. O estudo reforça a necessidade de adoção de melhores formas de avaliação da saúde de projetos ao examinar o CHAOSS em seus anos iniciais.

Oliveira e outros [7] mostram uma abordagem para a extração de limiares relativos de métricas de código-fonte, visando dar força ao desafio que é a definição de limiares na definição de métricas como um instrumento para medição de qualidade no software. É ressaltado que, apesar de que a proposição de métricas existe desde o início das primeiras linguagens de programação para medir propriedades do código, métricas não influenciavam positivamente na qualidade pela ausência de limiares aceitáveis. Este trabalho possui semelhança com a presente ferramenta no sentido de que também tem o objetivo de dar mais sentido ao uso de métricas, em vez de apenas observá-las isoladamente.

A HealthyEnv se diferencia dos trabalhos relacionados por introduzir uma outra abordagem para dar mais significado à aplicação de métricas de software. O trabalho de Oliveira [7] também é focado nesta direção, porém de um modo diferente da HealthyEnv, que utiliza conceitos de AM como apoio para chegar no objetivo. Além disso a ferramenta aqui apresentada também permite que as análises com as informações obtidas sejam feitas de forma intuitiva, com visualizações interativas e de feedback imediato.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresenta uma ferramenta com o propósito de auxiliar na análise de software de modo comparativo, através da mineração de informações de projetos abertos. Mesmo que não possa definir com precisão a saúde de um repositório, a ferramenta é capaz de fazer com que o desenvolvedor tenha uma ideia de como este repositório está diante dos semelhantes, sendo útil para que possa avaliar projetos próprios ou de terceiros, mediante a submissão para expansão do dataset.

A ferramenta pode e irá ser expandida para facilitar ainda mais o processo de submissão, bem como a qualidade dos feedbacks que podem ser apresentados durante a análise. Incluir ainda mais características de outros algoritmos de clusterização, ou até mesmo outras formas de pré-processamento de dados para se obter várias outras perspectivas é um ponto considerável para elevar a precisão. Analisar software de maneira automatizada e de maneira perfeita ainda é um desafio e cada trabalho que é produzido na área ajuda a chegar mais próximo desse objetivo.

Para trabalhos futuros, uma melhoria no algoritmo aqui proposto seria interessante para levar em consideração mais atributos que podem ser extraídos dos projetos, melhorando ainda mais a identificação da similaridade entre eles. Também fica como sugestão um trabalho que possa avaliar a relevância da ferramenta através de um estudo de caso com desenvolvedores.

ACKNOWLEDGMENTS

Agradecimentos à UFPI (PIBIC) pela bolsa concedida para o desenvolvimento da pesquisa.

REFERÊNCIAS

[1] [n.d.]. Community Health Analytics Open Source Software. https://chaoss.community/. Acessado em 14/02/2022.
[2] [n.d.]. GrimoireLab: free, libre, open source tools for software development analytics. https://chaoss.github.io/grimoirelab/. Acessado em 07/02/2022.
[3] Kevin Crowston and James Howison. 2006. Assessing the health of open source communities. *Computer* 39 (2006).
[4] Norman Fenton and Paul Krause. 2002. Software measurement: Uncertainty and causal modeling. *IEEE software* 19 (2002).
[5] Sean Goggins, Kevin Lumbard, and Matt Germonprez. 2021. Open Source Community Health: Analytical Metrics and Their Corresponding Narratives. *2021 IEEE/ACM 4th International Workshop on Software Health in Projects, Ecosystems and Communities (SoHeal)* (2021).
[6] Damrongsak Naparat, Patrick Finnegan, Michael Cahalane, et al. 2015. Healthy community and healthy commons:'Opensourcing'as a sustainable model of software production. *Australasian Journal of Information Systems* 19 (2015).
[7] Paloma Oliveira, Marco Tulio Valente, and Fernando Paim Lima. 2014. Extracting Relative Thresholds for Source Code Metrics. *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)* (2014).
[8] Stuart Russell and Peter Norvig. 2002. *Artificial intelligence: a modern approach.*
[9] Ashok Srivastava and Johann Schumann. 2013. Software health management: a necessity for safety critical systems. *Innovations in Systems and Software Engineering* 9 (2013).